ONVIF®

Profile Q Client Test Specification

Version 19.12

December 2019

REVISION HISTORY

Vers.	Date	Description
19.12	Sep 18, 2019	The following was done according to #325: Scope\Supplementary Features and Test Cases sections was added. Supplementary Features and Test Cases sections was added.
19.12	Aug 13, 2019	The following was done according to #325: EVENTHANDLING-3 METADATA STREAMING test was removed from Event Handling Feature and moved to Metadata Streaming Using Media2. Test case ID was changed to MEDIA2_METADATASTREAMING-1. Event Handling will use link to this test. EVENTHANDLING-4 METADATA STREAMING USING MEDIA was added for Profile S Devices.
19.12	Sep 6, 2019	DEVICEDISCOVERYTYPEFILTER-1 DEVICE DISCOVERY TYPE FILTER was updated according to #323: Unnecessary step with check that ProbeMatch is sent to Client IP address was removed.
19.12	Aug 14, 2019	The following was done according to #341: HTTP Digest section and HTTP Digest Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Capabilities section and Capabilities Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Get Services section and Get Services Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Discovery section and Discovery Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Device Discovery Type Filter section and Device Discovery Type Filter Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: User Handling section and User Handling Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.

19.12	Aug 14, 2019	The following was done according to #341: Event Handling section and Event Handling Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Network Configuration section and Network Configuration Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: System section and System Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: NTP section and NTP Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Zero Configuration section and Zero Configuration Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: System Date and Time Configuration section and System Date and Time Configuration Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: HTTP Firmware Upgrade section and HTTP Firmware Upgrade Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: HTTP System Backup section and HTTP System Backup Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: HTTP System Restore section and HTTP System Restore Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Device Management Notifications section and Device Management Notifications Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Monitoring Notifications section and Monitoring Notifications Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.

19.12	Aug 14, 2019	The following was done according to #341: Hostname Configuration section and Hostname Configuration Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: DNS Configuration section and DNS Configuration Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Network Protocols Configuration section and Network Protocols Configuration Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: TLS Configuration section and TLS Configuration Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Get Services with Capabilities section and Get Services with Capabilities Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Set Synchronization Point section and Set Synchronization Point Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Unsubscribe section and Unsubscribe Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.12	Aug 14, 2019	The following was done according to #341: Keep Alive for Pull Point Event Handling section and Keep Alive for Pull Point Event Handling Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile Q Client Test Specifications.
19.06	Jun 14, 2019	The following was done according to #309: 'Validated Feature' section for each feature updated to be synchronized with feature ID used in feature list. 'Feature Under Test' section for each test case updated to be synchronized with sub-feature ID used in feature list. 'Validated Feature List' test case section removed.
18.06	Jun 21, 2018	Reformatting document using new template
18.06	Apr 05, 2018	'Required Number of Devices Summary' Annex added according to #241
18.06	Feb 16, 2018	The following were updated in the scope of #241:

		Feature Level Requirement (updated with new rules) Each Feature Level Requirement (updated with Check Condition based on Device Features and Required Number of Devices)
17.06	Jun 15, 2017	Links in Normative references section were updated.
16.07	Mar 14, 2016	www.onvif.org was removed from Copyright section.
16.01	Nov 23, 2015	General item (Test Owerview) was added Minor updates in formatting, typos and terms.
16.01	Sep 25, 2015	Initial version: General parts added Transition to Operational State Test Cases added

Table of Contents

1 Introduction

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

This particular document defines test cases required for testing Profile Q features of a Client application e.g. Transition to Operational State. 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 Q Client Test Specification defines and regulates the conformance testing procedure for the ONVIF conformant Clients in the scope of Profile Q 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 Q features according to ONVIF Profile Specifications.

The principal intended purposes are:

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

1.2.1 HTTP Digest

HTTP Digest section defines security mechanism for HTTP Digest Authentication.

1.2.2 Capabilities

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

1.2.3 Get Services

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

1.2.4 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.

1.2.5 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 is equal to tds:Device or with skipped Types filter.

1.2.6 User Handling

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

1.2.7 Transition to Operational State

Transition to Operational State section specifies Client ability to transit an ONVIF Device from Factory Default State into Operational State.

1.3 Test Cases for Profile Conditional Features

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

1.3.1 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.

1.3.2 Network Configuration

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

1.3.3 System

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

1.3.4 NTP

NTP section defines Client ability to configure synchronization of time using NTP servers on Device.

1.3.5 Zero Configuration

Zero Configuration section defines Client ability to enable or disable zero configuration on Device.

