ONVIF®

Profile D Client Test Specification

Version 21.06

June 2021

© 2021 ONVIF, Inc. All rights reserved.

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

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

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

REVISION HISTORY

Vers.	Date	Description
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	May 27, 2021	Feedback feature was updated according to #423: Dependency on AccessControlIdentifier feature was removed.
21.06	May 27, 2021	The following were cnaged according to #421: Feature removed: Access Control with Identifier Access Feature added: Access Control With Identifier Request (Credential) Feature added: Access Control With Identifier Request (Anonymous)
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	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	Keep Alive for Pull Point Event Handling Feature: Check Condition based on Device Features was changed according to #325.
20.06	May 29, 2020	Feedback mandatory feature was added according to #386.
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	Dec 06, 2019	The following mandatory features were added according to #345: Get AccessPoint State Access Point State Changed Event Get Door State

		Door Mode State Changed Event Door Physical State Changed Event Lock Physical State Changed Event Double Lock Physical State Changed Event Door Alarm State Changed Event Door Tamper State Changed Event Door Fault State Changed Event
19.12	Dec 02, 2019	Get Services with Capabilities feature was moved from mandatory to optional according to #344
19.12	Dec 02, 2019	Digest Authentication for RTSP (Profile D) feature was added into Test Cases for Profile Conditional Features section according to #343
19.12	Nov 08, 2019	Test Cases for Profile Conditional Features section with the following features were added in the scope of #324: System (included) User Handling (included) Access Point Management (new) Access Point Control (included) Door Management (new) Credential Format Types (new) Credential Whitelisting (new) Credential Blacklisting (new)
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 23, 2019	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 feaures and related test cases required for testing Profile D features of a Client application e.g. door and access point configuration and control, black- and whitelist management, access control decisions. 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 D Client Test Specification defines and regulates the conformance testing procedure for the ONVIF conformant Clients in the scope of Profile D 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 D features according to ONVIF Profile Specifications.

The principal intended purposes are:

- Provide self-assessment tool for implementations.
- Provide comprehensive test suite coverage for Profile D 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 D Client conformance.

1.2.1 HTTP Digest

HTTP Digest section defines security mechanism for HTTP Digest Authentication.

1.2.2 Get Services

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

1.2.3 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.

1.2.4 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.

1.2.5 Network Configuration

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

1.2.6 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.

1.2.7 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.

1.2.8 Access Point Information

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

1.2.9 Access Point Information - Configuration Change Notifications

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

1.2.10 Get Access Point State

Get Access Point State section specifies Client ability to request information about the state of Access Points using the GetAccessPointState operation.

1.2.11 Access Point State Changed Event

Access Point State Changed Event section specifies Client ability to receive tns1:AccessPoint/State/Enabled events.

1.2.12 Get Door State

Get Door State section specifies Client ability to request information about the state of Doors using the GetDoorState operation.

1.2.13 Door Mode State Changed Event

Door Mode State Changed Event section specifies Client ability to receive tns1:Door/State/DoorMode events.

1.2.14 Door Physical State Changed Event

Door Physical State Changed Event section specifies Client ability to receive tns1:Door/State/DoorPhysicalState events.

1.2.15 Lock Physical State Changed Event

Lock Physical State Changed Event section specifies Client ability to receive tns1:Door/State/LockPhysicalState events.

1.2.16 Double Lock Physical State Changed Event

Double Lock Physical State Changed Event section specifies Client ability to receive tns1:Door/State/DoubleLockPhysicalState events.

1.2.17 Door Alarm State Changed Event

Door Alarm State Changed Event section specifies Client ability to receive tns1:Door/State/DoorAlarm events.

1.2.18 Door Tamper State Changed Event

Door Tamper State Changed Event section specifies Client ability to receive tns1:Door/State/DoorTamper events.

1.2.19 Door Fault State Changed Event

Door Fault State Changed Event section specifies Client ability to receive tns1:Door/State/DoorFault events.

1.2.20 Access Control With Anonymous Access

Access Control With Anonymous Access section specifies Client ability to control anonymous access using external authorization functionality of a Device.

1.2.21 Access Control With Identifier Request (Credential)

Access Control With Identifier Request (Credential) section specifies Client ability to control Identifier access with credential using external authorization functionality of a Device.

1.2.22 Access Control With Identifier Request (Anonymous)

Access Control With Identifier Request (Anonymous) section specifies Client ability to control Identifier access to anonymous using external authorization functionality of a Device.

1.2.23 Feedback

Feedback section specifies Client ability to send feedback indications using the Feedback operation.

1.2.24 Access Taken With Anonymous Access

Access Taken With Anonymous Access section specifies Client ability to receive access taken and access not taken notifications for anonymous access.

1.2.25 Access Taken With Identifier Access

Access Taken With Identifier Access section specifies Client ability to receive access taken and access not taken notifications for identifier access.

1.2.26 Door Information

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

1.2.27 Door Information - Configuration Change Notifications

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

1.2.28 Door Control

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

1.3 Test Cases for Profile Conditional Features

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

1.3.1 HTTP Digest Authentication for RTSP

HTTP Digest Authentication for RTSP section defines security mechanism for Digest Authentication for RTSP.

1.3.2 System

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

1.3.3 User Handling

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

1.3.4 Access Point Management

Access Point Management section specifies Client ability to create, modify, and delete Access Points.

1.3.5 Access Point Control

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

1.3.6 Door Management

Door Management section specifies Client ability to create, modify, and delete Doors.

1.3.7 Credential Format Types

Credential Format Types section specifies Client ability to retrieve supported credential format types from a device Device.

1.3.8 Credential Whitelisting

Credential Whitelisting section specifies Client ability to manage a Whitelist on a Device.

1.3.9 Credential Blacklisting

Credential Blacklisting section specifies Client ability to manage a Blacklist on a Device.

1.4 Test Cases for Profile Optional Features

This section defines test cases which are optional for Profile D 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.

1.5 Supplementary Features and Test Cases

This section defines supplementary features and test cases which are not the part of profile, but Profile D 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 D Specification:

  [TODO: put link to profile page] [https://www.onvif.org/profiles/profile-d/]

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 D	The Profile D Specification.

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.

3.4 Namespaces

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

Table 3.1. Defined namespaces in this specification

Prefix	Namespace URI	Description
soapenv	http://www.w3.org/2003/05/soap-envelope	Envelope namespace as defined by SOAP 1.2 [SOAP 1.2, Part 1]
xs	http://www.w3.org/2001/XMLSchema	Instance namespace as defined by XS [XML-Schema, Part1] and [XMLSchema,Part 2]
xsi	http://www.w3.org/2001/XMLSchema-instance	XML schema instance namespace
tns1	http://www.onvif.org/ver10/topics	The namespace for the ONVIF topic namespace
tt	http://www.onvif.org/ver10/schema	ONVIF XML schema descriptions
tds	http://www.onvif.org/ver10/device/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].
d	http://schemas.xmlsoap.org/ws/2005/04/discovery	Device discovery namespace as defined by [WS-Discovery]
wsadis	http://schemas.xmlsoap.org/ws/2004/08/addressing	Device addressing namespace referred in WS-Discovery [WS-Discovery]
tac	http://www.onvif.org/ver10/accesscontrol/wsdl	The namespace for the WSDL access control service

Prefix	Namespace URI	Description
tdc	http://www.onvif.org/ver10/doorcontrol/wsdl	The namespace for the WSDL door control service
tcr	http://www.onvif.org/ver10/credential/wsdl	The namespace for the WSDL credential 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 D is an ONVIF Client that at least supports the following features:

- HTTP Digest authentication.
- Receiving of Device services using GetServices operation.
- Discovering Devices using WS-Discovery.
- Configuration of a network interfaces using GetNetworkInterfaces and SetNetworkInterfaces operations.
- Configuration of a network default gateway using GetNetworkDefaultGateway and SetNetworkDefaultGateway operations.
- Event handling using the SetSynchronizationPoint, CreatePullPointSubscription and PullMessage operations.
- Retrieving of Access Point list using GetAccessPointInfoList operation.
- Receiving of Access Point configuration change notifications:
  - tns1:Configuration/AccessPoint/Changed
  - tns1:Configuration/AccessPoint/Removed
- Receiving of Access Point state using GetAccessPointState operation.
- Receiving of Access Point state notifications:
  - tns1:AccessPoint/State/Enabled
- Anonymous access control using ExternalAuthorization operation and related notifications:
  - tns1:AccessControl/Request/Anonymous
  - tns1:AccessControl/Request/Timeout
- Access control by Identifier using ExternalAuthorization operation and related notifications:
  - tns1:AccessControl/Request/Identifier
  - tns1:AccessControl/Request/Timeout

- Receiving of access decisions notifications:
  - tns1:AccessControl/AccessTaken/Identifier
  - tns1:AccessControl/AccessNotTaken/Identifier
  - tns1:AccessControl/AccessTaken/Anonymous
  - tns1:AccessControl/AccessNotTaken/Anonymous
- Retrieving of Door list using GetDoorInfoList operation.
- Receiving of Door configuration change notifications:
  - tns1:Configuration/Door/Changed
  - tns1:Configuration/Door/Removed
- Receiving of Door state using GetDoorState operation.
- Receiving of Door state notifications:
  - tns1:Door/State/DoorMode
  - tns1:Door/State/DoorPhysicalState
  - tns1:Door/State/LockPhysicalState
  - tns1:Door/State/DoubleLockPhysicalState
  - tns1:Door/State/DoorAlarm
  - tns1:Door/State/DoorTamper
  - tns1:Door/State/DoorFault
- Control of Doors using AccessDoor, LockDoor, and UnlockDoor operations.

