ONVIF®

Profile C Client Test Specification

Version 22.06

June 2022

REVISION HISTORY

Vers.	Date	Description
22.06	Dec 27, 2022	Profile Normative Reference were removed from test cases according to #364
21.12	Oct 12, 2021	The following was done according to #425: Check Condition based on Device Features of Discovery feature was changed from 'All' to 'Discovery'
21.06	Jun 03, 2021	The following was updated according to #325: SETSYNCHRONIZATIONPOINT-1 test name was changed from SET SYNCHRONIZATION POINT to SET SYNCHRONIZATION POINT (EVENT SERVICE). Set Synchronization Point feature was renamed to Set Synchronization Point (Event Service)
21.06	Jan 13, 2021	In the scope of #364 format of the following features were updated to show dependent test cases inside feature: Keep Alive for Pull Point Event Handling Access Control Decisions Area Information - Configuration Change Notifications Duress Notifications
20.12	Dec 8, 2020	DEVICEDISCOVERYTYPEFILTER-1 DEVICE DISCOVERY TYPE FILTER was updated according to #406: Types value check was updated to accept QName list instead of one QName value.
20.12	Nov 12, 2020	The following was done according to #399: System Date and Time Configuration: Check Condition based on Device Features was updated
20.12	Oct 27, 2020	The following was done according to #394: Check Condition based on Device Features of Network Configuration feature was changed from 'All' to 'Network Configuration'
20.12	Oct 27, 2020	The following was done according to #393: Check Condition based on Device Features of User Handling feature was changed from 'All' to 'User Configuration'
20.12	Aug 31, 2020	Set Synchronization Point Feature: Check Condition based on Device Features was changed according to #325.
20.12	Aug 31, 2020	Unsubscribe Feature: Check Condition based on Device Features was changed according to #325.
20.12	Aug 31, 2020	Keep Alive for Pull Point Event Handling Feature: Check Condition based on Device Features was changed according to #325.

20.12	Aug 31, 2020	Event Handling Feature: Check Condition based on Device Features was changed according to #325.
19.12	Dec 10, 2019	The following was done according to #355: ACCESSPOINTINFORMATION-1 LISTING OF ACCESS POINTS (description was updated with namespaces)
19.12	Sep 18, 2019	The following was done according to #325: Scope\Supplementary Features and Test Cases sections was added. Supplementary Features and Test Cases sections was added.
19.12	Aug 13, 2019	The following was done according to #325: EVENTHANDLING-3 METADATA STREAMING test was removed from Event Handling Feature and moved to Metadata Streaming Using Media2. Test case ID was changed to MEDIA2_METADATASTREAMING-1. Event Handling will use link to this test. EVENTHANDLING-4 METADATA STREAMING USING MEDIA was added for Profile S Devices.
19.12	Sep 6, 2019	DEVICEDISCOVERYTYPEFILTER-1 DEVICE DISCOVERY TYPE FILTER was updated according to #323: Unnecessary step with check that ProbeMatch is sent to Client IP address was removed.
19.12	Aug 26, 2019	The following was done according to #323: Area Information - Configuration Change Notifications feature and scope was extracted from Configuration Change Notifications feature. Scope\Area Information - Configuration Change Notifications section was added. Area Information - Configuration Change Notifications Test Cases section was added. Configuration Change Notifications section was removed. Configuration Change Notifications Test Cases section was removed.
19.12	Aug 26, 2019	The following was done according to #323: Door Information - Configuration Change Notifications feature and scope was extracted from Configuration Change Notifications feature. Scope\Door Information - Configuration Change Notifications section was added. Door Information - Configuration Change Notifications Test Cases section was added. Configuration Change Notifications section was removed. Configuration Change Notifications Test Cases section was removed.

19.12	Aug 26, 2019	The following was done according to #323: Access Point Information - Configuration Change Notifications feature and scope was extracted from Configuration Change Notifications feature. Scope\Access Point Information - Configuration Change Notifications section was added. Access Point Information - Configuration Change Notifications Test Cases section was added. Configuration Change Notifications section was removed. Configuration Change Notifications Test Cases section was removed.
19.12	Aug 13, 2019	The following was done according to #323: Area Information feature and scope was extracted from System Component Information feature. Scope\AreaInformation section was added. Area Information Test Cases section was added. System Component Information section was removed. System Component Information Test Cases section was removed.
19.12	Aug 13, 2019	The following was done according to #323: Door Information feature and scope was extracted from System Component Information feature to be reused for Profile D. Scope\Door Information section was added. Door Information Test Cases section was added.
19.12	Aug 13, 2019	The following was done according to #323: Access Point Information feature and scope was extracted from System Component Information feature to be reused for Profile D. Scope\Access Point Information section was added. Access Point Information Test Cases section was added.
19.12	Aug 14, 2019	The following was done according to #341: HTTP Digest section and HTTP Digest Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Capabilities section and Capabilities Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Get Services section and Get Services Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.

19.12	Aug 14, 2019	The following was done according to #341: Event Handling section and Event Handling Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Keep Alive for Pull Point Event Handling section and Keep Alive for Pull Point Event Handling Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Discovery section and Discovery Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Device Discovery Type Filter section and Device Discovery Type Filter Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Network Configuration section and Network Configuration Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: System section and System Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: User Handling section and User Handling Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: IP Address Filtering section and IP Address Filtering Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Persistent Notification Storage Retrieval section and Persistent Notification Storage Retrieval Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Get Services with Capabilities section and Get Services with Capabilities Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341:

		Set Synchronization Point section and Set Synchronization Point Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Unsubscribe section and Unsubscribe Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: System Date and Time Configuration section and System Date and Time Configuration Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Hostname Configuration section and Hostname Configuration Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: DNS Configuration section and DNS Configuration Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Network Protocols Configuration section and Network Protocols Configuration Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile C Client Test Specifications.
19.06	Jun 14, 2019	The following was done according to #309: 'Validated Feature' section for each feature updated to be synchronized with feature ID used in feature list. 'Feature Under Test' section for each test case updated to be synchronized with sub-feature ID used in feature list. 'Validated Feature List' test case section removed.
18.06	Jun 21, 2018	Reformatting document using new template
18.06	Apr 05, 2018	'Required Number of Devices Summary' Annex added according to #241
18.06	Feb 16, 2018	The following were updated in the scope of #241: Feature Level Requirement (updated with new rules) Each Feature Level Requirement (updated with Check Condition based on Device Features and Required Number of Devices)
17.06	Jun 15, 2017	Links in Normative references section were updated.
16.07	Apr 19, 2016	• Test cases about specific event were removed: SYSTEMCOMPONENTSTATE-1, SYSTEMCOMPONENTSTATE-2, ACCESSCONTROLDECISIONS-1, ACCESSCONTROLDECISIONS-2, ACCESSCONTROLDECISIONS-3, ACCESSCONTROLDECISIONS-4,

		ACCESSCONTROLDECISIONS-5, ACCESSCONTROLDECISIONS-6, ACCESSCONTROLDECISIONS-7, ACCESSCONTROLDECISIONS-8, ACCESSCONTROLDECISIONS-9, CONFIGURATIONCHANGENOTIFICATION-1, CONFIGURATIONCHANGENOTIFICATION-2, CONFIGURATIONCHANGENOTIFICATION-3, CONFIGURATIONCHANGENOTIFICATION-4, CONFIGURATIONCHANGENOTIFICATION-5, CONFIGURATIONCHANGENOTIFICATION-6, DURESS-1. • System Component State scenario updated • Access Control Decisions scenario updated • Configuration Change Notifications scenario updated • Duress Notifications scenario updated
16.07	Apr 05, 2016	The description about structure and hierarchy was replaced for the test cases: SYSTEMCOMPONENTINFORMATION-1, SYSTEMCOMPONENTINFORMATION-2, SYSTEMCOMPONENTINFORMATION-3, SYSTEMCOMPONENTSTATE-1, SYSTEMCOMPONENTSTATE-2, DOORCONTROL-1, DOORCONTROL-2, DOORCONTROL-3, DOORCONTROL-4, DOORCONTROL-5, DOORCONTROL-6, DOORCONTROL-7, ACCESSPOINTCONTROL-1, EXTERNALAUTHORIZATION-2 Old description: Client %COMMAND NAME% request message is a well-formed SOAP request (refer to onvif.xsd) AND Client %COMMAND NAME% request message has a proper hierarchy (refer to %SERVICE%.wsdl) AND New description: Client %COMMAND NAME% request messages are valid according to XML Schemas listed in Namespaces AND Client %COMMAND NAME% request in Test Procedure fulfills the following requirements: The following steps was removed because the requirements are fullfield by XML Schemas validation: • SYSTEMCOMPONENTSTATE-1: [S6] "<PullMessages>" includes tag: "<Timeout>" AND [S7] "<PullMessages>" includes tag: "<MessageLimit>" AND • SYSTEMCOMPONENTSTATE-2: [S6] "<PullMessages>" includes tag: "<Timeout>" AND [S7] "<PullMessages>" includes tag: "<MessageLimit>" AND • EXTERNALAUTHORIZATION-2: [S3] "<ExternalAuthorization>" includes tag: "<Decision>" AND [S4] "<Decision>" contains value EITHER ("Granted" OR "Denied") AND
16.07	Mar 14, 2016	www.onvif.org was removed from Copyright section.
16.01	Dec 2, 2015	General item (Test Owerview) was added

		Minor updates in formatting, typos and terms according review result of other Client Test Specifications EXTERNALAUTHORIZATION-3 was removed. Related feature was chnaged in accordance. EXTERNALAUTHORIZATION-1 was updated to include new pre-requisite and new test style was upplied.
16.01	Nov 20, 2015	Change according to # 67:Expected Scenarios Under Test of Access Control Decisions, Configuration Change Notifications, Duress Notifications were updated: dependence on Device features were added. New Note was added into corresponding test cases.
16.01	Sep 28, 2015	Added Access Control Decisions Test Cases, Configuration Change Notifications, Duress Notifications Test Cases sections
15.06	Jun 10, 2015	No major changes were made, just minor formatting fixes.
15.05	May 20, 2015	No major changes were made, just minor grammatical corrections.
15.03	Mar 20, 2015	Added External Authorization Test Cases section.
15.02	Feb 19, 2015	Pass criteria in SYSTEMCOMPONENTSTATE-1 and 2 test cases have been updated (added additional criteria for checking <TopicExpression> tag value).
14.12	Dec 11, 2014	Fixed typos and inconsistencies.
1.0	Oct 16, 2014	Initial version

Table of Contents

1 Introduction

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

This particular document defines test cases required for testing Profile C features of a Client application e.g. System Component Information, System Component State, Door Control and Access Point Control. It also describes the test framework, test setup, prerequisites, test policies needed for the execution of the described test cases.

1.1 Scope

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

The principal intended purposes are:

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

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

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

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

1.2 Test Cases for Profile Mandatory Features

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

1.2.1 HTTP Digest

HTTP Digest section defines security mechanism for HTTP Digest Authentication.

1.2.2 Capabilities

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

1.2.3 Get Services

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

1.2.4 Event Handling

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

The event handling test cases cover the following mandatory interfaces:

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

1.2.5 Keep Alive for Pull Point Event Handling

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

1.2.6 Access Point Information

Access Point Information section specifies Client ability to request lists of Access Points from Device.

1.2.7 Door Information

Door Information section specifies Client ability to request lists of Doors from Device.

1.2.8 Area Information

Area Information section specifies Client ability to request lists of Areas from Device.

1.2.9 System Component State

System Component State section specifies Client ability to request information about the state of Access Points (enabled/disabled) and Doors (locked, unlocked, etc.).

1.2.10 Door Control

Door Control section specifies Client ability to to control Doors (access door, lock door, unlock door, etc.).

1.2.11 Access Control Decisions

Access Control Decisions section specifies Client ability to receive notifications about access decisions related to Access Control.

1.2.12 Access Point Information - Configuration Change Notifications

Access Point Information - Configuration Change Notifications section specifies Client ability to receive Access Points configuration change notifications.

1.2.13 Door Information - Configuration Change Notifications

Door Information - Configuration Change Notifications section specifies Client ability to receive Doors configuration change notifications.

1.2.14 Area Information - Configuration Change Notifications

Area Information - Configuration Change Notifications section specifies Client ability to receive Areas configuration change notifications.

1.2.15 Duress Notifications

Duress Notifications section specifies Client ability to receive notifications about duress situation.

1.3 Test Cases for Profile Conditional Features

This section defines test cases which are conditional for Profile C Client conformance.