1.3.6 System Date and Time Configuration

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

1.3.7 HTTP Firmware Upgrade

HTTP Firmware Upgrade section defines Client ability to upgrade Device firmware over HTTP using StartFirmwareUpgrad operation and HTTP POST.

1.3.8 HTTP System Backup

HTTP System Backup section defines Client ability to backup system configurations over HTTP using GetSystemUris operation and HTTP GET.

1.3.9 HTTP System Restore

HTTP System Restore section defines Client ability to restore system configurations over HTTP using StartSystemRestore operation and HTTP POST.

1.3.10 Monitoring Notifications

Monitoring Notifications section specifies Client ability to receive from Device monitoring notifications.

1.3.11 Device Management Notifications

Device Management Notifications section specifies Client ability to receive from Device device management notifications.

1.3.12 Hostname Configuration

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

1.3.13 DNS Configuration

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

1.3.14 Network Protocols Configuration

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

1.3.15 TLS Configuration

TLS Configuration section specifies Client ability to manage the associations between certification paths and the TLS server on Device.

1.4 Test Cases for Profile Optional Features

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

1.4.1 Get Services with Capabilities

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

1.4.2 Set Synchronization Point

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

1.4.3 Unsubscribe

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

1.4.4 Keep Alive for Pull Point Event Handling

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

1.5 Supplementary Features and Test Cases

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

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

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 Q	The Profile Q Specification.
Factory Default State	The state of the Profile Q device prior to setting an Administrator password. In this state, the device accepts any commands without authentication.
Operational State	The state of the Profile Q device after setting an Administrator password. The device requires an authentication according to the ONVIF default access policy to accept commands.

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.
WS-I BP 2.0	Web Services Interoperability Basic Profile version 2.0.

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].
tas	http://www.onvif.org/ver10/advancedsecurity/wsdl	The namespace for the WSDL advanced security 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 Q is an ONVIF Client that can transit an ONVIF Device conformant to Profile Q into Operational State.

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 Q, 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 Capabilities Test Cases

5.2.1 Feature Level Requirement:

Validated Feature: Capabilities (Capabilities)

Check Condition based on Device Features: None

Required Number of Devices: 3

Profile A Requirement: Mandatory

Profile C Requirement: Mandatory

Profile G Requirement: Mandatory

Profile Q Requirement: Mandatory

Profile S Requirement: Mandatory

Profile T Requirement: Mandatory

5.2.2 Expected Scenarios Under Test:

1. Client invokes a specific Capabilities command which is under testing.
2. Client is considered as supporting Capabilities if the following conditions are met:
   - Device returns a valid response to GetServices request OR
   - Device returns a valid response to GetCapabilities request.
3. Client is considered as NOT supporting Capabilities if the following is TRUE:
   - No Valid Device Response to GetServices request AND
   - No Valid Device Response to GetCapabilities request.

5.2.3 GET SERVICES

Test Label: Capabilities - Determine the available Services

Test Case ID: CAPABILITIES-1

Feature Under Test: Get Services (Capabilities_GetServicesRequest)

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.

5.2.4 GET CAPABILITIES

Test Label: Capabilities - Get Device Capabilities

Test Case ID: CAPABILITIES-2

Feature Under Test: Get Capabilities (Capabilities_GetCapabilities)

Profile S Normative Reference: Mandatory

Profile G Normative Reference: Optional

Profile C Normative Reference: Optional

Profile T Normative Reference: None

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

5.3 Get Services Test Cases

5.3.1 Feature Level Requirement:

Validated Feature: Get Services (GetServices)

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

Required Number of Devices: 3

Profile A Requirement: Mandatory

Profile D Requirement: Mandatory

Profile C Requirement: Mandatory

Profile G Requirement: Mandatory

Profile Q Requirement: Mandatory

Profile T Requirement: Mandatory

Profile M Requirement: Mandatory

5.3.2 Expected Scenarios Under Test:

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

5.4 Discovery Test Cases

5.4.1 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

5.4.2 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.

5.4.3 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.

5.5 Device Discovery Type Filter Test Cases

5.5.1 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

5.5.2 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 is equal to **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 is equal to 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.

5.5.3 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** = 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 has 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.

5.6 User Handling Test Cases

5.6.1 Feature Level Requirement:

Validated Feature: User Handling (UserHandling)

Check Condition based on Device Features: None

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

5.6.2 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**).

5.6.3 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.

5.6.4 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.

5.6.5 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.

5.6.6 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.

5.7 Transition to Operational State Test Cases

5.7.1 Feature Level Requirement:

Validated Feature: Transition to Operational State (TransitionToOperationalState)

Check Condition based on Device Features: Profile Q is supported by Device.