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 D, 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 Q 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)

Profile S Normative Reference: Mandatory

Profile G Normative Reference: Mandatory

Profile C Normative Reference: Mandatory

Profile Q Normative Reference: Mandatory

Profile A Normative Reference: Mandatory

Profile T Normative Reference: Mandatory

Profile D Normative Reference: Mandatory

Profile M Normative Reference: Mandatory

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 Get Services Test Cases

5.2.1 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 Q Requirement: Mandatory

Profile T Requirement: Mandatory

Profile M Requirement: Mandatory

5.2.2 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).

5.2.3 GET SERVICES

Test Label: Capabilities - Determine the available Services

Test Case ID: CAPABILITIES-1

Feature Under Test: Get Services (Capabilities_GetServicesRequest)

Profile S Normative Reference: Mandatory

Profile G Normative Reference: Mandatory

Profile C Normative Reference: Mandatory

Profile Q Normative Reference: Mandatory

Profile A Normative Reference: Mandatory

Profile T Normative Reference: Mandatory

Profile M Normative Reference: Mandatory

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.

5.3 Discovery Test Cases

5.3.1 Feature Level Requirement:

Validated Feature: Discovery (Discovery)

Check Condition based on Device Features: None

Required Number of Devices: 3

Profile S Requirement: Conditional

Profile C Requirement: Conditional

Profile G Requirement: Conditional

Profile A Requirement: Mandatory

Profile Q Requirement: Mandatory

Profile T Requirement: Mandatory

Profile D Requirement: Mandatory

Profile M Requirement: Mandatory

5.3.2 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.

5.3.3 WS-DISCOVERY

Test Label: Discovery - WS-Discovery

Test Case ID: DISCOVERY-1

Feature Under Test: WS-Discovery (Discovery_WSDiscovery)

Profile S Normative Reference: Conditional

Profile G Normative Reference: Conditional

Profile C Normative Reference: Conditional

Profile Q Normative Reference: Mandatory

Profile A Normative Reference: Mandatory

Profile T Normative Reference: Mandatory

Profile D Normative Reference: Mandatory

Profile M Normative Reference: Mandatory

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.

5.4 Device Discovery Type Filter Test Cases

5.4.1 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 Q Requirement: Mandatory

Profile G Requirement: Conditional

Profile T Requirement: Mandatory

Profile M Requirement: Mandatory

5.4.2 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.

5.4.3 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)

Profile S Normative Reference: None

Profile G Normative Reference: Mandatory

Profile C Normative Reference: Mandatory

Profile Q Normative Reference: Mandatory

Profile A Normative Reference: Mandatory

Profile T Normative Reference: Mandatory

Profile D Normative Reference: Mandatory

Profile M Normative Reference: Mandatory

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.

5.5 Network Configuration Test Cases

5.5.1 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 Q Requirement: Conditional

Profile S Requirement: Conditional

Profile T Requirement: Mandatory

Profile M Requirement: Mandatory

5.5.2 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.

5.5.3 GET NETWORK INTERFACES

Test Label: Network Configuration - Get Network Interfaces

Test Case ID: NETWORKCONFIGURATION-1

Feature Under Test: Get Network Interfaces (NetworkConfiguration_GetNetworkInterfaces)

Profile S Normative Reference: Conditional

Profile G Normative Reference: Conditional

Profile C Normative Reference: Conditional

Profile Q Normative Reference: Conditional

Profile A Normative Reference: Conditional

Profile T Normative Reference: Mandatory

Profile D Normative Reference: Mandatory

Profile M Normative Reference: Mandatory

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.

5.5.4 SET NETWORK INTERFACES

Test Label: Network Configuration - Set Network Interfaces

Test Case ID: NETWORKCONFIGURATION-2

Feature Under Test: Set Network Interfaces (NetworkConfiguration_SetNetworkInterfaces)

Profile S Normative Reference: Conditional

Profile G Normative Reference: Conditional

Profile C Normative Reference: Conditional

Profile Q Normative Reference: Conditional

Profile A Normative Reference: Conditional

Profile T Normative Reference: Mandatory

Profile D Normative Reference: Mandatory

Profile M Normative Reference: Mandatory

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.

5.5.5 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)

Profile S Normative Reference: Conditional

Profile G Normative Reference: Conditional

Profile C Normative Reference: Conditional

Profile Q Normative Reference: Conditional

Profile A Normative Reference: Conditional

Profile T Normative Reference: Mandatory

Profile D Normative Reference: Mandatory

Profile M Normative Reference: Mandatory

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.

5.5.6 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)

Profile S Normative Reference: Conditional

Profile G Normative Reference: Conditional

Profile C Normative Reference: Conditional

Profile Q Normative Reference: Conditional

Profile A Normative Reference: Conditional

Profile T Normative Reference: Mandatory

Profile D Normative Reference: Mandatory

Profile M Normative Reference: Mandatory

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.

5.6 Event Handling Test Cases

5.6.1 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 Q Requirement: Conditional

Profile A Requirement: Mandatory

Profile C Requirement: Mandatory

Profile T Requirement: Mandatory

Profile D Requirement: Mandatory

5.6.2 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.

5.6.3 PULLPOINT

Test Label: Event Handling - Pull Point

Test Case ID: EVENTHANDLING-1

Feature Under Test: Pull Point (EventHandling_PullPoint)

Profile S Normative Reference: Conditional

Profile G Normative Reference: Conditional

Profile C Normative Reference: Governed by business rule #3

Profile Q Normative Reference: Conditional

Profile A Normative Reference: Mandatory

Profile T Normative Reference: Mandatory

Profile D Normative Reference: Mandatory

Profile M Normative Reference: Conditional

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.

5.6.4 BASE NOTIFICATION

Test Label: Event Handling - Basic Notification

Test Case ID: EVENTHANDLING-2

Feature Under Test: Base Notification (EventHandling_WSBaseNotification)

Profile S Normative Reference: Conditional

Profile G Normative Reference: Conditional

Profile C Normative Reference: Governed by business rule #3

Profile Q Normative Reference: None

Profile A Normative Reference: None

Profile T Normative Reference: None

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.

5.6.5 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)

Profile S Normative Reference: Conditional

Profile G Normative Reference: None

Profile C Normative Reference: None

Profile Q Normative Reference: None

Profile A Normative Reference: None

Profile T Normative Reference: None

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.

5.7 Set Synchronization Point (Event Service) Test Cases

5.7.1 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 Q Requirement: Optional

Profile G Requirement: Optional

Profile T Requirement: Mandatory

Profile D Requirement: Mandatory

5.7.2 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.

5.7.3 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)

Profile A Normative Reference: Mandatory

Profile C Normative Reference: Mandatory

Profile S Normative Reference: Conditional

Profile Q Normative Reference: Optional

Profile G Normative Reference: Conditional

Profile T Normative Reference: Mandatory

Profile D Normative Reference: Mandatory

Profile M Normative Reference: Mandatory

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.

5.8 Access Point Information Test Cases

5.8.1 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

5.8.2 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**.

5.8.3 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)

Profile C Normative Reference: Mandatory

Profile D Normative Reference: Mandatory

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.

5.9 Access Point Information - Configuration Change Notifications Test Cases

5.9.1 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

5.9.2 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.

5.10 Get Access Point State Test Cases

5.10.1 Feature Level Requirement:

Validated Feature: Get Access Point State (GetAccessPointState)

Check Condition based on Device Features: AccessPoint entity is supported by Device.