1.3.1 Discovery

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

1.3.2 Device Discovery Type Filter

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

1.3.3 Network Configuration

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

1.3.4 System

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

1.3.5 User Handling

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

1.3.6 IP Address Filtering

IP Address Filtering section defines Client ability to manage IP address filters on Device.

1.3.7 Persistent Notification Storage Retrieval

Persistent Notification Storage Retrieval section defines Client ability to seek stored events in Device.

1.3.8 Access Point Control

Access Point Control section specifies Client ability to control Access Points (enabled/disabled).

1.3.9 External Authorization

External Authorization section specifies Client ability to receive authorization request from Device and then make decisions about granting access and send it to Device. This section also specifies Client ability to retrieve and receive notifications about access decisions related to External Authorization.

1.4 Test Cases for Profile Optional Features

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

1.4.1 Get Services with Capabilities

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

1.4.2 Set Synchronization Point (Event Service)

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

1.4.3 Unsubscribe

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

1.4.4 System Date and Time Configuration

System Date and Time Configuration section defines Client ability to configure Device system date and time using GetSystemDateAndTime and SetSystemDateAndTime operations.

1.4.5 Hostname Configuration

Hostname Configuration section defines Client ability to obtain and configure of hostname settings on Device.

1.4.6 DNS Configuration

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

1.4.7 Network Protocols Configuration

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

1.5 Supplementary Features and Test Cases

This section defines supplementary features and test cases which are not the part of profile, but Profile C Features results depends on them.

2 Normative references

- ONVIF Conformance Process Specification:

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

- ONVIF Profile Policy:

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

- ONVIF Network Interface Specifications:

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

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

  www.iso.org/directives

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

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

- WS-BaseNotification:

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

- W3C SOAP 1.2, Part 1, Messaging Framework:

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

- W3C XML Schema Part 1: Structures Second Edition:

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

- W3C XML Schema Part 2: Datatypes Second Edition:

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

- W3C Web Services Addressing 1.0 – Core:

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

- ONVIF Profile C Specification:

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

3 Terms and Definitions

3.1 Conventions

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

3.2 Definitions

This section describes terms and definitions used in this document.

Address	An address refers to a URI.
Profile	See ONVIF Profile Policy.
ONVIF Device	Computer appliance or software program that exposes one or multiple ONVIF Web Services.
ONVIF Client	Computer appliance or software program that uses ONVIF Web Services.
Conversation	A Conversation is all exchanges between two MAC addresses that contains SOAP request and response.
Network	A network is an interconnected group of devices communicating using the Internet protocol.
Network Trace Capture file	Data file created by a network protocol analyzer software (such as Wireshark). Contains network packets data recorded during a live network communications.
SOAP	SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols.
Client Test Tool	ONVIF Client Test Tool that tests ONVIF Client implementation towards the ONVIF Test Specification set.
Valid Device Response	Device has responded to specific request with code HTTP or RTSP 200 OK and SOAP fault message has not appeared.
Profile C	The Profile C Specification.
Door	A physical door, barrier, turnstile, etc which can be controlled remotely and restricts access between two areas. A door is usually equipped with an electronic lock and a sensor.
Door Alarm	An abnormal state of the door where door is forced open or held open beyond the permitted time duration.
Door Mode	Logical state of the door indicating whether the door is locked, unlocked, blocked, locked down or locked open etc.
Lock	An operation after which a door is locked and alarm is unmasked.

Unlock	An operation to allow a door to be freely used for passage without any door alarms being triggered.
Access Point	A logical composition of a physical door and ID point(s) controlling access in one direction.
Disable Access Point	If an Access Point is disabled, it will not be considered in the decision making process and no commands will be issued from that Access Point to the Door configured for that Access Point. When an Access Point is disabled, the associated ID Point may or may not be disabled or shut down. Clients may still be able to command the Door Controller to control associated door even though that door is also referenced by a disabled access point.
ID Point	A device that converts reader signals to protocols recognized by an authorization engine. It can be card reader, REX, biometric reader etc.

3.3 Abbreviations

This section describes abbreviations used in this document.

HTTP Hyper Text Transport Protocol.

HTTPS Hyper Text Transport Protocol over Secure Socket Layer.

IP Internet Protocol.

IPv4 Internet Protocol version 4.

TCP Transport Control Protocol.

UDP User Datagram Protocol.

URI Uniform Resource Identifier.

WSDL Web Services Description Language.

XML eXtensible Markup Language.

PACS Physical Access Control System.

3.4 Namespaces

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

Table 3.1. Defined namespaces in this specification

Prefix	Namespace URI	Description
soapenv	http://www.w3.org/2003/05/soap-envelope	Envelope namespace as defined by SOAP 1.2 [SOAP 1.2, Part 1]
xs	http://www.w3.org/2001/XMLSchema	Instance namespace as defined by XS [XML-Schema, Part1] and [XMLSchema,Part 2]

Prefix	Namespace URI	Description
xsi	http://www.w3.org/2001/XMLSchema-instance	XML schema instance namespace
tns1	http://www.onvif.org/ver10/topics	The namespace for the ONVIF topic namespace
tt	http://www.onvif.org/ver10/schema	ONVIF XML schema descriptions
tds	http://www.onvif.org/ver10/device/wsdl	The namespace for the WSDL device service
tev	http://www.onvif.org/ver10/events/wsdl	The namespace for the WSDL event service
ter	http://www.onvif.org/ver10/error	The namespace for ONVIF defined faults
wsnt	http://docs.oasis-open.org/wsn/b-2	Schema namespace of the [WS-BaseNotification] specification.
wsa	http://www.w3.org/2005/08/addressing	Device addressing namespace as defined by [WS-Addressing].
tac	http://www.onvif.org/ver10/accesscontrol/wsdl	The namespace for the WSDL access control service
tdc	http://www.onvif.org/ver10/doorcontrol/wsdl	The namespace for the WSDL door control service

4 Test Overview

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

An ONVIF Client conformant to Profile C is an ONVIF Client that can request information regarding the Physical Access Control System (PACS) related entities from an ONVIF Device conformant to Profile C and do basic control of Doors and Access Points over an IP network. ONVIF Client can also retrieve and receive standardized PACS related events.

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

4.1 General

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

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

4.1.1 Feature Level Requirement

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

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

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

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

4.1.2 Expected Scenarios Under Test

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

4.1.3 Test Cases

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

Each Test Case contains the following parts:

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

4.2 Test Setup

Collect Network traces files required by the test cases.

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

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

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

4.3 Prerequisites

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

The Device shall be configured with an IPv4 address.

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

5 Test Cases for Profile Mandatory Features

5.1 HTTP Digest Test Cases

5.1.1 Feature Level Requirement:

Validated Feature: HTTP Digest authentication (HTTPDigest)

Check Condition based on Device Features: Digest

Required Number of Devices: 3

Profile A Requirement: Mandatory

Profile C Requirement: Mandatory

Profile D Requirement: Mandatory

Profile G Requirement: Mandatory

Profile S Requirement: Mandatory

Profile T Requirement: Mandatory

Profile M Requirement: Mandatory

5.1.2 Expected Scenarios Under Test:

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

- All HTTP Digest attempts detected are failed.

5.1.3 HTTP DIGEST

Test Label: Security - HTTP Digest Authentication.

Test Case ID: HTTPDIGEST-1

Feature Under Test: HTTP Digest (HTTPDigest_HTTPDigestAuthentication)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

- Client HTTP GET request with HTTP Authentication has a proper hierarchy (refer to [RFC 1945]) AND
  - [S6] Client request contains "realm=*" element with value from Device response AND
  - [S7] Client request contains "nonce=*" element with value from Device response AND
  - [S8] Client request contains "uri=*" element AND
  - [S9] Device response contains "HTTP/* 200 OK".

FAIL -

- The Client failed PASS criteria.

5.2 Capabilities Test Cases

5.2.1 Feature Level Requirement:

Validated Feature: Capabilities (Capabilities)

Check Condition based on Device Features: None

Required Number of Devices: 3

Profile A Requirement: Mandatory

Profile C Requirement: Mandatory

Profile G Requirement: Mandatory

Profile S Requirement: Mandatory

Profile T Requirement: Mandatory

5.2.2 Expected Scenarios Under Test:

1. Client invokes a specific Capabilities command which is under testing.
2. Client is considered as supporting Capabilities if the following conditions are met:
   - Device returns a valid response to GetServices request OR
   - Device returns a valid response to GetCapabilities request.

3. Client is considered as NOT supporting Capabilities if the following is TRUE:
    - No Valid Device Response to GetServices request AND
    - No Valid Device Response to GetCapabilities request.

5.2.3 GET SERVICES

Test Label: Capabilities - Determine the available Services

Test Case ID: CAPABILITIES-1

Feature Under Test: Get Services (Capabilities_GetServicesRequest)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

5.2.4 GET CAPABILITIES

Test Label: Capabilities - Get Device Capabilities

Test Case ID: CAPABILITIES-2

Feature Under Test: Get Capabilities (Capabilities_GetCapabilities)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

5.3 Get Services Test Cases

5.3.1 Feature Level Requirement:

Validated Feature: Get Services (GetServices)

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

Required Number of Devices: 3

Profile A Requirement: Mandatory

Profile D Requirement: Mandatory

Profile C Requirement: Mandatory

Profile G Requirement: Mandatory

Profile T Requirement: Mandatory

Profile M Requirement: Mandatory

5.3.2 Expected Scenarios Under Test:

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

5.3.3 GET SERVICES

Test Label: Capabilities - Determine the available Services

Test Case ID: CAPABILITIES-1

Feature Under Test: Get Services (Capabilities_GetServicesRequest)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

5.4 Event Handling Test Cases

5.4.1 Feature Level Requirement:

Validated Feature: Event Handling (EventHandling)

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

Required Number of Devices: 3

Profile S Requirement: Conditional

Profile G Requirement: Conditional

Profile A Requirement: Mandatory

Profile C Requirement: Mandatory

Profile T Requirement: Mandatory

Profile D Requirement: Mandatory

5.4.2 Expected Scenarios Under Test:

1. Client connects to Device to initiate Event Handling.
2. Client is considered as supporting Event Handling if the following conditions are met:
   - Client is able to handle the Pull Point Event mechanism OR
   - Client is able to handle the Base Notification Event mechanism OR
   - Client is able to handle the Metadata Streaming by supporting EventHandling_MetadataStreamingUsingMedia feature (please see EVENTHANDLING-4 METADATA STREAMING USING MEDIA section) OR Media2_MetadataStreaming_MetadataStreamingUsingMedia2 feature (please see MEDIA2_METADATASTREAMING-1 METADATA STREAMING USING MEDIA2 section).
3. Client is considered as NOT supporting Event Handling if the following is TRUE:
   - All Pull Point attempts detected have failed AND
   - All Base Notification attempts detected have failed AND
   - All Metadata Streaming attempts detected have failed.

5.4.3 PULLPOINT

Test Label: Event Handling - Pull Point

Test Case ID: EVENTHANDLING-1

Feature Under Test: Pull Point (EventHandling_PullPoint)

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

Pre-Requisite:

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

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

1. Client invokes CreatePullPointSubscription message.

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

5.4.4 BASE NOTIFICATION

Test Label: Event Handling - Basic Notification

Test Case ID: EVENTHANDLING-2

Feature Under Test: Base Notification (EventHandling_WSBaseNotification)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

5.4.5 METADATA STREAMING USING MEDIA

Test Label: Event Handling - Metadata Streaming Using Media Streaming

Test Case ID: EVENTHANDLING-4

Feature Under Test: Metadata Streaming (EventHandling_MetadataStreamingUsingMedia)

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

Pre-Requisite:

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

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

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

Test Result:

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

PASS -

- There is Client **RTSP DESCRIBE** request in Test Procedure
- Device response on the **RTSP DESCRIBE** request fulfills the following requirements:
  - [S1] It has RTSP 200 response code AND
  - [S2] SDP packet contains media type "application" (m=application) with sessions attribute "rtpmap" with encoding name "vnd.onvif.metadata" OR "vnd.onvif.metadata.gzip" OR "vnd.onvif.metadata.exi.onvif" OR "vnd.onvif.metadata.exi.ext" (see ONVIF Streaming Spec) AND

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

- [S17] It invoked for the same Device as for the Client **RTSP SETUP** request AND
- [S18] It invoked after the Client **RTSP PLAY** request AND
- [S19] RTSP address that was used to send it is correspond to corresponding media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND

- If there is Device response on the **RTSP TEARDOWN** request then it fulfills the following requirements:
  - [S20] It has RTSP 200 response code.