Required Number of Devices: 3

Profile D Requirement: Mandatory

5.10.2 Expected Scenarios Under Test:

1. Client connects to Device to get state of an access point.
2. Client is considered as supporting Get Access Point State if the following conditions are met:
   - Client is able to get an access point state using the **GetAccessPointState** operation.
3. Client is considered as NOT supporting Get Access Point State if the following conditions are met:
   - Client is not able to get an access point state using the **GetAccessPointState** operation.

5.10.3 GET ACCESS POINT STATE

Test Label: Get Access Point State

Test Case ID: GETACCESSPOINTSTATE-1

Feature Under Test: Get Access Point State (GetAccessPointState_GetAccessPointStateRequest)

Profile D Normative Reference: Mandatory

Test Purpose: To verify that Client is able to retrieve access point state using the **GetAccessPointState** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetAccessPointState** operation.
- Device supports AccessPoint entity.

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

1. Client invokes **GetAccessPointState** request message to retrieve access point state from the Device.
2. Device responds with code HTTP 200 OK and **GetAccessPointStateResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

5.11 Access Point State Changed Event Test Cases

5.11.1 Feature Level Requirement:

Validated Feature: Access Point State Changed Event (AccessPointStateChangedEvent)

Check Condition based on Device Features: AccessPointStateEnabled event is supported by Device.

Required Number of Devices: 3

Profile D Requirement: Mandatory

5.11.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation to get notifications about the state of access points.
2. Client is considered as supporting Access Point State Changed Event if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is able to receive **tns1:AccessPoint/State/Enabled** notification (SystemComponentState_AccessPointStateEnabledNotification feature) about a state of access point if Device supports AccessPointStateEnabledEvent.
3. Client is considered as NOT supporting Access Point State Changed Event if the following conditions are met:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is NOT able to receive **tns1:AccessPoint/State/Enabled** notification about a state of access point (SystemComponentState_AccessPointStateEnabledNotification feature).

5.12 Get Door State Test Cases

5.12.1 Feature Level Requirement:

Validated Feature: Get Door State (GetDoorState)

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

Required Number of Devices: 3

Profile D Requirement: Mandatory

5.12.2 Expected Scenarios Under Test:

1. Client connects to Device to get state of a door.

2. Client is considered as supporting Get Door State if the following conditions are met:
   - Client is able to get a door state using the **GetDoorState** operation.
3. Client is considered as NOT supporting Get Door State if the following conditions are met:
   - Client is not able to get a door state using the **GetDoorState** operation.

5.12.3 GET DOOR STATE

Test Label: Get Door State

Test Case ID: GETDOORSTATE-1

Feature Under Test: Get Door State (GetDoorState_GetDoorStateRequest)

Profile D Normative Reference: Mandatory

Test Purpose: To verify that Client is able to retrieve door state using the **GetDoorState** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetDoorState** operation.
- Device supports Door entity.

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

1. Client invokes **GetDoorState** request message to retrieve door state from the Device.
2. Device responds with code HTTP 200 OK and **GetDoorStateResponse** message.

Test Result:

PASS -

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

- [S3] **soapenv:Body** element has child element **tdc:GetDoorStateResponse**.

FAIL -

- The Client failed PASS criteria.

5.13 Door Mode State Changed Event Test Cases

5.13.1 Feature Level Requirement:

Validated Feature: Door Mode State Changed Event (DoorModeStateChangedEvent)

Check Condition based on Device Features: DoorMode event is supported by Device.

Required Number of Devices: 3

Profile D Requirement: Mandatory

5.13.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation to get notifications about the state of door mode.
2. Client is considered as supporting Door Mode State Changed Event if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is able to receive **tns1:Door/State/DoorMode** notification (SystemComponentState_DoorStateDoorModeNotification feature) about a state of door mode if Device supports DoorModeEvent.
3. Client is considered as NOT supporting Door Mode State Changed Event if the following conditions are met:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is NOT able to receive **tns1:Door/State/DoorMode** notification (SystemComponentState_DoorStateDoorModeNotification feature) about a state of door mode.

5.14 Door Physical State Changed Event Test Cases

5.14.1 Feature Level Requirement:

Validated Feature: Door Physical State Changed Event (DoorPhysicalStateChangedEvent)

Check Condition based on Device Features: DoorPhysicalState event is supported by Device.

Required Number of Devices: 3

Profile D Requirement: Mandatory

5.14.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation to get notifications about the physical state of door.
2. Client is considered as supporting Door Physical State Changed Event if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is able to receive **tns1:Door/State/DoorPhysicalState** notification (SystemComponentState_DoorStateDoorPhysicalStateNotification feature) if Device supports DoorPhysicalStateEvent.
3. Client is considered as NOT supporting Door Physical State Changed Event if the following conditions are met:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is NOT able to receive **tns1:Door/State/DoorPhysicalState** notification (SystemComponentState_DoorStateDoorPhysicalStateNotification feature).

5.15 Lock Physical State Changed Event Test Cases

5.15.1 Feature Level Requirement:

Validated Feature: Lock Physical State Changed Event (LockPhysicalStateChangedEvent)

Check Condition based on Device Features: LockPhysicalState event is supported by Device.

Required Number of Devices: 3

Profile D Requirement: Mandatory

5.15.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation to get notifications about the physical state of lock.
2. Client is considered as supporting Lock Physical State Changed Event if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is able to receive **tns1:Door/State/LockPhysicalState** notification (SystemComponentState_DoorStateLockPhysicalStateNotification feature) if Device supports LockPhysicalStateEvent.
3. Client is considered as NOT supporting Lock Physical State Changed Event if the following conditions are met:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is NOT able to receive **tns1:Door/State/LockPhysicalState** notification (SystemComponentState_DoorStateLockPhysicalStateNotification feature).

5.16 Double Lock Physical State Changed Event Test Cases

5.16.1 Feature Level Requirement:

Validated Feature: Double Lock Physical State Changed Event (DoubleLockPhysicalStateChangedEvent)

Check Condition based on Device Features: DoubleLockPhysicalState event is supported by Device.

Required Number of Devices: 3

Profile D Requirement: Mandatory

5.16.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation to get notifications about the physical state of double lock.

2. Client is considered as supporting Double Lock Physical State Changed Event if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is able to receive **tns1:Door/State/DoubleLockPhysicalState** notification (SystemComponentState_DoorStateDoubleLockPhysicalStateNotification feature) if Device supports DoubleLockPhysicalStateEvent.
3. Client is considered as NOT supporting Double Lock Physical State Changed Event if the following conditions are met:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is NOT able to receive **tns1:Door/State/DoubleLockPhysicalState** notification (SystemComponentState_DoorStateDoubleLockPhysicalStateNotification feature).

5.17 Door Alarm State Changed Event Test Cases

5.17.1 Feature Level Requirement:

Validated Feature: Door Alarm State Changed Event (DoorAlarmStateChangedEvent)

Check Condition based on Device Features: DoorAlarm event is supported by Device.

Required Number of Devices: 3

Profile D Requirement: Mandatory

5.17.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation to get notifications about the door alarm state.
2. Client is considered as supporting Door Alarm State Changed Event if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is able to receive **tns1:Door/State/DoorAlarm** notification (SystemComponentState_DoorStateDoorAlarmNotification feature) if Device supports DoorAlarmEvent.

3. Client is considered as NOT supporting Door Alarm State Changed Event if the following conditions are met:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is NOT able to receive **tns1:Door/State/DoorAlarm** notification (SystemComponentState_DoorStateDoorAlarmNotification feature).

5.18 Door Tamper State Changed Event Test Cases

5.18.1 Feature Level Requirement:

Validated Feature: Door Tamper State Changed Event (DoorTamperStateChangedEvent)

Check Condition based on Device Features: DoorTamper event is supported by Device.

Required Number of Devices: 3

Profile D Requirement: Mandatory

5.18.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation to get notifications about the door tamper state.
2. Client is considered as supporting Door Tamper State Changed Event if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is able to receive **tns1:Door/State/DoorTamper** notification (SystemComponentState_DoorStateDoorTamperNotification feature) if Device supports DoorTamperEvent.
3. Client is considered as NOT supporting Door Tamper State Changed Event if the following conditions are met:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is NOT able to receive **tns1:Door/State/DoorTamper** notification (SystemComponentState_DoorStateDoorTamperNotification feature).

5.19 Door Fault State Changed Event Test Cases

5.19.1 Feature Level Requirement:

Validated Feature: Door Fault State Changed Event (DoorFaultStateChangedEvent)

Check Condition based on Device Features: DoorFault event is supported by Device.

Required Number of Devices: 3

Profile D Requirement: Mandatory

5.19.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation to get notifications about the door fault state.
2. Client is considered as supporting Door Fault State Changed Event if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is able to receive **tns1:Door/State/DoorFault** notification (SystemComponentState_DoorStateDoorFaultNotification feature) if Device supports DoorFaultEvent.
3. Client is considered as NOT supporting Door Fault State Changed Event if the following conditions are met:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is NOT able to receive **tns1:Door/State/DoorFault** notification (SystemComponentState_DoorStateDoorFaultNotification feature).

5.20 Access Control With Anonymous Access Test Cases

5.20.1 Feature Level Requirement:

Validated Feature: Access Control With Anonymous Access (AccessControlAnonymous)

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

Required Number of Devices: 3

Profile D Requirement: Mandatory

5.20.2 Expected Scenarios Under Test:

1. Client subscribes to Device messages to retrieve notification events from Device using **CreatePullPointSubscription** operation and **PullMessage** operation.
2. Client receives authorization request from Device with **tns1:AccessControl/Request/Anonymous** notification.
3. Client makes a decision about granting or denying anonymous access by **ExternalAuthorization** operation.
4. Client receives notifications about access decisions related to External Authorization with **tns1:AccessControl/AccessGranted/Anonymous** notification or **tns1:AccessControl/Denied/Anonymous**.
5. Client receives notifications about request timeout with **tns1:AccessControl/Request/Timeout** notification.
6. Client is considered as supporting Access Control With Anonymous Access if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client supports ExternalAuthorization_SendAuthDecision feature (please see EXTERNALAUTHORIZATION-2 SEND AUTHORIZATION DECISION section) AND
   - Client is able to retrieve **tns1:AccessControl/Request/Anonymous** 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/Request/Timeout** notification.
7. Client is considered as NOT supporting Access Control With Anonymous Access 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 ExternalAuthorization_SendAuthDecision feature (please see EXTERNALAUTHORIZATION-2 SEND AUTHORIZATION DECISION section) OR
- Client unable to retrieve **tns1:AccessControl/Request/Anonymous** 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/Request/Timeout** notification.

5.21 Access Control With Identifier Request (Credential) Test Cases

5.21.1 Feature Level Requirement:

Validated Feature: Access Control With Identifier Request (Credential) (AccessControlIdentifierCredential)

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

Required Number of Devices: 3

Profile D Requirement: Mandatory

5.21.2 Expected Scenarios Under Test:

1. Client subscribes to Device messages to retrieve notification events from Device using **CreatePullPointSubscription** operation and **PullMessage** operation.
2. Client receives authorization request from Device with **tns1:AccessControl/Request/Identifier** notification.
3. Client makes a decision about granting or denying Identifier access by **ExternalAuthorization** operation with **CredentialToken** in request.
4. Client receives notifications about access decisions related to External Authorization with **tns1:AccessControl/AccessGranted/Credential** notification or **tns1:AccessControl/Denied/Credential**.
5. Client receives notifications about request timeout with **tns1:AccessControl/Request/Timeout** notification.

6. Client is considered as supporting Access Control With Identifier Request (Credential) if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is able to send authorization decision to Device using **ExternalAuthorization** operation with **CredentialToken** field AND
   - Client is able to retrieve **tns1:AccessControl/Request/Identifier** notification 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/Request/Timeout** notification.
7. Client is considered as NOT supporting Access Control With Identifier Request (Credential) if ANY of the following is TRUE:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) OR
   - Client unable to send authorization decision to Device using **ExternalAuthorization** operation with **CredentialToken** field OR
   - Client unable to retrieve **tns1:AccessControl/Request/Identifier** notification 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/Request/Timeout** notification.

5.21.3 EXTERNAL AUTHORIZATION WITH CREDENTIAL

Test Case ID: ACCESSCONTROLIDENTIFIERCREDENTIAL-1

Feature Under Test: Send Authorization Desicion (AccessControlIdentifierCredential_ExternalAuthorizationWithCredential)

Profile D Normative Reference: Mandatory

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

Pre-Requisite:

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

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

1. Client sends ExternalAuthorization message to Device with Granted or Denied decision with CredentialToken.
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:
  - [S0] **soapenv:Body** element has child **tac:ExternalAuthorization** element AND
  - [S1] **tac:ExternalAuthorization** contains **tac:CredentialToken** element AND
- Device response on the **ExternalAuthorization** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tac:ExternalAuthorizationResponse**.

FAIL -

- The Client failed PASS criteria.

5.22 Access Control With Identifier Request (Anonymous) Test Cases

5.22.1 Feature Level Requirement:

Validated Feature: Access Control With Identifier Request (Anonymous) (AccessControlIdentifierAnonymous)

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

Required Number of Devices: 3

Profile D Requirement: Mandatory

5.22.2 Expected Scenarios Under Test:

1. Client subscribes to Device messages to retrieve notification events from Device using **CreatePullPointSubscription** operation and **PullMessage** operation.
2. Client receives authorization request from Device with **tns1:AccessControl/Request/Identifier** notification.
3. Client makes a decision about granting or denying Identifier access by **ExternalAuthorization** operation without **CredentialToken** in request.
4. Client receives notifications about access decisions related to External Authorization with **tns1:AccessControl/AccessGranted/Anonymous** notification or **tns1:AccessControl/Denied/Anonymous**.
5. Client receives notifications about request timeout with **tns1:AccessControl/Request/Timeout** notification.
6. Client is considered as supporting Access Control With Identifier Request (Anonymous) if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is able to send authorization decision to Device using **ExternalAuthorization** operation without **CredentialToken** field AND
   - Client is able to retrieve **tns1:AccessControl/Request/Identifier** 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/Request/Timeout** notification.
7. Client is considered as NOT supporting Access Control With Identifier Request (Anonymous) if ANY of the following is TRUE:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) OR
   - Client unable to send authorization decision to Device using **ExternalAuthorization** operation without **CredentialToken** field OR

- Client unable to retrieve **tns1:AccessControl/Request/Identifier** 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/Request/Timeout** notification.

5.22.3 EXTERNAL AUTHORIZATION TO ANONYMOUS

Test Case ID: ACCESSCONTROLIDENTIFIERANONYMOUS-1

Feature Under Test: Send Authorization Desicion (AccessControlIdentifierAnonymous_ExternalAuthorizationToAnonymous)

Profile D Normative Reference: Mandatory

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

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with ExternalAuthorization operation without CredentialToken field in request present.

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

1. Client sends ExternalAuthorization message to Device with Granted or Denied decision without CredentialToken.
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:
  - [S0] **soapenv:Body** element has child **tac:ExternalAuthorization** element AND
  - [S1] **tac:ExternalAuthorization** does not contain **tac:CredentialToken** element AND
- Device response on the **ExternalAuthorization** request fulfills the following requirements:

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

FAIL -

- The Client failed PASS criteria.

5.23 Feedback Test Cases

5.23.1 Feature Level Requirement:

Validated Feature: Feedback (Feedback)

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

Required Number of Devices: 3

Profile D Requirement: Mandatory

5.23.2 Expected Scenarios Under Test:

1. Client instructs the access point about progress of ExternalAuthorization operation using **Feedback** operation.
2. Client is considered as supporting Feedback if the following conditions are met:
   - Client is able to send feedback using **Feedback** operation.
3. Client is considered as NOT supporting Feedback if ANY of the following is TRUE:
   - No valid Device response to **Feedback** request.

5.23.3 FEEDBACK OPERATION

Test Label: Feedback Operation

Test Case ID: FEEDBACK-1

Feature Under Test: Feedback (Feedback_FeedbackRequest)

Profile D Normative Reference: Mandatory

Test Purpose: To verify that Client is able to send feedback indications using the Feedback operation.

Pre-Requisite:

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

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

1. Client invokes **Feedback** request message to instruct the access point about progress of ExternalAuthorization operation.
2. Device responds with code HTTP 200 OK and **FeedbackResponse** message.

Test Result:

PASS -

- Client **Feedback** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **Feedback** request in Test Procedure fulfills the following requirements:
  - [S0] **soapenv:Body** element has child **tac:Feedback** element AND
  - [S1] If **tac:Feedback/tac:FeedbackType** = "pt:RequireIdentifier", then **tac:Feedback** contains at least one **tac:RecognitionType** element AND
- Device response on the **Feedback** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tac:FeedbackResponse**.

FAIL -

- The Client failed PASS criteria.