FAIL -

- The Client failed PASS criteria.

5.5 Keep Alive for Pull Point Event Handling Test Cases

5.5.1 Feature Level Requirement:

Validated Feature: Keep Alive for Pull Point Event Handling (KeepAliveForPullPointEventHandling)

Check Condition based on Device Features: Pull Point Notification is supported by Device.

Required Number of Devices: 3

Profile A Requirement: Mandatory

Profile C Requirement: Mandatory

Profile S Requirement: Conditional

Profile G Requirement: Conditional

Profile T Requirement: Optional

5.5.2 Expected Scenarios Under Test:

1. Client connects to Device to initiate Pull Point Event Handling.
2. Client is considered as supporting Keep Alive for Pull Point Event Handling if the following conditions are met:

- Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
- Client is able to renew pull point subscribtion using **Renew** operation OR **PullMessages** operation mechanism.

3. Client is considered as NOT supporting Keep Alive for Pull Point Event Handling if the following is TRUE:
   - No valid responses for **Renew** request AND for **CreatePullPointSubscription** request in the case if **PullMessages** used for keep alive OR
   - No valid responses for **Renew** request if detected OR
   - No valid responses for **CreatePullPointSubscription** request in the case if **PullMessages** used for keep alive if detected OR
   - **Renew** request was invoked to address which was not specified in **tev:SubscriptionReference\wsa:Address** element of corresponding **CreatePullPointSubscriptionResponse** message.

5.5.3 PULLPOINT

Test Label: Event Handling - Pull Point

Test Case ID: EVENTHANDLING-1

Feature Under Test: Pull Point (EventHandling_PullPoint)

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

Pre-Requisite:

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

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

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

4. Device responds with code HTTP 200 OK and PullMessagesResponse message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

5.5.4 RENEW

Test Label: Advanced Pull Point Event Handling - Renew

Test Case ID: KEEPALIVEFORPULLPOINTEVENTHANDLING-1

Feature Under Test: Renew (KeepAliveForPullPointEventHandling_Renew)

Test Purpose: To verify that the Client is able to use **Renew** operation as keep alive for Pull Point subscribtion.

Pre-Requisite:

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

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

1. Client invokes **CreatePullPointSubscription** message.
2. Device responds with code HTTP 200 OK and **CreatePullPointSubscriptionResponse** message.
3. Client invokes **Renew** message to valid address recieved in **CreatePullPointSubscriptionResponse** message for the created Pull Point subscribtion with valid address recieved in **CreatePullPointSubscriptionResponse** message.
4. Device responds with code HTTP 200 OK and **RenewResponse** message.

Test Result:

PASS -

- Client **Renew** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **Renew** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **wsnt:Renew** AND
- Device response on the **Renew** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **wsnt:RenewResponse** AND
- There is a Device response on the **CreatePullPointSubscription** request in Test Procedure fulfills the following requirements:
  - [S4] It has HTTP 200 response code AND
  - [S5] It received for the same Device as for the Client **Renew** request AND
  - [S6] It received before the Client **Renew** request AND
  - [S7] It contains **tev:SubscriptionReference\wsa:Address** element which is equal to HTTP address that was used to send the **Renew** request.

FAIL -

- The Client failed PASS criteria.

5.5.5 PULL MESSAGES AS KEEP ALIVE

Test Label: Advanced Pull Point Event Handling - Pull Messages as Keep Alive

Test Case ID: KEEPALIVEFORPULLPOINTEVENTHANDLING-2

Feature Under Test: Pull Messages as Keep Alive (KeepAliveForPullPointEventHandling_PullMessagesAsKeepAlive)

Test Purpose: To verify that the Client is able to use **PullMessages** operation as keep alive for Pull Point subscribtion.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **CreatePullPointSubscription** operations whithout **tev:InitialTerminationTime** element present.

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

1. Client invokes **CreatePullPointSubscription** message.
2. Device responds with code HTTP 200 OK and **CreatePullPointSubscriptionResponse** message whithout **tev:InitialTerminationTime** element.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

5.6 Access Point Information Test Cases

5.6.1 Feature Level Requirement:

Validated Feature: Access Point Information (AccessPointInformation)

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

Required Number of Devices: 3

Profile C Requirement: Mandatory

Profile D Requirement: Mandatory

5.6.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve a lists of Access Points.
2. Client is considered as supporting Access Point Information if the following conditions are met:
   - Client is able to list available Access Points using GetAccessPointInfoList operation.
3. Client is considered as NOT supporting Access Point Information if ANY of the following is TRUE:
   - • No valid responses for **GetAccessPointInfoList**.

5.6.3 LISTING OF ACCESS POINTS

Test Label: System Component Information - Listing of Access Points

Test Case ID: ACCESSPOINTINFORMATION-1

Feature Under Test: Listing of Access Points (AccessPointInformation_ListingOfAccessPoints)

Test Purpose: To verify that list of all access points items provided by Device is received by Client using the GetAccessPointInfoList operation.

Pre-Requisite:

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

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

1. Client invokes GetAccessPointInfoList request message to retrieve complete list of all access points configured on the Device.
2. Device responds with code HTTP 200 OK and GetAccessPointInfoListResponse message.

Test Result:

PASS -

- Client **GetAccessPointInfoList** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetAccessPointInfoList** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tcr:GetAccessPointInfoList** AND
- Device response on the **GetAccessPointInfoList** request fulfils the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tcr:GetAccessPointInfoListResponse**.
  - [S4] **tcr:GetAccessPointInfoListResponse** does not contain **tcr:NextStartReference** element.

FAIL -

- The Client failed PASS criteria.

5.7 Door Information Test Cases

5.7.1 Feature Level Requirement:

Validated Feature: Door Information (DoorInformation)

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

Required Number of Devices: 3

Profile C Requirement: Mandatory

Profile D Requirement: Mandatory

5.7.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve a lists of Doors.
2. Client is considered as supporting Door Information if the following conditions are met:
   - Client is able to list available Doors using GetDoorInfoList operation.
3. Client is considered as NOT supporting Door Information if ANY of the following is TRUE:
   - No valid responses for **GetDoorInfoList**.

5.7.3 LISTING OF DOORS

Test Label: System Component Information - Listing of Doors

Test Case ID: DOORINFORMATION-1

Feature Under Test: Listing of Doors (DoorInformation_ListingOfDoors)

Test Purpose: To verify that list of all doors items provided by Device is received by Client using the GetDoorInfoList operation.

Pre-Requisite:

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

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

1. Client invokes GetDoorInfoList request message to retrieve complete list of all doors configured on the Device.
2. Device responds with code HTTP 200 OK and GetDoorInfoListResponse message.

Test Result:

PASS -

- Client **GetDoorInfoList** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetDoorInfoList** request in Test Procedure fulfills the following requirements:

- [S1] Client request contains "<GetDoorInfoList>" tag after the "<Body>" tag AND
- [S2] Device response contains "HTTP/* 200 OK" AND
- [S3] Device response contains "<GetDoorInfoListResponse>" tag AND
- [S4] At least one Device response in the same Conversation does not contain: "<NextStartReference>" tag.

FAIL -

- The Client failed PASS criteria.

5.8 Area Information Test Cases

5.8.1 Feature Level Requirement:

Validated Feature: Area Information (AreaInformation)

Check Condition based on Device Features: Access Control Service is supported by Device. Area Entity is supported by Device.

Required Number of Devices: 3

Profile C Requirement: Mandatory

5.8.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve a lists of Areas.
2. Client is considered as supporting Area Information if the following conditions are met:
   - Client is able to list available Areas using GetAreaInfoList operation.
3. Client is considered as NOT supporting Area Information if ANY of the following is TRUE:
   - No valid responses for **GetAreaInfoList**.

5.8.3 LISTING OF AREAS

Test Label: Area Information - Listing of Areas

Test Case ID: AREAINFORMATION-1

Feature Under Test: Listing of Areas (AreaInformation_ListingOfAreas)

Test Purpose: To verify that list of all areas items provided by Device is received by Client using the GetAreaInfoList operation.

Pre-Requisite:

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

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

1. Client invokes GetAreaInfoList request message to retrieve complete list of all areas configured on the Device.
2. Device responds with code HTTP 200 OK and GetAreaInfoListResponse message.

Test Result:

PASS -

- Client **GetAreaInfoList** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetAreaInfoList** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<GetAreaInfoList>" tag after the "<Body>" tag AND
  - [S2] Device response contains "HTTP/* 200 OK" AND
  - [S3] Device response contains "<GetAreaInfoListResponse>" tag AND
  - [S4] At least one Device response in the same Conversation does not contain: "<NextStartReference>" tag.

FAIL -

- The Client failed PASS criteria.

5.9 System Component State Test Cases

5.9.1 Feature Level Requirement:

Validated Feature: System Component State (SystemComponentState)

Check Condition based on Device Features: Access Control Service and Door Control Service are supported by Device.

Required Number of Devices: 3

Profile C Requirement: Mandatory

5.9.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation to get notifications about the state of access points.
2. Client uses Pull Point event mechanism to retrieve notification events from Device.
3. Client is considered as supporting System Component State if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client supports AccessPointInformation_ListingOfAccessPoints feature (please see ACCESSPOINTINFORMATION-1 LISTING OF ACCESS POINTS section) AND
   - Client supports DoorInformation_ListingOfDoors feature (please see DOORINFORMATION-1 LISTING OF DOORS section) AND
   - Client is able to receive tns1:AccessPoint/State/Enabled notification about a state of access point if Device supports AccessPointStateEnabledEvent AND
   - Client is able to retrieve at least one of the following notifications about a state of door:
     - tns1:Door/State/DoorMode notification if Device supports DoorModeEvent
     - tns1:Door/State/DoorPhysicalState notification if Device supports DoorPhysicalStateEvent
     - tns1:Door/State/LockPhysicalState notification if Device supports LockPhysicalStateEvent
     - tns1:Door/State/DoubleLockPhysicalState notification if Device supports DoubleLockPhysicalStateEvent
     - tns1:Door/State/DoorAlarm notification if Device supports DoorAlarmEvent
     - tns1:Door/State/DoorTamper notification if Device supports DoorTamperEvent
     - tns1:Door/State/DoorFault notification if Device supports DoorFaultEvent

4. Client is considered as NOT supporting System Component State if ANY of the following is TRUE:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client does not support AccessPointInformation_ListingOfAccessPoints feature (please see ACCESSPOINTINFORMATION-1 LISTING OF ACCESS POINTS section) AND
   - Client does not support DoorInformation_ListingOfDoors feature (please see DOORINFORMATION-1 LISTING OF DOORS section) AND
   - Client is not able to receive tns1:AccessPoint/State/Enabled notification about a state of access point if Device supports AccessPointStateEnabledEvent AND
   - Client is not able to retrieve the following notifications about a state of door:
     - tns1:Door/State/DoorMode notification if Device supports DoorModeEvent
     - tns1:Door/State/DoorPhysicalState notification if Device supports DoorPhysicalStateEvent
     - tns1:Door/State/LockPhysicalState notification if Device supports LockPhysicalStateEvent
     - tns1:Door/State/DoubleLockPhysicalState notification if Device supports DoubleLockPhysicalStateEvent
     - tns1:Door/State/DoorAlarm notification if Device supports DoorAlarmEvent
     - tns1:Door/State/DoorTamper notification if Device supports DoorTamperEvent
     - tns1:Door/State/DoorFault notification if Device supports DoorFaultEvent

5.10 Door Control Test Cases

5.10.1 Feature Level Requirement:

Validated Feature: Door Control (DoorControl)

Check Condition based on Device Features: Door Control Service and Access Door and Lock Door and Unlock Door are supported by Device.