5.24 Access Taken With Anonymous Access Test Cases

5.24.1 Feature Level Requirement:

Validated Feature: Access Taken With Anonymous Access (AccessTakenAnonymous)

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

Required Number of Devices: 3

Profile D Requirement: Mandatory

5.24.2 Expected Scenarios Under Test:

1. Client subscribes to Device messages to retrieve notification events from Device using **CreatePullPointSubscription** operation and **PullMessage** operation.
2. Client receives access taken notification from Device with **tns1:AccessControl/AccessTaken/Anonymous** notification.
3. Client receives access not taken notification from Device with **tns1:AccessControl/AccessNotTaken/Anonymous** notification.
4. Client is considered as supporting Access Control With Anonymous Access if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is able to retrieve **tns1:AccessControl/AccessTaken/Anonymous** notification AND
   - Client is able to retrieve **tns1:AccessControl/AccessNotTaken/Anonymous** notification AND
5. Client is considered as NOT supporting Access Control With Anonymous Access if ANY of the following is TRUE:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) OR
   - Client unable to retrieve **tns1:AccessControl/AccessTaken/Anonymous** notification OR
   - Client unable to retrieve **tns1:AccessControl/AccessNotTaken/Anonymous** notification.

5.25 Access Taken With Identifier Access Test Cases

5.25.1 Feature Level Requirement:

Validated Feature: Access Taken With Identifier Access (AccessTakenIdentifier)

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

Required Number of Devices: 3

Profile D Requirement: Mandatory

5.25.2 Expected Scenarios Under Test:

1. Client subscribes to Device messages to retrieve notification events from Device using **CreatePullPointSubscription** operation and **PullMessage** operation.
2. Client receives access taken notification from Device with **tns1:AccessControl/AccessTaken/Identifier** notification.
3. Client receives access not taken notification from Device with **tns1:AccessControl/AccessNotTaken/Identifier** notification.
4. Client is considered as supporting Access Taken With Identifier Access if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is able to retrieve **tns1:AccessControl/AccessTaken/Identifier** notification AND
   - Client is able to retrieve **tns1:AccessControl/AccessNotTaken/Identifier** notification AND
5. Client is considered as NOT supporting Access Taken With Identifier Access if ANY of the following is TRUE:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) OR
   - Client unable to retrieve **tns1:AccessControl/AccessTaken/Identifier** notification OR
   - Client unable to retrieve **tns1:AccessControl/AccessNotTaken/Identifier** notification.

5.26 Door Information Test Cases

5.26.1 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

5.26.2 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**.

5.26.3 LISTING OF DOORS

Test Label: System Component Information - Listing of Doors

Test Case ID: DOORINFORMATION-1

Feature Under Test: Listing of Doors (DoorInformation_ListingOfDoors)

Profile C Normative Reference: Mandatory

Profile D Normative Reference: Mandatory

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.

5.27 Door Information - Configuration Change Notifications Test Cases

5.27.1 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

5.27.2 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.

5.28 Door Control Test Cases

5.28.1 Feature Level Requirement:

Validated Feature: Door Control (DoorControlProfileD)

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 D Requirement: Mandatory

5.28.2 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:

- Client supports DoorControl_AccessDoor feature (please see DOORCONTROL-1 ACCESS DOOR section) AND
- Client supports DoorControl_LockDoor feature (please see DOORCONTROL-2 LOCK DOOR section) AND
- Client supports DoorControl_UnlockDoor feature (please see DOORCONTROL-3 UNLOCK DOOR section).

3. Client is considered as NOT supporting Door Control if ANY of the following is TRUE:
   - Client does not support DoorControl_AccessDoor feature (please see DOORCONTROL-1 ACCESS DOOR section) OR
   - Client does not support DoorControl_LockDoor feature (please see DOORCONTROL-2 LOCK DOOR section) OR
   - Client does not support DoorControl_UnlockDoor feature (please see DOORCONTROL-3 UNLOCK DOOR section).

6 Test Cases for Profile Conditional Features

6.1 Digest Authentication for RTSP (Profile D) Test Cases

6.1.1 Feature Level Requirement:

Validated Feature: Digest Authentication for RTSP (DigestForRTSPProfileD)

Check Condition based on Device Features: Profile D

Required Number of Devices: 1

Profile D Requirement: Conditional

6.1.2 Expected Scenarios Under Test:

1. Client invokes a specific RTSP command which is under testing without any user credentials (no UsernameToken, no HTTP Digest authentication header).
2. IF Device returns RTSP 401 Unauthorized error along with WWW-Authentication: Digest header, then Client resends RTSP command with WWW-Authenticate header.
3. Client is considered as supporting Digest Authentication for RTSP (Profile D) if the following conditions are met:
   - Client supports HTTPDigestForRTSP_HTTPDigestForRTSPTest feature (please see HTTPDIGESTFORRTSP-1 HTTP DIGEST AUTHENTICATION FOR RTSP section)
4. Client is considered as NOT supporting Digest Authentication for RTSP (Profile D) if the following is TRUE:
   - Client does not support HTTPDigestForRTSP_HTTPDigestForRTSPTest feature (please see HTTPDIGESTFORRTSP-1 HTTP DIGEST AUTHENTICATION FOR RTSP section)

6.2 System Test Cases

6.2.1 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 Q Requirement: Conditional

Profile S Requirement: Conditional

Profile T Requirement: Conditional

Profile D Requirement: Conditional

Profile M Requirement: Conditional

6.2.2 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.

6.2.3 GET DEVICE INFORMATION

Test Label: System - Get Device Information

Test Case ID: SYSTEM-1

Feature Under Test: Get Device Information (System_GetDeviceInformation)

Profile S Normative Reference: Conditional

Profile G Normative Reference: Conditional

Profile C Normative Reference: Conditional

Profile Q Normative Reference: Conditional

Profile A Normative Reference: Conditional

Profile T Normative Reference: Conditional

Profile D Normative Reference: Conditional

Profile M Normative Reference: Conditional

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.

6.3 User Handling Test Cases

6.3.1 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 Q Requirement: Mandatory

Profile S Requirement: Conditional

Profile C Requirement: Conditional

Profile G Requirement: Conditional

Profile T Requirement: Conditional

Profile D Requirement: Conditional

6.3.2 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**).

6.3.3 CREATE USERS

Test Label: User Handling - CreateUsers

Test Case ID: USERHANDLING-1

Feature Under Test: Create Users (UserHandling_CreateUsers)

Profile S Normative Reference: Conditional

Profile G Normative Reference: Conditional

Profile C Normative Reference: Conditional

Profile Q Normative Reference: Mandatory

Profile A Normative Reference: Mandatory

Profile T Normative Reference: Conditional

Profile D Normative Reference: Conditional

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.

6.3.4 GET USERS

Test Label: User Handling - GetUsers

Test Case ID: USERHANDLING-2

Feature Under Test: Get Users (UserHandling_GetUsers)

Profile S Normative Reference: Conditional

Profile G Normative Reference: Conditional

Profile C Normative Reference: Conditional

Profile Q Normative Reference: Mandatory

Profile A Normative Reference: Mandatory

Profile T Normative Reference: Conditional

Profile D Normative Reference: Conditional

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.

6.3.5 SET USER

Test Label: User Handling - SetUser

Test Case ID: USERHANDLING-3

Feature Under Test: Set User (UserHandling_SetUser)

Profile S Normative Reference: Conditional

Profile G Normative Reference: Conditional

Profile C Normative Reference: Conditional

Profile Q Normative Reference: Mandatory

Profile A Normative Reference: Mandatory

Profile T Normative Reference: Conditional

Profile D Normative Reference: Conditional

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.

6.3.6 DELETE USERS

Test Label: User Handling - DeleteUsers

Test Case ID: USERHANDLING-4

Feature Under Test: Delete Users (UserHandling_DeleteUsers)

Profile S Normative Reference: Conditional

Profile G Normative Reference: Conditional

Profile C Normative Reference: Conditional

Profile Q Normative Reference: Mandatory

Profile A Normative Reference: Mandatory

Profile T Normative Reference: Conditional

Profile D Normative Reference: Conditional

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.

6.4 Access Point Management Test Cases

6.4.1 Feature Level Requirement:

Validated Feature: Access Point Management (AccessPointManagement)

Check Condition based on Device Features: Access Point Management is supported by Device.

Required Number of Devices: 1

Profile D Requirement: Conditional

6.4.2 Expected Scenarios Under Test:

1. Client connects to Device to create, modify and delete an access point.
2. Client is considered as supporting Access Point Management if the following conditions are met:
   - Client is able to create an access point using the **CreateAccessPoint** operation AND
   - Client is able to modify an access point using the **ModifyAccessPoint** operation AND
   - Client is able to delete an access point using the **DeleteAccessPoint** operation.
3. Client is considered as NOT supporting Access Point Management if ANY of the following is TRUE:
   - No valid responses for **CreateAccessPoint** request OR
   - No valid responses for **ModifyAccessPoint** request OR
   - No valid responses for **DeleteAccessPoint** request.

6.4.3 CREATE ACCESS POINT

Test Label: Create Access Point

Test Case ID: ACCESSPOINTMANAGEMENT-1

Feature Under Test: Create Access Point (AccessPointManagement_CreateAccessPoint)

Profile D Normative Reference: Conditional

Test Purpose: To verify that Client is able to create the specified access point in the device using the **CreateAccessPoint** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **CreateAccessPoint** operation.
- Device supports Access Point Management.

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

1. Client invokes **CreateAccessPoint** request message with empty **@token** attribute to create access point on the Device.

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

Test Result:

PASS -

- Client **CreateAccessPoint** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **CreateAccessPoint** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child **tac:CreateAccessPoint** element AND
  - [S2] **tac:AccessPoint/@token** attribute is empty (has empty string value) AND
- Device response on the **CreateAccessPoint** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tac:CreateAccessPointResponse** AND

FAIL -

- The Client failed PASS criteria.

6.4.4 MODIFY ACCESS POINT

Test Label: Modify Access Point

Test Case ID: ACCESSPOINTMANAGEMENT-2

Feature Under Test: Modify Access Point (AccessPointManagement_ModifyAccessPoint)

Profile D Normative Reference: Conditional

Test Purpose: To verify that Client is able to modify the specified access point using the **ModifyAccessPoint** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **ModifyAccessPoint** operation.
- Device supports Access Point Management.

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

1. Client invokes **ModifyAccessPoint** request message to modify access point on the Device.

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.4.5 DELETE ACCESS POINT

Test Label: Delete Access Point

Test Case ID: ACCESSPOINTMANAGEMENT-3

Feature Under Test: Delete Access Point (AccessPointManagement_DeleteAccessPoint)

Profile D Normative Reference: Conditional

Test Purpose: To verify that Client is able to delete the specified access point using the **DeleteAccessPoint** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **DeleteAccessPoint** operation.
- Device supports Access Point Management.

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

1. Client invokes **DeleteAccessPoint** request message to delete access point on the Device.
2. Device responds with code HTTP 200 OK and **DeleteAccessPointResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.5 Access Points Control Test Cases

6.5.1 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

6.5.2 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.

6.5.3 DISABLE ENABLE ACCESS POINT

Test Label: Access Points Control - DisableEnableAccessPoint

Test Case ID: ACCESSPOINTCONTROL-1

Feature Under Test: Disable Enable Access Point (AccessPointControl_DisableEnableAccessPoint)

Profile C Normative Reference: Conditional

Profile D Normative Reference: Conditional

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.

6.6 Door Management Test Cases

6.6.1 Feature Level Requirement:

Validated Feature: Door Management (DoorManagement)

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

Required Number of Devices: 1

Profile D Requirement: Conditional

6.6.2 Expected Scenarios Under Test:

1. Client connects to Device to create, modify and delete a door.
2. Client is considered as supporting Door Management if the following conditions are met:
   - Client is able to create a door using the **CreateDoor** operation AND

- Client is able to modify a door using the **ModifyDoor** operation AND
- Client is able to delete a door using the **DeleteDoor** operation.

3. Client is considered as NOT supporting Door Management if ANY of the following is TRUE:
   - No valid responses for **CreateDoor** request OR
   - No valid responses for **ModifyDoor** request OR
   - No valid responses for **DeleteDoor** request.

6.6.3 CREATE DOOR

Test Label: Create Door

Test Case ID: DOORMANAGEMENT-1

Feature Under Test: Create Door (DoorManagement_CreateDoor)

Profile D Normative Reference: Conditional

Test Purpose: To verify that Client is able to create the specified door in the device using the **CreateDoor** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **CreateDoor** operation.
- Device supports Door Management.

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

1. Client invokes **CreateDoor** request message with empty **@token** attribute to create door on the Device.
2. Device responds with code HTTP 200 OK and **CreateDoorResponse** message.

Test Result:

PASS -

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

- [S1] **soapenv:Body** element has child **tdc:CreateDoor** element AND
- [S2] **tdc:Door/@token** attribute is empty (has empty string value) AND

- Device response on the **CreateDoor** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tdc:CreateDoorResponse** AND

FAIL -

- The Client failed PASS criteria.

6.6.4 MODIFY DOOR

Test Label: Modify Door

Test Case ID: DOORMANAGEMENT-2

Feature Under Test: Modify Door (DoorManagement_ModifyDoor)

Profile D Normative Reference: Conditional

Test Purpose: To verify that Client is able to modify the specified door using the **ModifyDoor** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **ModifyDoor** operation.
- Device supports Door Management.

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

1. Client invokes **ModifyDoor** request message to modify door on the Device.
2. Device responds with code HTTP 200 OK and **ModifyDoorResponse** message.

Test Result:

PASS -

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

- Client **ModifyDoor** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child **tdc:ModifyDoor** element AND
- Device response on the **ModifyDoor** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tdc:ModifyDoorResponse** AND

FAIL -

- The Client failed PASS criteria.

6.6.5 DELETE DOOR

Test Label: Delete Door

Test Case ID: DOORMANAGEMENT-3

Feature Under Test: Delete Door (DoorManagement_DeleteDoor)

Profile D Normative Reference: Conditional

Test Purpose: To verify that Client is able to delete the specified door using the **DeleteDoor** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **DeleteDoor** operation.
- Device supports Door Management.

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

1. Client invokes **DeleteDoor** request message to delete door on the Device.
2. Device responds with code HTTP 200 OK and **DeleteDoorResponse** message.

Test Result:

PASS -

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

- [S1] **soapenv:Body** element has child **tdc:DeleteDoor** element AND

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

  - [S2] It has HTTP 200 response code AND

  - [S3] **soapenv:Body** element has child element **tdc:DeleteDoorResponse** AND

FAIL -

- The Client failed PASS criteria.

6.7 Credential Format Types Test Cases

6.7.1 Feature Level Requirement:

Validated Feature: Credential Format Types (CredentialFormatTypes)

Check Condition based on Device Features: Whitelist or Blacklist is supported by Device.

Required Number of Devices: 1

Profile D Requirement: Conditional

6.7.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve supported credential format types from a device.

2. Client is considered as supporting Credential Format Types if the following conditions are met:

   - Client supports ConfigureCredentials_GetSupportedFormatTypes feature (please see CONFIGURECREDENTIALS-1 GET SUPPORTED FORMAT TYPES section).

   - Client supports GetServicesWithCapabilities feature (please see Get Services with Capabilities section).

3. Client is considered as NOT supporting Credential Format Types if ANY of the following is TRUE:

   - Client does not support ConfigureCredentials_GetSupportedFormatTypes feature (please see CONFIGURECREDENTIALS-1 GET SUPPORTED FORMAT TYPES section).

- Client does not support GetServicesWithCapabilities feature (please see Get Services with Capabilities section).

6.8 Credential Whitelisting Test Cases

6.8.1 Feature Level Normative Reference:

Validated Feature: Credential Whitelisting (CredentialWhitelisting)

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

Required Number of Devices: 1

Profile D Requirement: Conditional

6.8.2 Expected Scenarios Under Test:

1. Client manages whitelists on a device using **GetWhitelist**, **AddToWhitelist**, **RemoveFromWhitelist**, and **DeleteWhitelist** operations.
2. Client is considered as supporting Credential Whitelisting if the following conditions are met:
   - Client is able to retrieve whitelisted credential identifiers using **GetWhitelist** operation AND
   - Client is able to add the specified credential identifiers to the whitelist using **AddToWhitelist** operation AND
   - Client is able to remove the specified credential identifiers from the whitelist using **RemoveFromWhitelist** operation AND
   - Client is able to delete all credential identifiers from the whitelist using **DeleteWhitelist** operation AND
   - Client is able to retrieve **tns1:AccessControl/AccessGranted/Identifier** notification.
   - Client supports CredentialFormatTypes feature (please see Credential Format Types section).
3. Client is considered as NOT supporting Credential Whitelisting if ANY of the following is TRUE:
   - No valid responses for **GetWhitelist** request OR

- No valid responses for **AddToWhitelist** request OR
- No valid responses for **RemoveFromWhitelist** request OR
- No valid responses for **DeleteWhitelist** request OR
- Client is not able to retrieve **tns1:AccessControl/AccessGranted/Identifier** notification.
- Client does not support CredentialFormatTypes feature (please see Credential Format Types section).