Required Number of Devices: 3

Profile C Requirement: Mandatory

5.10.2 Expected Scenarios Under Test:

1. Client invokes a specific, valid mandatory Door Control command in order to change the state of door.
2. Client is considered as supporting Door Control if the following conditions are met:
   - Device returns a valid response to AccessDoor request AND
   - Device returns a valid response to LockDoor request AND
   - Device returns a valid response to UnlockDoor request
   - When Device and Client support any of the following conditional features:
     - Device returns a valid response to DoubleLockDoor request OR
     - Device returns a valid response to BlockDoor request
   - When Device and Client support LockDown conditional features:
     - Device returns a valid response to LockDownDoor request AND
     - Device returns a valid response to LockDownReleaseDoor request
   - When Device and Client support LockOpen conditional features:
     - Device returns a valid response to LockOpenDoor request AND
     - Device returns a valid response to LockOpenReleaseDoor request.
3. Client is considered as NOT supporting Door Control if ANY of the following is TRUE:
   - No valid Device response to AccessDoor request OR
   - No valid Device response to LockDoor request OR
   - No valid Device response to UnlockDoor request
   - When Device and Client support any of the following conditional features:
     - No valid Device response to DoubleLockDoor request AND
     - No valid Device response to BlockDoor request
   - When Device and Client support LockDown conditional features:
     - No valid Device response to LockDownDoor request OR

- No valid Device response to LockDownReleaseDoor request
- When Device and Client support LockOpen conditional features:
  - No valid Device response to LockOpenDoor request OR
  - No valid Device response to LockOpenReleaseDoor request.

5.10.3 ACCESS DOOR

Test Label: Door Control - AccessDoor

Test Case ID: DOORCONTROL-1

Feature Under Test: Access Door (DoorControl_AccessDoor)

Test Purpose: To verify that Client is able to change the state of door using AccessDoor operation.

Pre-Requisite:

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

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

1. Client invokes AccessDoor request message to change the state of door.
2. Device responds with code HTTP 200 OK and AccessDoorResponse message.

Test Result:

PASS -

- Client **AccessDoor** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **AccessDoor** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<AccessDoor>" tag after the "<Body>" tag AND
  - [S2] "<AccessDoor>" includes tag: "<Token>" with non-empty string value of specific token AND
  - [S3] Device response contains "HTTP/* 200 OK" AND
  - [S4] Device response contains "<AccessDoorResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.10.4 LOCK DOOR

Test Label: Door Control - LockDoor

Test Case ID: DOORCONTROL-2

Feature Under Test: Lock Door (DoorControl_LockDoor)

Test Purpose: To verify that Client is able to change the state of door using LockDoor operation.

Pre-Requisite:

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

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

1. Client invokes LockDoor request message to change the state of door.
2. Device responds with code HTTP 200 OK and LockDoorResponse message.

Test Result:

PASS -

- Client **LockDoor** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **LockDoor** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<LockDoor>" tag after the "<Body>" tag AND
  - [S2] "<LockDoor>" includes tag: "<Token>" with non-empty string value of specific token AND
  - [S3] Device response contains "HTTP/* 200 OK" AND
  - [S4] Device response contains "<LockDoorResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.10.5 UNLOCK DOOR

Test Label: Door Control - UnlockDoor

Test Case ID: DOORCONTROL-3

Feature Under Test: Unlock Door (DoorControl_UnlockDoor)

Test Purpose: To verify that Client is able to change the state of door using UnlockDoor operation.

Pre-Requisite:

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

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

1. Client invokes UnlockDoor request message to change the state of door.
2. Device responds with code HTTP 200 OK and UnlockDoorResponse message.

Test Result:

PASS -

- Client **UnlockDoor** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **UnlockDoor** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<UnlockDoor>" tag after the "<Body>" tag AND
  - [S2] "<UnlockDoor>" includes tag: "<Token>" with non-empty string value of specific token AND
  - [S3] Device response contains "HTTP/* 200 OK" AND
  - [S4] Device response contains "<UnlockDoorResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.10.6 DOUBLE LOCK DOOR

Test Label: Door Control - DoubleLockDoor

Test Case ID: DOORCONTROL-4

Feature Under Test: Double Lock Door (DoorControl_DoubleLockDoor)

Test Purpose: To verify that Client is able to change the state of door using DoubleLockDoor operation.

Pre-Requisite:

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

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

1. Client invokes DoubleLockDoor request message to change the state of door.
2. Device responds with code HTTP 200 OK and DoubleLockDoorResponse message.

Test Result:

PASS -

- Client **DoubleLockDoor** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **DoubleLockDoor** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<DoubleLockDoor>" tag after the "<Body>" tag AND
  - [S2] "<DoubleLockDoor>" includes tag: "<Token>" with non-empty string value of specific token AND
  - [S3] Device response contains "HTTP/* 200 OK" AND
  - [S4] Device response contains "<DoubleLockDoorResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.10.7 BLOCK DOOR

Test Label: Door Control - BlockDoor

Test Case ID: DOORCONTROL-5

Feature Under Test: Block Door (DoorControl_BlockDoor)

Test Purpose: To verify that Client is able to change the state of door using BlockDoor operation.

Pre-Requisite:

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

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

1. Client invokes BlockDoor request message to change the state of door.
2. Device responds with code HTTP 200 OK and BlockDoorResponse message.

Test Result:

PASS -

- Client **BlockDoor** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **BlockDoor** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<BlockDoor>" tag after the "<Body>" tag AND
  - [S2] "<BlockDoor>" includes tag: "<Token>" with non-empty string value of specific token AND
  - [S3] Device response contains "HTTP/* 200 OK" AND
  - [S4] Device response contains "<BlockDoorResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.10.8 LOCK DOWN DOOR

Test Label: Door Control - LockDownDoor

Test Case ID: DOORCONTROL-6

Feature Under Test: Lock Down Door (DoorControl_LockDownDoor)

Test Purpose: To verify that Client is able to change the state of Door using LockDownDoor operation and then releasing this state using LockDownReleaseDoor operation.

Pre-Requisite:

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

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

1. Client invokes LockDownDoor request message to change the state of door.
2. Device responds with code HTTP 200 OK and LockDownDoorResponse message.
3. Client invokes LockDownReleaseDoor request message to release the LockedDown state.

4. Device responds with code HTTP 200 OK and LockDownReleaseDoorResponse message.

Test Result:

PASS -

- Client **LockDownDoor** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **LockDownDoor** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<LockDownDoor>" tag after the "<Body>" tag AND
  - [S2] "<LockDownDoor>" includes tag: "<Token>" with non-empty string value of specific token AND
  - [S3] Device response contains "HTTP/* 200 OK" AND
  - [S4] Device response contains "<LockDownDoorResponse>" tag AND
- Client **LockDownReleaseDoor** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **LockDownReleaseDoor** request in Test Procedure fulfills the following requirements:
  - [S5] Client request contains "<LockDownReleaseDoor>" tag after the "<Body>" tag AND
  - [S6] "<LockDownReleaseDoor>" includes tag: "<Token>" with token value from LockDownDoor operation AND
  - [S7] Device response contains "HTTP/* 200 OK" AND
  - [S8] Device response contains "<LockDownReleaseDoorResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.10.9 LOCK OPEN DOOR

Test Label: Door Control - LockOpenDoor

Test Case ID: DOORCONTROL-7

Feature Under Test: Lock Open Door (DoorControl_LockOpenDoor)

Test Purpose: To verify that Client is able to change the state of Door using LockOpenDoor operation and then releasing this state using LockOpenReleaseDoor operation.

Pre-Requisite:

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

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

1. Client invokes LockOpenDoor request message to change the state of door.
2. Device responds with code HTTP 200 OK and LockOpenDoorResponse message.
3. Client invokes LockOpenReleaseDoor request message to release the LockOpenDoor state.
4. Device responds with code HTTP 200 OK and LockOpenReleaseDoorResponse message.

Test Result:

PASS -

- Client **LockOpenDoor** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **LockOpenDoor** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<LockOpenDoor>" tag after the "<Body>" tag AND
  - [S2] "<LockOpenDoor>" includes tag: "<Token>" with non-empty string value of specific token AND
  - [S3] Device response contains "HTTP/* 200 OK" AND
  - [S4] Device response contains "<LockOpenDoorResponse>" tag. AND
- Client **LockOpenReleaseDoor** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **LockOpenReleaseDoor** request in Test Procedure fulfills the following requirements:
  - [S5] Client request contains "<LockOpenReleaseDoor>" tag after the "<Body>" tag AND
  - [S6] "<LockOpenReleaseDoor>" includes tag: "<Token>" with token value from LockOpenDoor operation AND
  - [S7] Device response contains "HTTP/* 200 OK" AND
  - [S8] Device response contains "<LockOpenReleaseDoorResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.11 Access Control Decisions Test Cases

5.11.1 Feature Level Requirement:

Validated Feature: Access Control Decisions (AccessControlDecisions)

Check Condition based on Device Features: Access Control Service and tns1:AccessControl/AccessGranted/Credential and tns1:AccessControl/AccessDenied/Credential and tns1:AccessControl/AccessGranted/Anonymous and tns1:AccessControl/AccessDenied/AnonymousEvent tns1:AccessControl/AccessDenied/Credential/CredentialNotFoundCard and tns1:AccessControl/AccessTaken/Anonymous and tns1:AccessControl/AccessTaken/Credential and tns1:AccessControl/AccessNotTaken/Anonymous and tns1:AccessControl/AccessNotTaken/CredentialEvent are supported by Device.

Required Number of Devices: 3

Profile C Requirement: Mandatory

5.11.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using CreatePullPointSubscription operation to get Access Control Decisions notifications.
2. Client is considered as supporting Access Control Decisions if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client supports AccessPointInformation_ListingOfAccessPoints feature (please see ACCESSPOINTINFORMATION-1 LISTING OF ACCESS POINTS section) AND
   - Client is able to retrieve **tns1:AccessControl/AccessGranted/Credential** notification AND
   - Client is able to retrieve **tns1:AccessControl/Denied/Credential** notification AND
   - Client is able to retrieve **tns1:AccessControl/AccessGranted/Anonymous** notification AND
   - Client is able to retrieve **tns1:AccessControl/Denied/Anonymous** notification AND

- Client is able to retrieve **tns1:AccessControl/Denied/CredentialNotFound/Card** notification AND
- Client is able to retrieve **tns1:AccessControl/AccessTaken/Credential** notification AND
- Client is able to retrieve **tns1:AccessControl/AccessTaken/Anonymous** notification AND
- Client is able to retrieve **tns1:AccessControl/AccessNotTaken/Credential** notification AND
- Client is able to retrieve **tns1:AccessControl/AccessNotTaken/Anonymous** notification.

3. Client is considered as NOT supporting Access Control Decisions if ANY of the following is TRUE:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) OR
   - Client does not support SystemComponentInformation_AccessPointInfoList feature (please see ACCESSPOINTINFORMATION-1 LISTING OF ACCESS POINTS section) OR
   - Client unable to retrieve **tns1:AccessControl/AccessGranted/Credential** notification OR
   - Client unable to retrieve **tns1:AccessControl/Denied/Credential** notification OR
   - Client unable to retrieve **tns1:AccessControl/AccessGranted/Anonymous** notification OR
   - Client unable to retrieve **tns1:AccessControl/Denied/Anonymous** notification OR
   - Client unable to retrieve **tns1:AccessControl/Denied/CredentialNotFound/Card** notification OR
   - Client unable to retrieve **tns1:AccessControl/AccessTaken/Credential** notification OR
   - Client unable to retrieve **tns1:AccessControl/AccessTaken/Anonymous** notification OR
   - Client unable to retrieve **tns1:AccessControl/AccessNotTaken/Credential** notification OR

- Client unable to retrieve **tns1:AccessControl/AccessNotTaken/Anonymous** notification.

5.11.3 PULLPOINT

Test Label: Event Handling - Pull Point

Test Case ID: EVENTHANDLING-1

Feature Under Test: Pull Point (EventHandling_PullPoint)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

- Client **PullMessages** request in Test Procedure fulfills the following requirements:
  - [S4] Client request contains "<PullMessages>" tag after the "<Body>" tag AND
  - [S7] Device response contains "HTTP/* 200 OK" AND
  - [S8] Device response contains "<PullMessagesResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.12 Access Point Information - Configuration Change Notifications Test Cases

5.12.1 Feature Level Requirement:

Validated Feature: Access Point Information - Configuration Change Notifications (AccessPointConfigurationChangeNotifications)

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

Required Number of Devices: 3

Profile C Requirement: Mandatory

Profile D Requirement: Mandatory

5.12.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation to get Configuration Change notifications.
2. Client uses Pull Point event mechanism to retrieve notification events from Device.
3. Client is considered as supporting Configuration change notification if the following conditions are met:
   - Client supports EventHandling_PullPoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client supports AccessPointInformation_ListingOfAccessPoints feature (please see ACCESSPOINTINFORMATION-1 LISTING OF ACCESS POINTS section) AND
   - Client is able to retrieve **tns1:Configuration/AccessPoint/Changed** notification AND

- Client is able to retrieve **tns1:Configuration/AccessPoint/Removed** notification.