6.8.3 GET WHITELIST

Test Label: Get Whitelist

Test Case ID: CREDENTIALWHITELISTING-1

Feature Under Test: Get Whitelist (CredentialWhitelisting_GetWhitelist)

Profile D Normative Reference: Conditional

Test Purpose: To verify that Client is able to retrieve whitelisted credential identifiers using **GetWhitelist** operation

Pre-Requisite:

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

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

1. Client invokes **GetWhitelist** request messages to get full list of whitelisted credential identifiers from a device.
2. Device responds with code HTTP 200 OK and **GetWhitelistResponse** message.

Test Result:

PASS -

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

- [S1] **soapenv:Body** element has child element **tcr:GetWhitelist** AND

- Device response on the **GetWhitelist** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tcr:GetWhitelistResponse**.
  - [S4] **tcr:GetWhitelistResponse** does not contain **tcr:NextStartReference** element.

FAIL -

- The Client failed PASS criteria.

6.8.4 ADD TO WHITELIST

Test Label: Add to Whitelist

Test Case ID: CREDENTIALWHITELISTING-2

Feature Under Test: Add to Whitelist (CredentialWhitelisting_AddToWhitelist)

Profile D Normative Reference: Conditional

Test Purpose: To verify that Client is able to add the specified credential identifiers to the whitelist using **AddToWhitelist** operation

Pre-Requisite:

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

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

1. Client invokes **AddToWhitelist** request messages to add the specified credential identifiers to the whitelist on a device.
2. Device responds with code HTTP 200 OK and **AddToWhitelistResponse** message.

Test Result:

PASS -

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

- Client **AddToWhitelist** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tcr:AddToWhitelist** AND
- Device response on the **AddToWhitelist** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tcr:AddToWhitelistResponse**.

FAIL -

- The Client failed PASS criteria.

6.8.5 REMOVE FROM WHITELIST

Test Label: Remove from Whitelist

Test Case ID: CREDENTIALWHITELISTING-3

Feature Under Test: Remove from Whitelist (CredentialWhitelisting_RemoveFromWhitelist)

Profile D Normative Reference: Conditional

Test Purpose: To verify that Client is able to remove the specified credential identifiers from the whitelist using **RemoveFromWhitelist** operation

Pre-Requisite:

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

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

1. Client invokes **RemoveFromWhitelist** request messages to remove the specified credential identifiers from the whitelist on a device.
2. Device responds with code HTTP 200 OK and **RemoveFromWhitelistResponse** message.

Test Result:

PASS -

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

- Client **RemoveFromWhitelist** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tcr:RemoveFromWhitelist** AND
- Device response on the **RemoveFromWhitelist** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tcr:RemoveFromWhitelistResponse**.

FAIL -

- The Client failed PASS criteria.

6.8.6 DELETE WHITELIST

Test Label: Delete Whitelist

Test Case ID: CREDENTIALWHITELISTING-4

Feature Under Test: Delete Whitelist (CredentialWhitelisting_DeleteWhitelist)

Profile D Normative Reference: Conditional

Test Purpose: To verify that Client is able to delete all credential identifiers from the whitelist using **DeleteWhitelist** operation

Pre-Requisite:

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

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

1. Client invokes **DeleteWhitelist** request messages to delete all credential identifiers from the whitelist on a device.
2. Device responds with code HTTP 200 OK and **DeleteWhitelistResponse** message.

Test Result:

PASS -

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

- Client **DeleteWhitelist** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tcr:DeleteWhitelist** AND
- Device response on the **DeleteWhitelist** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tcr:DeleteWhitelistResponse**.

FAIL -

- The Client failed PASS criteria.

6.9 Credential Blacklisting Test Cases

6.9.1 Feature Level Normative Reference:

Validated Feature: Credential Blacklisting (CredentialBlacklisting)

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

Required Number of Devices: 1

Profile D Requirement: Conditional

6.9.2 Expected Scenarios Under Test:

1. Client manages blacklists on a device using **GetBlacklist**, **AddToBlacklist**, **RemoveFromBlacklist**, and **DeleteBlacklist** operations.
2. Client is considered as supporting Credential Blacklisting if the following conditions are met:
   - Client is able to retrieve blacklisted credential identifiers using **GetBlacklist** operation AND
   - Client is able to add the specified credential identifiers to the blacklist using **AddToBlacklist** operation AND
   - Client is able to remove the specified credential identifiers from the blacklist using **RemoveFromBlacklist** operation AND
   - Client is able to delete all credential identifiers from the blacklist using **DeleteBlacklist** operation AND
   - Client is able to retrieve **tns1:AccessControl/Denied/Identifier** notification.

- Client supports CredentialFormatTypes feature (please see Credential Format Types section).

3. Client is considered as NOT supporting Credential Blacklisting if ANY of the following is TRUE:
   - No valid responses for **GetBlacklist** request OR
   - No valid responses for **AddToBlacklist** request OR
   - No valid responses for **RemoveFromBlacklist** request OR
   - No valid responses for **DeleteBlacklist** request OR
   - Client is not able to retrieve **tns1:AccessControl/Denied/Identifier** notification.
   - Client does not support CredentialFormatTypes feature (please see Credential Format Types section).

6.9.3 GET BLACKLIST

Test Label: Get Blacklist

Test Case ID: CREDENTIALBLACKLISTING-1

Feature Under Test: Get Blacklist (CredentialBlacklisting_GetBlacklist)

Profile D Normative Reference: Conditional

Test Purpose: To verify that Client is able to retrieve blacklisted credential identifiers using **GetBlacklist** operation

Pre-Requisite:

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

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

1. Client invokes **GetBlacklist** request messages to get full list of blacklisted credential identifiers from a device.
2. Device responds with code HTTP 200 OK and **GetBlacklistResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.9.4 ADD TO BLACKLIST

Test Label: Add to Blacklist

Test Case ID: CREDENTIALBLACKLISTING-2

Feature Under Test: Add to Blacklist (CredentialBlacklisting_AddToBlacklist)

Profile D Normative Reference: Conditional

Test Purpose: To verify that Client is able to add the specified credential identifiers to the blacklist using **AddToBlacklist** operation

Pre-Requisite:

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

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

1. Client invokes **AddToBlacklist** request messages to add the specified credential identifiers to the blacklist on a device.
2. Device responds with code HTTP 200 OK and **AddToBlacklistResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.9.5 REMOVE FROM BLACKLIST

Test Label: Remove from Blacklist

Test Case ID: CREDENTIALBLACKLISTING-3

Feature Under Test: Remove from Blacklist (CredentialBlacklisting_RemoveFromBlacklist)

Profile D Normative Reference: Conditional

Test Purpose: To verify that Client is able to remove the specified credential identifiers from the blacklist using **RemoveFromBlacklist** operation

Pre-Requisite:

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

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

1. Client invokes **RemoveFromBlacklist** request messages to remove the specified credential identifiers from the blacklist on a device.
2. Device responds with code HTTP 200 OK and **RemoveFromBlacklistResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.9.6 DELETE BLACKLIST

Test Label: Delete Blacklist

Test Case ID: CREDENTIALBLACKLISTING-4

Feature Under Test: Delete Blacklist (CredentialBlacklisting_DeleteBlacklist)

Profile D Normative Reference: Conditional

Test Purpose: To verify that Client is able to delete all credential identifiers from the blacklist using **DeleteBlacklist** operation

Pre-Requisite:

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

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

1. Client invokes **DeleteBlacklist** request messages to delete all credential identifiers from the blacklist on a device.
2. Device responds with code HTTP 200 OK and **DeleteBlacklistResponse** message.

Test Result:

PASS -

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

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

Profile Q 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)

Profile A Normative Reference: Optional

Profile C Normative Reference: Optional

Profile G Normative Reference: Optional

Profile Q Normative Reference: Optional

Profile D Normative Reference: Optional

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.

8 Supplementary Features and Test Cases

8.1 GET SERVICES

Test Label: Capabilities - Determine the available Services

Test Case ID: CAPABILITIES-1

Feature Under Test: Get Services (Capabilities_GetServicesRequest)

Profile S Normative Reference: Mandatory

Profile G Normative Reference: Mandatory

Profile C Normative Reference: Mandatory

Profile Q Normative Reference: Mandatory

Profile A Normative Reference: Mandatory

Profile T Normative Reference: Mandatory

Profile M Normative Reference: Mandatory

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.

8.2 SEND AUTHORIZATION DECISION

Test Label: External Authorization - Send Authorization Decision

Test Case ID: EXTERNALAUTHORIZATION-2

Feature Under Test: Send Authorization Desicion (ExternalAuthorization_SendAuthDecision)

Profile C Normative Reference: Conditional

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.

8.3 ACCESS DOOR

Test Label: Door Control - AccessDoor

Test Case ID: DOORCONTROL-1

Feature Under Test: Access Door (DoorControl_AccessDoor)

Profile C Normative Reference: Mandatory

Profile D Normative Reference: Mandatory

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.

8.4 LOCK DOOR

Test Label: Door Control - LockDoor

Test Case ID: DOORCONTROL-2

Feature Under Test: Lock Door (DoorControl_LockDoor)

Profile C Normative Reference: Mandatory

Profile D Normative Reference: Mandatory

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.

8.5 UNLOCK DOOR

Test Label: Door Control - UnlockDoor

Test Case ID: DOORCONTROL-3

Feature Under Test: Unlock Door (DoorControl_UnlockDoor)

Profile C Normative Reference: Mandatory

Profile D Normative Reference: Mandatory

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.

8.6 METADATA STREAMING USING MEDIA2

Test Label: Metadata Streaming Using Media2

Test Case ID: MEDIA2_METADATASTREAMING-1

Feature Under Test: Metadata Streaming (Media2_MetadataStreaming_MetadataStreamingUsingMedia2)

Profile T Normative Reference: Conditional

Profile M Normative Reference: Mandatory

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.

8.7 GET SUPPORTED FORMAT TYPES

Test Label: Configure Credentials - Get Supported Format Types

Test Case ID: CONFIGURECREDENTIALS-1

Feature Under Test: Get Supported Format Types (ConfigureCredentials_GetSupportedFormatTypes)

Profile A Normative Reference: Mandatory

Profile D Normative Reference: Conditional

Test Purpose: To verify that Client is able to get supported format types from Device for specified identifier type using the **GetSupportedFormatTypes** operation.

Pre-Requisite:

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

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

1. Client invokes **GetSupportedFormatTypes** request message to get supported format types from Device for specified identifier type.
2. Device responds with code HTTP 200 OK and **GetSupportedFormatTypesResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

8.8 HTTP DIGEST AUTHENTICATION FOR RTSP

Test Label: HTTP Digest For RTSP

Test Case ID: HTTPDIGESTFORRTSP-1

Feature Under Test: HTTP Digest For RTSP (HTTPDigestForRTSP_HTTPDigestForRTSPTest)

Profile A Normative Reference: None

Profile C Normative Reference: None

Profile G Normative Reference: None

Profile Q Normative Reference: None

Profile T Normative Reference: Mandatory

Profile M Normative Reference: Mandatory

Profile D Normative Reference: Conditional

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

Pre-Requisite:

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

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

1. Client sends a RTSP request that requires authentication (e.g. DESCRIBE) to the Device without any authentication.
2. Device rejects the request with a RTSP 401 status code, AND a WWW-Authenticate Response Header.
3. Client re-sends the RTSP request with a Authorization Request Header.
4. Device accepts the correct request with RTSP 200 OK status code.

Test Result:

PASS -

- There is Client RTSP request in Test Procedure that does not contain any authentication AND
- Device response on the Client RTSP request fulfills the following requirements:
  - It has RTSP 401 status code AND
  - WWW-Authenticate Response Header contains challenge = "Digest" element AND

- WW-Authenticate Response Header contains "realm=*" element AND
- WW-Authenticate Response Header contains "nonce=*" element AND

- There is Client RTSP request in Test Procedure that fulfills the following requirements
  - WW-Authenticate Request Header credentials = "Digest" element AND
  - WW-Authenticate Request Header contains "realm=*" element with value from Device response AND
  - WW-Authenticate Request Header contains "nonce=*" element with value from Device response AND
  - WW-Authenticate Request Header contains "uri=*" element AND
- Device responds with code RTSP 200 OK.

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.GetServices	Get Services	3	GetServices is supported by Device.	GetServices
tc.Discovery	Discovery	3	None	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.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.SetSynchronizationPoint	Set Synchronization Point (Event Service)	1	Pull Point Notification OR WS-Basic Notification is supported by Device.	no UnsupportedPullPointNotification OR WSBasicNotification
tc.AccessPointInformation	Access Point Information	3	Access Control Service is	AccessControlService

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.AccessPointConfigurationChangeNotifications	Access Point Information - Configuration Change Notifications	3	Access Control Service is supported by Device.	AccessControlService
tc.GetAccessPointState	Get Access Point State	3	AccessPoint entity is supported by Device.	AccessPointEntity
tc.AccessPointStateChangedEvent	Access Point State Changed Event	3	AccessPointStateEnabled event is supported by Device.	AccessPointStateEnabledEvent
tc.GetDoorState	Get Door State	3	Door entity is supported by Device.	DoorEntity
tc.DoorModeStateChangedEvent	Door Mode State Changed Event	3	DoorMode event is supported by Device.	DoorModeEvent
tc.DoorPhysicalStateChangedEvent	Door Physical State Changed Event	3	DoorPhysical State event is supported by Device.	DoorPhysical StateEvent
tc.LockPhysicalStateChangedEvent	Lock Physical State Changed Event	3	LockPhysicalState event is supported by Device.	LockPhysical StateEvent
tc.DoubleLockPhysicalStateChangedEvent	Double Lock Physical State Changed Event	3	DoubleLockPhysicalState event is supported by Device.	DoubleLockPhysicalStateEvent
tc.DoorAlarmStateChangedEvent	Door Alarm State Changed Event	3	DoorAlarm event is supported by Device.	DoorAlarmEvent

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
tc.DoorTamperStateChangedEvent	Door Tamper State Changed Event	3	DoorTamper event is supported by Device.	DoorTamperEvent
tc.DoorFaultStateChangedEvent	Door Fault State Changed Event	3	DoorFault event is supported by Device.	DoorFaultEvent
tc.AccessControlAnonymous	Access Control With Anonymous Access	3	Access Control Service is supported by Device. Anonymous Access is supported by Device.	ExternalAuthorization AND AnonymousAccess
tc.AccessControlIdentifierCredential	Access Control With Identifier Request (Credential)	3	Access Control Service is supported by Device. Identifier Access is supported by Device. Access Point Entity is supported by Device.	ExternalAuthorization AND IdentifierAccess AND AccessPointEntity
tc.AccessControlIdentifierAnonymous	Access Control With Identifier Request (Anonymous)	3	Access Control Service is supported by Device. Identifier Access is supported by Device. Access Point Entity is supported by Device.	ExternalAuthorization AND IdentifierAccess AND AccessPointEntity
tc.Feedback	Feedback	3	Feedback is supported by Device.	Feedback

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
tc.AccessTakenAnonymous	Access Taken With Anonymous Access	3	Access Control Service is supported by Device. Anonymous Access is supported by Device. Access Taken notifications is supported by Device.	AccessTaken AND AnonymousAccess
tc.AccessTakenIdentifier	Access Taken With Identifier Access	3	Access Control Service is supported by Device. Identifier Access is supported by Device. Access Taken notifications is supported by Device.	AccessTaken AND IdentifierAccess
tc.DoorInformation	Door Information	3	Door Control Service is supported by Device.	DoorControlService
tc.DoorConfigurationChangeNotifications	Door Information - Configuration Change Notifications	3	Door Control Service is supported by Device.	DoorControlService
tc.DoorControlProfileD	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

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
tc.DigestForRTSPProfileD	Digest Authentication for RTSP (Profile D)	1	Profile D	ProfileDSupported
tc.System	System	3	None	All
tc.UserHandling	User Handling	3	User Configuration	no UserConfigNotSupported
tc.AccessPointManagement	Access Point Management	1	Access Point Management is supported by Device.	AccessPointManagement
tc.AccessPointControl	Access Points Control	1	Enable/Disable Access Point is supported by Device.	EnableDisableAccessPoint
tc.DoorManagement	Door Management	1	Door Management is supported by Device.	DoorManagement
tc.CredentialFormatTypes	Credential Format Types	1	Whitelist or Blacklist is supported by Device.	Whitelist OR Blacklist
tc.CredentialWhitelisting	Credential Whitelisting	1	Whitelist is supported by Device.	Whitelist
tc.CredentialBlacklisting	Credential Blacklisting	1	Blacklist is supported by Device.	Blacklist
tc.GetServicesWithCapabilities	Get Services with Capabilities	1	GetServices is supported by Device.	GetServices