4. Client is considered as NOT supporting Configuration change notification if ANY of the following is TRUE:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) OR
   - Client does not support AccessPointInformation_ListingOfAccessPoints feature (please see ACCESSPOINTINFORMATION-1 LISTING OF ACCESS POINTS section) OR
   - Client unable to retrieve **tns1:Configuration/AccessPoint/Changed** notification OR
   - Client unable to retrieve **tns1:Configuration/AccessPoint/Removed** notification.

5.13 Door Information - Configuration Change Notifications Test Cases

5.13.1 Feature Level Requirement:

Validated Feature: Door Information - Configuration Change Notifications (DoorConfigurationChangeNotifications)

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

Required Number of Devices: 3

Profile C Requirement: Mandatory

Profile D Requirement: Mandatory

5.13.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation to get Configuration Change notifications.
2. Client uses Pull Point event mechanism to retrieve notification events from Device.
3. Client is considered as supporting Configuration change notification if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND

- Client supports DoorInformation_ListingOfDoors feature (please see DOORINFORMATION-1 LISTING OF DOORS section) AND
- Client is able to retrieve **tns1:Configuration/Door/Changed** notification AND
- Client is able to retrieve **tns1:Configuration/Door/Removed** notification.

4. Client is considered as NOT supporting Configuration change notification if ANY of the following is TRUE:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) OR
   - Client does not support DoorInformation_ListingOfDoors feature (please see DOORINFORMATION-1 LISTING OF DOORS section) OR
   - Client unable to retrieve **tns1:Configuration/Door/Changed** notification OR
   - Client unable to retrieve **tns1:Configuration/Door/Removed** notification.

5.14 Area Information - Configuration Change Notifications Test Cases

5.14.1 Feature Level Requirement:

Validated Feature: Area Information - Configuration Change Notifications (AreaConfigurationChangeNotifications)

Check Condition based on Device Features: Access Control Service is supported by Device. Area Entity is supported by Device.

Required Number of Devices: 3

Profile C Requirement: Mandatory

5.14.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation to get Configuration Change notifications.
2. Client uses Pull Point event mechanism to retrieve notification events from Device.
3. Client is considered as supporting Configuration change notification if the following conditions are met:

- Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
- Client supports AreaInformation_AreaInfoList feature (please see AREAINFORMATION-1 LISTING OF AREAS section) AND
- Client is able to retrieve **tns1:Configuration/Area/Changed** notification AND
- Client is able to retrieve **tns1:Configuration/Area/Removed** notification.

4. Client is considered as NOT supporting Configuration change notification if ANY of the following is TRUE:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) OR
   - Client does not support AreaInformation_AreaInfoList feature (please see AREAINFORMATION-1 LISTING OF AREAS section) OR
   - Client unable to retrieve **tns1:Configuration/Area/Changed** notification OR
   - Client unable to retrieve **tns1:Configuration/Area/Removed** notification.

5.14.3 PULLPOINT

Test Label: Event Handling - Pull Point

Test Case ID: EVENTHANDLING-1

Feature Under Test: Pull Point (EventHandling_PullPoint)

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

Pre-Requisite:

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

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

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

4. Device responds with code HTTP 200 OK and PullMessagesResponse message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

5.15 Duress Notifications Test Cases

5.15.1 Feature Level Requirement:

Validated Feature: Duress Notifications (DuressNotifications)

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

Required Number of Devices: 3

Profile C Requirement: Mandatory

5.15.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation.
2. Client uses Pull Point event mechanism to retrieve notification events from Device.
3. Client is considered as supporting Duress notification if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client supports AccessPointInformation_ListingOfAccessPoints feature (please see ACCESSPOINTINFORMATION-1 LISTING OF ACCESS POINTS section) AND
   - Client is able to retrieve **tns1:AccessControl/Duress** notification.
4. Client is considered as NOT supporting Duress notification if ANY of the following is TRUE:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) OR
   - Client does not support AccessPointInformation_ListingOfAccessPoints feature (please see ACCESSPOINTINFORMATION-1 LISTING OF ACCESS POINTS section) OR
   - Client unable to retrieve **tns1:AccessControl/Duress** notification.

6 Test Cases for Profile Conditional Features

6.1 Discovery Test Cases

6.1.1 Feature Level Requirement:

Validated Feature: Discovery (Discovery)

Check Condition based on Device Features: Discovery

Required Number of Devices: 3

Profile S Requirement: Conditional

Profile C Requirement: Conditional

Profile G Requirement: Conditional

Profile A Requirement: Mandatory

Profile T Requirement: Mandatory

Profile D Requirement: Mandatory

Profile M Requirement: Mandatory

6.1.2 Expected Scenarios Under Test:

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

6.1.3 WS-DISCOVERY

Test Label: Discovery - WS-Discovery

Test Case ID: DISCOVERY-1

Feature Under Test: WS-Discovery (Discovery_WSDiscovery)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.2 Device Discovery Type Filter Test Cases

6.2.1 Feature Level Requirement:

Validated Feature: Device Discovery Type Filter (DeviceDiscoveryTypeFilter)

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

Required Number of Devices: 3

Profile S Requirement: None

Profile A Requirement: Mandatory

Profile C Requirement: Conditional

Profile D Requirement: Mandatory

Profile G Requirement: Conditional

Profile T Requirement: Mandatory

Profile M Requirement: Mandatory

6.2.2 Expected Scenarios Under Test:

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

6.2.3 DEVICE DISCOVERY TYPE FILTER

Test Label: Discovery - Device Discovery Type Filter

Test Case ID: DEVICEDISCOVERYTYPEFILTER-1

Feature Under Test: Device Discovery Type Filter (DeviceDiscoveryTypeFilter_DeviceDiscoveryFilter)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

PASS WITH WARNING -

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

FAIL -

- The Client failed PASS criteria.

6.3 Network Configuration Test Cases

6.3.1 Feature Level Requirement:

Validated Feature: Network Configuration (NetworkConfiguration)

Check Condition based on Device Features: Network Configuration

Required Number of Devices: 3

Profile A Requirement: Conditional

Profile C Requirement: Conditional

Profile D Requirement: Mandatory

Profile G Requirement: Conditional

Profile S Requirement: Conditional

Profile T Requirement: Mandatory

Profile M Requirement: Mandatory

6.3.2 Expected Scenarios Under Test:

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

- Client is able to list default gateway of Device using the GetNetworkDefaultGateway operation AND
- Client is able set default gateway of Device using the SetNetworkDefaultGateway operation.

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

6.3.3 GET NETWORK INTERFACES

Test Label: Network Configuration - Get Network Interfaces

Test Case ID: NETWORKCONFIGURATION-1

Feature Under Test: Get Network Interfaces (NetworkConfiguration_GetNetworkInterfaces)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

- Client **GetNetworkInterfaces** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<GetNetworkInterfaces>" tag after the "<Body>" tag AND
  - [S2] Device response contains "HTTP/* 200 OK" AND
  - [S3] Device response contains "<GetNetworkInterfacesResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.3.4 SET NETWORK INTERFACES

Test Label: Network Configuration - Set Network Interfaces

Test Case ID: NETWORKCONFIGURATION-2

Feature Under Test: Set Network Interfaces (NetworkConfiguration_SetNetworkInterfaces)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

- [S4] Device response contains "HTTP/* 200 OK" AND
- [S5] Device response contains "<SetNetworkInterfacesResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.3.5 GET NETWORK DEFAULT GATEWAY

Test Label: Network Configuration - Get Network Default Gateway

Test Case ID: NETWORKCONFIGURATION-3

Feature Under Test: Get Network Default Gateway (NetworkConfiguration_GetNetworkDefaultGateway)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

- [S3] Device response contains "<GetNetworkDefaultGatewayResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.3.6 SET NETWORK DEFAULT GATEWAY

Test Label: Network Configuration - Set Network Default Gateway

Test Case ID: NETWORKCONFIGURATION-4

Feature Under Test: Set Network Default Gateway (NetworkConfiguration_SetNetworkDefaultGateway)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

- [S3] Device response contains "HTTP/* 200 OK" AND
- [S4] Device response contains "<SetNetworkDefaultGatewayResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.4 System Test Cases

6.4.1 Feature Level Requirement:

Validated Feature: System (System)

Check Condition based on Device Features: None

Required Number of Devices: 3

Profile A Requirement: Conditional

Profile C Requirement: Conditional

Profile G Requirement: Conditional

Profile S Requirement: Conditional

Profile T Requirement: Conditional

Profile D Requirement: Conditional

Profile M Requirement: Conditional

6.4.2 Expected Scenarios Under Test:

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

6.4.3 GET DEVICE INFORMATION

Test Label: System - Get Device Information

Test Case ID: SYSTEM-1

Feature Under Test: Get Device Information (System_GetDeviceInformation)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.5 User Handling Test Cases

6.5.1 Feature Level Requirement:

Validated Feature: User Handling (UserHandling)

Check Condition based on Device Features: User Configuration

Required Number of Devices: 3

Profile A Requirement: Mandatory

Profile S Requirement: Conditional

Profile C Requirement: Conditional

Profile G Requirement: Conditional

Profile T Requirement: Conditional

Profile D Requirement: Conditional

6.5.2 Expected Scenarios Under Test:

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

6.5.3 CREATE USERS

Test Label: User Handling - CreateUsers

Test Case ID: USERHANDLING-1

Feature Under Test: Create Users (UserHandling_CreateUsers)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.5.4 GET USERS

Test Label: User Handling - GetUsers

Test Case ID: USERHANDLING-2

Feature Under Test: Get Users (UserHandling_GetUsers)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.5.5 SET USER

Test Label: User Handling - SetUser

Test Case ID: USERHANDLING-3

Feature Under Test: Set User (UserHandling_SetUser)

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

Pre-Requisite:

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

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

1. Client invokes SetUser request message to update the authentication settings on Device.
2. Device responds with code HTTP 200 OK and SetUserResponse message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.5.6 DELETE USERS

Test Label: User Handling - DeleteUsers

Test Case ID: USERHANDLING-4

Feature Under Test: Delete Users (UserHandling_DeleteUsers)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.6 IP Address Filtering Test Cases

6.6.1 Feature Level Requirement:

Validated Feature: IP Address Filtering (IPAddressFiltering)

Check Condition based on Device Features: IP Filter is supported by Device.

Required Number of Devices: 1

Profile S Requirement: Conditional

Profile C Requirement: Conditional

Profile A Requirement: Conditional

6.6.2 Expected Scenarios Under Test:

1. Client connects to Device to manage IP address filters.

2. Client is considered as supporting IP Address Filtering if the following conditions are met:
   - Client is able to get the IP address filter settings from Device using the GetIPAddressFilter operation AND
   - Client is able to set the IP address filter settings on Device using the SetIPAddressFilter operation AND
   - Client is able to add the IP address filter settings to Device using the AddIPAddressFilter operation AND
   - Client is able to delete the IP address filter settings from Device using the RemoveIPAddressFilter operation.
   - **NOTE:** Requests SetIPAddressFilter, AddIPAddressFilter and RemoveIPAddressFilter are permitted to use the IPv4 OR IPv6 protocol settings.
3. Client is considered as NOT supporting IP Address Filtering if ANY of the following is TRUE:
   - No Valid Device Response to GetIPAddressFilter request OR
   - No Valid Device Response to SetIPAddressFilter request OR
   - No Valid Device Response to AddIPAddressFilter request OR
   - No Valid Device Response to RemoveIPAddressFilter request.
   - NOTE: Requests SetIPAddressFilter, AddIPAddressFilter and RemoveIPAddressFilter should be deemed as failed if both IPv4 AND IPv6 protocol settings have No Valid Device Responses.

6.6.3 GET IP ADDRESS FILTER

Test Label: IP Address Filtering - GetIPAddressFilter

Test Case ID: IPADDRESSFILTERING-1

Feature Under Test: Get Ip Address Filter (IPAddressFiltering_GetIpAddressFilter)

Test Purpose: To verify that Client is able to get the IP address filter settings from Device using the GetIPAddressFilter operation.

Pre-Requisite:

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

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

1. Client invokes GetIPAddressFilter request message to get the IP address filter settings from Device.
2. Device responds with code HTTP 200 OK and GetIPAddressFilterResponse message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.6.4 SET IPv4 ADDRESS FILTER

Test Label: IP Address Filtering - SetIPv4AddressFilter

Test Case ID: IPADDRESSFILTERING-2

Feature Under Test: Set IPv4 Address Filter (IPAddressFiltering_SetIpV4AddressFilter)

Test Purpose: To verify that Client is able to set the IP address filter settings on Device using the SetIPAddressFilter operation.

Pre-Requisite:

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

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

1. Client invokes SetIPAddressFilter request message to set the IP address filter settings on Device.
2. Device responds with code HTTP 200 OK and SetIPAddressFilterResponse message.

Test Result:

NOTE: If Client SetIPAddressFilter request message does not contain "<IPv4Address>" tag then Test shall be deemed as "NOT DETECTED".

PASS -

- Client **SetIPAddressFilter** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetIPAddressFilter** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<SetIPAddressFilter>" tag after the "<Body>" tag AND
  - [S3] "<SetIPAddressFilter>" includes tag: "<IPv4Address>" AND
  - [S4] "<IPv4Address>" includes tag: "<Address>" with specific IPv4 address value AND
  - [S5] "<IPv4Address>" includes tag: "<PrefixLength>" with value range from "0" to "32" AND
  - [S6] Device response contains "HTTP/* 200 OK" AND
  - [S7] Device response contains "<SetIPAddressFilterResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.6.5 SET IPv6 ADDRESS FILTER

Test Label: IP Address Filtering - SetIPv6AddressFilter

Test Case ID: IPADDRESSFILTERING-3

Feature Under Test: Set IPv6 Address Filter (IPAddressFiltering_SetIpV6AddressFilter)

Test Purpose: To verify that Client is able to set the IP address filter settings on Device using the SetIPAddressFilter operation.

Pre-Requisite:

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

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

1. Client invokes SetIPAddressFilter request message to set the IP address filter settings on Device.

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

Test Result:

NOTE: If Client SetIPAddressFilter request message does not contain "<IPv6Address>" tag then Test shall be deemed as "NOT DETECTED".

PASS -

- Client **SetIPAddressFilter** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetIPAddressFilter** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<SetIPAddressFilter>" tag after the "<Body>" tag AND
  - [S3] "<SetIPAddressFilter>" includes tag: "<IPv6Address>" AND
  - [S4] "<IPv6Address>" includes tag: "<Address>" with specific IPv6 address value AND
  - [S5] "<IPv6Address>" includes tag: "<PrefixLength>" with value range from "0" to "128" AND
  - [S6] Device response contains "HTTP/* 200 OK" AND
  - [S7] Device response contains "<SetIPAddressFilterResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.6.6 ADD IPv4 ADDRESS FILTER

Test Label: IP Address Filtering - AddIPv4AddressFilter

Test Case ID: IPADDRESSFILTERING-4

Feature Under Test: Add IPv4 Address Filter (IPAddressFiltering_AddIpV4AddressFilter)

Test Purpose: To verify that Client is able to add the IP address filter to Device using the AddIPAddressFilter operation.

Pre-Requisite:

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

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

1. Client invokes AddIPAddressFilter request message to add the IP address filter on Device.
2. Device responds with code HTTP 200 OK and AddIPAddressFilterResponse message.

Test Result:

NOTE: If Client AddIPAddressFilter request message does not contain "<IPv4Address>" tag then Test shall be deemed as "NOT DETECTED".

PASS -

- Client **AddIPAddressFilter** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **AddIPAddressFilter** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<AddIPAddressFilter>" tag after the "<Body>" tag AND
  - [S3] "<AddIPAddressFilter>" includes tag: "<IPv4Address>" AND
  - [S4] "<IPv4Address>" includes tag: "<Address>" with specific IPv4 address value AND
  - [S5] "<IPv4Address>" includes tag: "<PrefixLength>" with value range from "0" to "32" AND
  - [S6] Device response contains "HTTP/* 200 OK" AND
  - [S7] Device response contains "<AddIPAddressFilterResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.6.7 ADD IPv6 ADDRESS FILTER

Test Label: IP Address Filtering - AddIPv6AddressFilter

Test Case ID: IPADDRESSFILTERING-5

Feature Under Test: Add IPv6 Address Filter (IPAddressFiltering_AddIpV6AddressFilter)

Test Purpose: To verify that Client is able to add the IP address filter to Device using the AddIPAddressFilter operation.

Pre-Requisite:

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

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

1. Client invokes AddIPAddressFilter request message to add the IP address filter on Device.
2. Device responds with code HTTP 200 OK and AddIPAddressFilterResponse message.

Test Result:

NOTE: If Client AddIPAddressFilter request message does not contain "<IPv6Address>" tag then Test shall be deemed as "NOT DETECTED".

PASS -

- Client **AddIPAddressFilter** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **AddIPAddressFilter** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<AddIPAddressFilter>" tag after the "<Body>" tag AND
  - [S3] "<AddIPAddressFilter>" includes tag: "<IPv6Address>" AND
  - [S4] "<IPv6Address>" includes tag: "<Address>" with specific IPv6 address value AND
  - [S5] "<IPv6Address>" includes tag: "<PrefixLength>" with value range from "0" to "128" AND
  - [S6] Device response contains "HTTP/* 200 OK" AND
  - [S7] Device response contains "<AddIPAddressFilterResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.6.8 REMOVE IPv4 ADDRESS FILTER

Test Label: IP Address Filtering - RemoveIPv4AddressFilter

Test Case ID: IPADDRESSFILTERING-6

Feature Under Test: Remove IPv4 Address Filter (IPAddressFiltering_RemoveIpV4AddressFilter)

Test Purpose: To verify that Client is able to delete the IP address filter from Device using the RemoveIPAddressFilter operation.

Pre-Requisite:

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

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

1. Client invokes RemoveIPAddressFilter request message to delete the IP address filter from Device.
2. Device responds with code HTTP 200 OK and RemoveIPAddressFilterResponse message.

Test Result:

NOTE: If Client RemoveIPAddressFilter request message does not contain "<IPv4Address>" tag then Test shall be deemed as "NOT DETECTED".

PASS -

- Client **RemoveIPAddressFilter** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **RemoveIPAddressFilter** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<RemoveIPAddressFilter>" tag after the "<Body>" tag AND
  - [S3] "<RemoveIPAddressFilter>" includes tag: "<IPv4Address>" AND
  - [S4] "<IPv4Address>" includes tag: "<Address>" with specific IPv4 address value AND
  - [S5] "<IPv4Address>" includes tag: "<PrefixLength>" with value range from "0" to "32" AND
  - [S6] Device response contains "HTTP/* 200 OK" AND
  - [S7] Device response contains "<RemoveIPAddressFilterResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.6.9 REMOVE IPv6 ADDRESS FILTER

Test Label: IP Address Filtering - RemoveIPv6AddressFilter

Test Case ID: IPADDRESSFILTERING-7

Feature Under Test: Remove IPv6 Address Filter (IPAddressFiltering_RemoveIpV6AddressFilter)

Test Purpose: To verify that Client is able to delete the IP address filter from Device using the RemoveIPAddressFilter operation.

Pre-Requisite:

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

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

1. Client invokes RemoveIPAddressFilter request message to delete the IP address filter from Device.
2. Device responds with code HTTP 200 OK and RemoveIPAddressFilterResponse message.

Test Result:

NOTE: If Client RemoveIPAddressFilter request message does not contain "<IPv6Address>" tag then Test shall be deemed as "NOT DETECTED".

PASS -

- Client **RemoveIPAddressFilter** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **RemoveIPAddressFilter** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<RemoveIPAddressFilter>" tag after the "<Body>" tag AND
  - [S3] "<RemoveIPAddressFilter>" includes tag: "<IPv6Address>" AND
  - [S4] "<IPv6Address>" includes tag: "<Address>" with specific IPv6 address value AND
  - [S5] "<IPv6Address>" includes tag: "<PrefixLength>" with value range from "0" to "128" AND
  - [S6] Device response contains "HTTP/* 200 OK" AND
  - [S7] Device response contains "<RemoveIPAddressFilterResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.7 Persistent Notification Storage Retrieval Test Cases

6.7.1 Feature Level Requirement:

Validated Feature: Persistent Notification Storage Retrieval (PersistentNotificationStorageRetrieval)

Check Condition based on Device Features: Persistent Notification Storage is supported by Device.

Required Number of Devices: 1

Profile C Requirement: Conditional

Profile A Requirement: Conditional

6.7.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using CreatePullPointSubscription operation.
2. Client uses Seek method to change position of the pull pointer to include all NotificationMessages in the persistent storage with UtcTime attribute greater than or equal to the Seek argument.
3. Client uses Pull Point event mechanism to retrieve notification events from Device.
4. Client is considered as supporting Persistent Notification Storage Retrieval if the following conditions are met:
   - Client is able to seek stored events in Device using the Seek operation.
5. Client is considered as NOT supporting Persistent Notification Storage Retrieval if ANY of the following is TRUE:
   - No Valid Device Response to Seek request.

6.7.3 SEEK

Test Label: Persistent Notification Storage Retrieval - Seek

Test Case ID: PERSISTENTNOTIFICATIONSTORAGERETRIEVAL-1

Feature Under Test: Seek (PersistentNotificationStorageRetrieval_Seek)

Test Purpose: To verify that Client is able to seek stored events in Device using Pull Point event mechanism and Seek operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with CreatePullPointSubscription, Seek and PullMessages operations present.

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

1. Client invokes CreatePullPointSubscription message.
2. Device responds with code HTTP 200 OK and CreatePullPointSubscriptionResponse message.
3. Client invokes Seek message to re-adjust the pull pointer into the past.
4. Device responds with code HTTP 200 OK and SeekResponse message.
5. Client invokes PullMessages command with Timeout and MessageLimit elements.
6. Device responds with code HTTP 200 OK and PullMessagesResponse message.

Test Result:

PASS -

- Client **CreatePullPointSubscription** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **CreatePullPointSubscription** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<CreatePullPointSubscription>" tag after the "<Body>" tag AND
  - [S2] Device response contains "HTTP/* 200 OK" AND
  - [S3] Device response contains "<CreatePullPointSubscriptionResponse>" tag AND
- Client **Seek** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **Seek** request in Test Procedure fulfills the following requirements:
  - [S4] Client request contains "<Seek>" tag after the "<Body>" tag AND
  - [S6] Device response contains "HTTP/* 200 OK" AND
  - [S7] Device response contains "<SeekResponse>" tag AND
- Client **PullMessages** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **PullMessages** request in Test Procedure fulfills the following requirements:
  - [S8] Client request contains "<PullMessages>" tag after the "<Body>" tag AND
  - [S11] Device response contains "HTTP/* 200 OK" AND

- [S12] Device response contains "<PullMessagesResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.8 Access Points Control Test Cases

6.8.1 Feature Level Requirement:

Validated Feature: Access Points Control (AccessPointControl)

Check Condition based on Device Features: Enable/Disable Access Point is supported by Device.

Required Number of Devices: 1

Profile C Requirement: Conditional

Profile D Requirement: Conditional

6.8.2 Expected Scenarios Under Test:

1. Client invokes a specific Access Points Control commands in order to change the state of access point.
2. Client is considered as supporting Access Points Control if the following conditions are met:
   - Device returns a valid response to EnableAccessPoint request AND
   - Device returns a valid response to DisableAccessPoint request.
3. Client is considered as NOT supporting Access Points Control if ANY of the following is TRUE:
   - No valid Device response to EnableAccessPoint request OR
   - No valid Device response to DisableAccessPoint request.

6.8.3 DISABLE ENABLE ACCESS POINT

Test Label: Access Points Control - DisableEnableAccessPoint

Test Case ID: ACCESSPOINTCONTROL-1

Feature Under Test: Disable Enable Access Point (AccessPointControl_DisableEnableAccessPoint)

Test Purpose: To verify that Client is able to disable Access Point using DisableAccessPoint operation and enable Access Point using EnableAccessPoint operation.

Pre-Requisite:

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

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

1. Client invokes DisableAccessPoint request message to disable Access Point.
2. Device responds with code HTTP 200 OK and DisableAccessPointResponse message.
3. Client invokes EnableAccessPoint request message to enable access point.
4. Device responds with code HTTP 200 OK and EnableAccessPointResponse message.

Test Result:

PASS -

- Client **DisableAccessPoint** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **DisableAccessPoint** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<DisableAccessPoint>" tag after the "<Body>" tag AND
  - [S2] "<DisableAccessPoint>" includes tag: "<Token>" with non-empty string value of specific token AND
  - [S3] Device response contains "HTTP/* 200 OK" AND
  - [S4] Device response contains "<DisableAccessPointResponse>" tag AND
- Client **EnableAccessPoint** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **EnableAccessPoint** request in Test Procedure fulfills the following requirements:
  - [S5] Client request contains "<EnableAccessPoint>" tag after the "<Body>" tag AND
  - [S6] "<EnableAccessPoint>" includes tag: "<Token>" with token value from DisableAccessPoint operation AND

- [S7] Device response contains "HTTP/* 200 OK" AND
- [S8] Device response contains "<EnableAccessPointResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.9 External Authorization Test Cases

6.9.1 Feature Level Requirement:

Validated Feature: External Authorization (ExternalAuthorization)

Check Condition based on Device Features: External Authorization is supported by Device.

Required Number of Devices: 1

Profile C Requirement: Conditional

6.9.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation.
2. Client receives authorization request from Device and makes a decision about granting access.
3. Client uses Pull Point event mechanism to retrieve notification events from Device.
4. Client receives notifications about access decisions related to External Authorization.
5. Client is considered as supporting External Authorization if the following conditions are met:
   - Client is able to receive authorization request from Device AND
   - Client is able to send authorization decision to Device using **ExternalAuthorization** operation.
6. Client is considered as NOT supporting External Authorization if ANY of the following is TRUE:
   - Client unable to receive authorization request from Device OR
   - No Valid Device Response to **ExternalAuthorization** request.

6.9.3 RECEIVE AUTHORIZATION REQUEST

Test Label: External Authorization - Receive Authorization Request

Test Case ID: EXTERNALAUTHORIZATION-1

Feature Under Test: Receive Authorization Request (ExternalAuthorization_ReceiveAuthRequest)

Test Purpose: To verify that Client is able to receive authorization request from Device using Pull Point event mechanism.

Pre-Requisite:

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

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

1. Client invokes **CreatePullPointSubscription** message without any filter or with appropriate filter.
2. Device responds with code HTTP 200 OK and **CreatePullPointSubscriptionResponse** message.
3. Client invokes **PullMessages** command with Timeout and MessageLimit elements.
4. Device responds with code HTTP 200 OK and **PullMessagesResponse** message with corresponding event topic value.

Test Result:

PASS -

- Client **CreatePullPointSubscription** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **CreatePullPointSubscription** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tev:CreatePullPointSubscription** AND
  - If it contains **tev:Filter/wsnt:TopicExpression** with **Dialect** attribute equal to **http://docs.oasis-open.org/wsn/t-1/TopicExpression/Concrete** then it fulfills the following requirements (else skip the check):

- [S2] **wsnt:TopicExpression** element is equal to **tns1:AccessControl/Request/Credential** OR **tns1:AccessControl/Request/Anonymous** AND

- If it contains **tev:Filter/wsnt:TopicExpression** with **Dialect** attribute equal to **http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet** then it fulfills the following requirements (else skip the check):

  - [S3] **wsnt:TopicExpression** element contains **tns1:AccessControl/Request/Credential** OR **tns1:AccessControl/Request/Anonymous** OR **tns1:AccessControl/Request//.** OR **tns1:AccessControl//.** in expression AND

- Device response on the **CreatePullPointSubscription** request fulfills the following requirements:

  - [S4] It has HTTP 200 response code AND

  - [S5] **soapenv:Body** element has child element **tev:CreatePullPointSubscriptionResponse** AND

- Client **PullMessages** request in Test Procedure fulfills the following requirements:

  - [S6] **soapenv:Body** element has child element **tev:PullMessages** AND

- Device response on the **PullMessages** request fulfills the following requirements:

  - [S7] It has HTTP 200 response code AND

  - [S8] **soapenv:Body** element has child element **tev:PullMessagesResponse** AND

  - [S9] A least one **wsnt:NotificationMessage/wsnt:Topic** element has value equal to EITHER **tns1:AccessControl/Request/Credential** OR **tns1:AccessControl/Request/Anonymous**.

FAIL -

- The Client failed PASS criteria.

6.9.4 SEND AUTHORIZATION DECISION

Test Label: External Authorization - Send Authorization Decision

Test Case ID: EXTERNALAUTHORIZATION-2

Feature Under Test: Send Authorization Desicion (ExternalAuthorization_SendAuthDecision)

Test Purpose: To verify that Client is able to send Granted or Denied decision to Device.

Pre-Requisite:

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

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

1. Client sends ExternalAuthorization message to Device with Granted or Denied decision.
2. Device responds with code HTTP 200 OK and ExternalAuthorizationResponse message.

Test Result:

PASS -

- Client **ExternalAuthorization** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **ExternalAuthorization** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<ExternalAuthorization>" tag after the "<Body>" tag AND
  - [S2] "<ExternalAuthorization>" includes tag: "<AccessPointToken>" with non-empty string value of specific token AND
  - [S3] Device response contains "HTTP/* 200 OK" AND
  - [S4] Device response contains "<ExternalAuthorizationResponse>" tag.

FAIL -

- The Client failed PASS criteria.

7 Test Cases for Profile Optional Features

7.1 Get Services with Capabilities Test Cases

7.1.1 Feature Level Requirement:

Validated Feature: Get Services with Capabilities (GetServicesWithCapabilities)

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

Required Number of Devices: 1

Profile A Requirement: Optional

Profile C Requirement: Optional

Profile D Requirement: Optional

Profile G Requirement: Optional

7.1.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve a service capabilities.
2. Client is considered as supporting Get Services with Capabilities if the following conditions are met:
   - Client is able to retrieve a services capabilities using **GetServices** operation.
3. Client is considered as NOT supporting Get Services with Capabilities if ANY of the following is TRUE:
   - No valid responses for **GetServices** request.

7.1.3 GET SERVICES

Test Label: Get Services with Capabilities - Get Services

Test Case ID: GETSERVICESWITHCAPABILITIES-1

Feature Under Test: Get Services with Capabilities (GetServicesWithCapabilities_GetServicesWithCapabilitiesRequest)

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

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetServices** operation with **tds:IncludeCapability** element equal to true present.
- The Device supports GetServices command.

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

1. Client invokes **GetServices** request message with **tds:IncludeCapability** element equal to true to retrieve redential service capabilities from the Device.
2. Device responds with code HTTP 200 OK and **GetServicesResponse** message.

Test Result:

PASS -

- Client **GetServices** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetServices** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tds:GetServices** AND
  - [S2] It contains **tds:IncludeCapability** element equal to true AND
- Device response on the **GetServices** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tds:GetServicesResponse**.

FAIL -

- The Client failed PASS criteria.

7.2 Set Synchronization Point (Event Service) Test Cases

7.2.1 Feature Level Requirement:

Validated Feature: Set Synchronization Point (SetSynchronizationPoint)

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

Required Number of Devices: 1

Profile A Requirement: Optional

Profile C Requirement: Optional

Profile S Requirement: Optional

Profile G Requirement: Optional

Profile T Requirement: Mandatory

Profile D Requirement: Mandatory

7.2.2 Expected Scenarios Under Test:

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

7.2.3 SET SYNCHRONIZATION POINT (EVENT SERVICE)

Test Label: Set Synchronization Point - Set Synchronization Point

Test Case ID: SETSYNCHRONIZATIONPOINT-1

Feature Under Test: Set Synchronization Point (SetSynchronizationPoint_SetSynchronizationPointAction)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

7.3 Unsubscribe Test Cases

Validated Feature: Unsubscribe (Unsubscribe)

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

Required Number of Devices: 1

Profile A Requirement: Optional

Profile C Requirement: Optional

Profile S Requirement: Optional

Profile G Requirement: Optional

Profile T Requirement: Optional

7.3.1 Expected Scenarios Under Test:

1. Client connects to Device to Unsubscribe subscribtions.
2. Client is considered as supporting Unsubscribe if the following conditions are met:
   - Client is able to unsubscribe subscribtions using **Unsubscribe** operation.
3. Client is considered as NOT supporting Unsubscribe if the following is TRUE:
   - No valid responses for **Unsubscribe** request OR
   - **Unsubscribe** request does not contains valid **wsa:Action** header.

7.3.2 UNSUBSCRIBE

Test Label: Unsubscribe - Unsubscribe

Test Case ID: UNSUBSCRIBE-1

Feature Under Test: Unsubscribe (Unsubscribe_UnsubscribeAction)

Test Purpose: To verify that the Client is able to use **Unsubscribe** operation to terminate a subscribtion.

Pre-Requisite:

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

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

1. Client invokes **Unsubscribe** message with valid **wsa:Action** header to terminete a subscription.
2. Device responses with code HTTP 200 OK and **UnsubscribeResponse** message.

Test Result:

PASS -

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

- Client **Unsubscribe** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **wsnt:Unsubscribe** AND
  - [S2] It contains **wsa:Action** element in header equal to "http://docs.oasis-open.org/wsn/bw-2/SubscriptionManager/UnsubscribeRequest" AND
- Device response on the **Unsubscribe** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **wsnt:UnsubscribeResponse**

FAIL -

- The Client failed PASS criteria.

7.4 System Date and Time Configuration Test Cases

7.4.1 Feature Level Requirement:

Validated Feature: System Date and Time Configuration (SystemDateAndTimeConfiguration)

Check Condition based on Device Features: Profile A OR Profile C OR Profile G OR Profile S OR Profile T OR Profile D

Required Number of Devices: 1

Profile A Requirement: Conditional

Profile C Requirement: Optional

Profile G Requirement: Optional

Profile S Requirement: Optional

7.4.2 Expected Scenarios Under Test:

1. Client connects to Device to configure system date and time.
2. Client is considered as supporting System Date and Time Configuration if the following conditions are met:
   - Client is able to retrieve a system date and time using **GetSystemDateAndTime** operation AND
   - Client is able to configure a system date and time using EITHER **SetSystemDateAndTime** operation OR **SetNTP** operation.

3. Client is considered as NOT supporting System Date and Time Configuration if ANY of the following is TRUE:
   - No valid responses for **GetSystemDateAndTime** request OR
   - No valid responses for **SetSystemDateAndTime** request if detected AND
   - Client does not support NTP feature.

7.4.3 GET SYSTEM DATE AND TIME

Test Label: System Date and Time Configuration - Get System Date And Time

Test Case ID: SYSTEMDATEANDTIMECONFIGURATION-1

Feature Under Test: Get System Date And Time (SystemDateAndTimeConfiguration_GetSystemDateAndTime)

Test Purpose: To verify that Device system date and time is received by Client using the **GetSystemDateAndTime** operation.

Pre-Requisite:

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

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

1. Client invokes **GetSystemDateAndTime** request message to retrieve system date and time from the Device.
2. Device responds with code HTTP 200 OK and **GetSystemDateAndTimeResponse** message.

Test Result:

PASS -

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

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

FAIL -

- The Client failed PASS criteria.

7.4.4 SET SYSTEM DATE AND TIME

Test Label: System Date and Time Configuration - Set System Date And Time

Test Case ID: SYSTEMDATEANDTIMECONFIGURATION-2

Feature Under Test: Set System Date And Time (SystemDateAndTimeConfiguration_SetSystemDateAndTime)

Test Purpose: To verify that Client is able to configure system date and time on Device using the **SetSystemDateAndTime** operation.

Pre-Requisite:

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

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

1. Client invokes **SetSystemDateAndTime** request message to set Device system date and time.
2. Device responds with code HTTP 200 OK and **SetSystemDateAndTimeResponse** message.

Test Result:

PASS -

- Client **SetSystemDateAndTime** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetSystemDateAndTime** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tds:SetSystemDateAndTime** AND
  - [S2] If **tds:DateTimeType** element value is equal to "Manual" THEN **tds:SetSystemDateAndTime** contains **tds:UTCDateTime** element AND

- Device response on the **SetSystemDateAndTime** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tds:SetSystemDateAndTimeResponse**.

FAIL -

- The Client failed PASS criteria.

7.5 Hostname Configuration Test Cases

7.5.1 Feature Level Requirement:

Validated Feature: Hostname Configuration (HostnameConfiguration)

Check Condition based on Device Features: None

Required Number of Devices: 1

Profile A Requirement: Optional

Profile C Requirement: Optional

Profile G Requirement: Optional

Profile S Requirement: Optional

7.5.2 Expected Scenarios Under Test:

1. Client connects to Device to configure hostname.
2. Client is considered as supporting Hostname Configuration if the following conditions are met:
   - Client is able to retrieve a hostname information from the Device using **GetHostname** operation AND
   - Client is able set a network hostname on the Device using **SetHostname** operation.
3. Client is considered as NOT supporting Hostname Configuration if ANY of the following is TRUE:
   - No valid responses for **GetHostname** request OR
   - No valid responses for **SetHostname** request.

7.5.3 GET HOSTNAME

Test Label: Hostname Configuration - Get Hostname

Test Case ID: HOSTNAMECONFIGURATION-1

Feature Under Test: Get Hostname (HostnameConfiguration_GetHostname)

Test Purpose: To verify that hostname settings of the Device are received by Client using the **GetHostname** operation.

Pre-Requisite:

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

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

1. Client invokes **GetHostname** request message to retrieve hostname from the Device.
2. Device responds with code HTTP 200 OK and **GetHostnameResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

7.5.4 SET HOSTNAME

Test Label: Hostname Configuration - Set Hostname

Test Case ID: HOSTNAMECONFIGURATION-2

Feature Under Test: Set Hostname (HostnameConfiguration_SetHostname)

Test Purpose: To verify that Client is able to set the Hostname settings on Device using the **SetHostname** operation.

Pre-Requisite:

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

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

1. Client invokes **SetHostname** request message to set hostname on the Device.
2. Device responds with code HTTP 200 OK and **SetHostnameResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

7.6 DNS Configuration Test Cases

7.6.1 Feature Level Requirement:

Validated Feature: DNS Configuration (DNSConfiguration)

Check Condition based on Device Features: None

Required Number of Devices: 1

Profile A Requirement: Optional

Profile C Requirement: Optional

Profile G Requirement: Optional

Profile S Requirement: Optional

7.6.2 Expected Scenarios Under Test:

1. Client connects to Device to configure a domain name server.
2. Client is considered as supporting DNS Configuration if the following conditions are met:
   - Client is able to get DNS settings from the Device using **GetDNS** operation AND
   - Client is able set DNS settings on the Device using **SetDNS** operation.
3. Client is considered as NOT supporting DNS Configuration if ANY of the following is TRUE:
   - No valid responses for **GetDNS** request OR
   - No valid responses for **SetDNS** request.

7.6.3 GET DNS

Test Label: DNS Configuration - Get DNS

Test Case ID: DNSCONFIGURATION-1

Feature Under Test: Get DNS (DNSConfiguration_GetDNS)

Test Purpose: To verify that DNS settings of Device are received by Client using the **GetDNS** operation.

Pre-Requisite:

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

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

1. Client invokes **GetDNS** request message to retrieve DNS settings from the Device.
2. Device responds with code HTTP 200 OK and **GetDNSResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

7.6.4 SET DNS

Test Label: DNS Configuration - Set DNS

Test Case ID: DNSCONFIGURATION-2

Feature Under Test: Set DNS (DNSConfiguration_SetDNS)

Test Purpose: To verify that Client is able to set the DNS settings on Device using the **SetDNS** operation.

Pre-Requisite:

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

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

1. Client invokes **SetDNS** request message to set hostname on the Device.
2. Device responds with code HTTP 200 OK and **SetDNSResponse** message.

Test Result:

PASS -

- Client **SetDNS** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetDNS** request in Test Procedure fulfills the following requirements:

- [S1] **soapenv:Body** element has child element **tds:SetDNS** AND

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

FAIL -

- The Client failed PASS criteria.

7.7 Network Protocols Configuration Test Cases

7.7.1 Feature Level Requirement:

Validated Feature: Network Protocols Configuration (NetworkProtocolsConfiguration)

Check Condition based on Device Features: None

Required Number of Devices: 1

Profile A Requirement: Optional

Profile C Requirement: Optional

Profile G Requirement: Optional

Profile S Requirement: Optional

7.7.2 Expected Scenarios Under Test:

1. Client connects to Device to configure a network protocols.
2. Client is considered as supporting Network Protocols Configuration if the following conditions are met:
   - Client is able to get defined network protocols from the Device using **GetNetworkProtocols** operation AND
   - Client is able configures defined network protocols on the Device using **SetNetworkProtocols** operation.
3. Client is considered as NOT supporting Network Protocols Configuration if ANY of the following is TRUE:

- No valid responses for **GetNetworkProtocols** request OR
- No valid responses for **SetNetworkProtocols** request.

7.7.3 GET NETWORK PROTOCOLS

Test Label: Network Protocols Configuration - Get Network Protocols

Test Case ID: NETWORKPROTOCOLSCONFIGURATION-1

Feature Under Test: Get Network Protocols (NetworkProtocolsConfiguration_GetNetworkProtocols)

Test Purpose: To verify that network protocols of Device are received by Client using the **GetNetworkProtocols** operation.

Pre-Requisite:

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

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

1. Client invokes **GetNetworkProtocols** request message to retrieve network protocols from the Device.
2. Device responds with code HTTP 200 OK and **GetNetworkProtocolsResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

7.7.4 SET NETWORK PROTOCOLS

Test Label: Network Protocols Configuration - Set Network Protocols

Test Case ID: NETWORKPROTOCOLSCONFIGURATION-2

Feature Under Test: Set Network Protocols (NetworkProtocolsConfiguration_SetNetworkProtocols)

Test Purpose: To verify that Client is able to configure defined network protocols on Device using the **SetNetworkProtocols** operation.

Pre-Requisite:

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

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

1. Client invokes **SetNetworkProtocols** request message to set hostname on the Device.
2. Device responds with code HTTP 200 OK and **SetNetworkProtocolsResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

8 Supplementary Features and Test Cases

8.1 METADATA STREAMING USING MEDIA2

Test Label: Metadata Streaming Using Media2

Test Case ID: MEDIA2_METADATASTREAMING-1

Feature Under Test: Metadata Streaming (Media2_MetadataStreaming_MetadataStreamingUsingMedia2)

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

Pre-Requisite:

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

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

1. Client invokes **GetStreamUri** request message for Media2 service for media profile that contains Metadata Configuration. GetStreamUri request is set for RtspUnicast OR RtspMulticast OR RTSP OR RtspOverHttp transport.
2. Device responds with code HTTP 200 OK and **GetStreamUriResponse** message.
3. Client invokes **RTSP DESCRIBE** request to retrieve media stream description.
4. Device responds with code RTSP 200 OK and SDP information with Media Type: "application" and with encoding name "vnd.onvif.metadata" or "vnd.onvif.metadata.gzip" or "vnd.onvif.metadata.exi.onvif" or "vnd.onvif.metadata.exi.ext".
5. Client invokes **RTSP SETUP** request without "onvif-replay" Require header and with transport parameter element to to set media session parameters for metadata streaming.
6. Device responds with code RTSP 200 OK.
7. Client invokes **RTSP PLAY** request without "onvif-replay" Require header to start media stream.
8. Device responds with code RTSP 200 OK.
9. Client invokes **RTSP TEARDOWN** request to terminate the RTSP session.
10. If Device sends response to RTSP TEARDOWN, it has code RTSP 200 OK or RTSP 454.

Test Result:

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

PASS -

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

- [S12] It invoked for the same Device as for the Client **RTSP SETUP** request AND
- [S13] It invoked after the Client **RTSP SETUP** request AND
- [S14] RTSP address that was used to send it is correspond to corresponding media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
- [S15] It does not contain **Require** request header field with value is equal to "onvif-replay" AND

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

FAIL -

- The Client failed PASS criteria.

Annex A Test for Appendix A

A.1 Required Number of Devices Summary

Required number of devices and Device feature dependency used in this test specification are listed in the Table.

Table A.1. Required Number of Devices Summary

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
tc.HTTPDigest	HTTP Digest	3	Digest	Digest
tc.Capabilities	Capabilities	3	None	All
tc.GetServices	Get Services	3	GetServices is supported by Device.	GetServices
tc.EventHandling	Event Handling	3	Pull Point Notification OR WS Basic Notification OR Profile S OR Metadata under Media2 service is supported by Device.	no UnsupportedPullPointNotification OR WSBasicNotification OR Profile S OR Media2_Metadata
tc.KeepAliveForPullPointEventHandling	Keep Alive for Pull Point Event Handling	3	Pull Point Notification is supported by Device.	no UnsupportedPullPointNotificatio
tc.AccessPointInformation	Access Point Information	3	Access Control Service is supported by Device.	AccessControlService
tc.DoorInformation	Door Information	3	Door Control Service is supported by Device.	DoorControlService

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
tc.AreaInformation	Area Information	3	Access Control Service is supported by Device. Area Entity is supported by Device.	AccessControlService AND AreaEntity
tc.SystemComponentState	System Component State	3	Access Control Service and Door Control Service are supported by Device.	AccessControlService AND DoorControlService
tc.DoorControl	Door Control	3	Door Control Service and Access Door and Lock Door and Unlock Door are supported by Device.	DoorControlService AND AccessDoor AND LockDoor AND UnlockDoor
tc.AccessControlDecisions	Access Control Decisions	3	Check Condition based on Device Features: Access Control Service and tns1:AccessControl/AccessGranted/Credential and tns1:AccessControl/AccessDenied/Credential and tns1:AccessControl/AccessGranted/Anonymous and tns1:AccessControl/AccessDenied/AnonymousEvent tns1:AccessC	AccessControlService AND AccessGrantedCredentialEvent AND AccessDeniedCredentialEvent AND AccessGrantedAnonymousEvent AND AccessDeniedAnonymousEvent AccessDeniedCredentialCredentialNotFoundCardEvent AND AccessTakenAnonymousEvent

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
			ontrol/Access Denied/Credential/CredentialNotFoundCard and tns1:AccessControl/AccessTaken/Anonymous and tns1:AccessControl/AccessTaken/Credential and tns1:AccessControl/AccessNotTaken/Anonymous and tns1:AccessControl/AccessNotTaken/CredentialEvent are supported by Device.	AND AccessTakenCredentialEvent AND AccessNotTakenAnonymousEvent AND AccessNotTakenCredentialEvent
tc.AccessPointConfigurationChangeNotifications	Access Point Information - Configuration Change Notifications	3	Access Control Service is supported by Device.	AccessControlService
tc.DoorConfigurationChangeNotifications	Door Information - Configuration Change Notifications	3	Door Control Service is supported by Device.	DoorControlService
tc.AreaConfigurationChangeNotifications	Area Information - Configuration Change Notifications	3	Access Control Service is supported by Device. Area Entity is supported by Device.	AccessControlService AND AreaEntity

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
tc.DuressNotifications	Duress Notifications	3	Duress is supported by Device.	DuressEvent
tc.Discovery	Discovery	3	Discovery	All
tc.DeviceDiscoveryTypeFilter	Device Discovery Type Filter	3	Device Discovery Type is supported by Device.	DiscoveryTypesTdsDevice
tc.NetworkConfiguration	Network Configuration	3	Network Configuration	no NetworkConfigNotSupported
tc.System	System	3	None	All
tc.UserHandling	User Handling	3	User Configuration	no UserConfigNotSupported
tc.IPAddressFiltering	IP Address Filtering	1	IP Filter is supported by Device.	IPFilter
tc.PersistentNotificationStorageRetrieval	Persistent Notification Storage Retrieval	1	Persistent Notification Storage is supported by Device.	PersistentNotificationStorage
tc.AccessPointControl	Access Points Control	1	Enable/Disable Access Point is supported by Device.	EnableDisableAccessPoint
tc.ExternalAuthorization	External Authorization	1	External Authorization is supported by Device.	ExternalAuthorization
tc.GetServicesWithCapabilities	Get Services with Capabilities	1	GetServices is supported by Device.	GetServices
tc.SetSynchronizationPoint	Set Synchronization Point (Event Service)	1	Pull Point Notification OR WS-Basic Notification is	no UnsupportedPullPointNotification OR WSBasicNotification

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
			supported by Device.	
tc.SystemDateAndTimeConfiguration	System Date and Time Configuration	1	Profile A OR Profile C OR Profile G OR Profile S OR Profile T OR Profile D	Profile A OR Profile C OR Profile G OR Profile S OR Profile T OR Profile D
tc.HostnameConfiguration	Hostname Configuration	1	None	All
tc.DNSConfiguration	DNS Configuration	1	None	All
tc.NetworkProtocolsConfiguration	Network Protocols Configuration	1	None	All