Required Number of Devices: 3

Profile Q Requirement: Mandatory

5.7.2 Expected Scenarios Under Test:

1. A Client connects to a Device in Factory Default State to invoke its transition into Operational State.
2. The Client is considered as supporting Transition to Operational State if the following conditions are met:
   - The Client is able to invoke the Device transition into the Operational State by using EITHER **CreateUsers** OR **SetUser** operations.
3. The Client is considered as NOT supporting Transition to Operational State if ANY of the following is TRUE:
   - No valid response to **CreateUsers** request OR
   - No valid response to **SetUser** request AND
   - **SetUser** request does not contain user with **Username** value contained in **GetUsers** response.

5.7.3 TRANSITION TO OPERATIONAL STATE BY CREATEUSERS

Test Label: Transition to Operational State by Create User

Test Case ID: TRANSITIONTOOPERATIONALSTATE-1

Feature Under Test: Transition to Operational State by CreateUsers (TransitionToOperationalState_TransitionToOperationalStateByCreateUsers)

Profile Q Normative Reference: Mandatory

Test Purpose: To verify that a Client is able to invoke Device transition into Operational State using the **CreateUsers**.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device in Factory Default state with **CreateUsers** operation without any authentication which contains User with "Administrator" user level present.

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

1. Client invokes **CreateUsers** request message without any authentication and with non-empty password to create a new admin user.
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] **soapenv:Body** element has child element **tds:CreateUsers** AND
  - [S2] It does not contain Digest Authentication part AND
  - [S3] It does not contain WS-Username Token Authentication part AND
  - It contains **tds:User** element which fulfills the following requirements:
    - [S4] **tt:Username** element has non-empty string value AND

- [S5] It contains **tt:Password** element AND
- [S6] **tt:Password** element has non-empty string value AND
- [S7] **tt:UserLevel** element value equals "Administrator" AND

- Device response to the **CreateUsers** request fulfills the following requirements:
  - [S8] It has HTTP 200 response code AND
  - [S9] **soapenv:Body** element has child element **tds:CreateUsersResponse**

FAIL -

- The Client failed PASS criteria.

5.7.4 TRANSITION TO OPERATIONAL STATE BY SET USER

Test Label: Transition to Operational State by Set User

Test Case ID: TRANSITIONTOOPERATIONALSTATE-2

Feature Under Test: Transition to Operational State by SetUser (TransitionToOperationalState_TransitionToOperationalStateBySetUser)

Profile Q Normative Reference: Mandatory

Test Purpose: To verify that a Client is able to invoke Device transition into Operational State using the **SetUser**.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device in Factory Default state with **SetUser** operation without any authentication and with UserLevel is equal to "Administrator" present.

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

1. Client invokes **GetUsers** request message without any authentication to retrieve user list from Device.
2. Device responds with code HTTP 200 OK and **GetUsersResponse** message.
3. Client invokes **SetUser** request message without any authentication to modify the password of an existing admin user.
4. 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] **soapenv:Body** element has child element **tds:SetUser** AND
  - [S2] It does not contain Digest Authentication part AND
  - [S3] It does not contain WS-Username Token Authentication part AND
  - It contains **tds:User** element which fulfills the following requirements:
    - [S4] **tt:Username** element has non-empty string value AND
    - [S5] It contains **tt:Password** element AND
    - [S6] **tt:Password** element has non-empty string value AND
    - [S7] **tt:UserLevel** element value equals "Administrator" AND
- Device response to the **SetUser** request fulfills the following requirements:
  - [S8] It has HTTP 200 response code AND
  - [S9] **soapenv:Body** element has child element **tds:SetUserResponse**
- There is a Client **GetUsers** request message in Test Procedure fulfills the following requirements:
  - [S10] It is invoked for the same Device as the response for the **SetUser** request AND
  - [S11] It is invoked before the Client **SetUser** request AND
  - [S12] It does not contain digest authentication part AND
  - [S13] It does not contain WS-username token authentication part AND
- Device response to the **GetUsers** request fulfills the following requirements:
  - [S14] It has HTTP 200 response code AND
  - [S15] **soapenv:Body** element has child element **tds:GetUsersResponse**
  - [S16] It contains **tt:User** element which fulfills the following requirements:

- [S17] **tt:Username** element value equals to **tt:Username** value from the **SetUser** request AND
- [S18] **UserLevel** element value equals "Administrator".

FAIL -

- The Client failed PASS criteria.

6 Test Cases for Profile Conditional Features

6.1 Event Handling Test Cases

6.1.1 Feature Level Requirement:

Validated Feature: Event Handling (EventHandling)

Check Condition based on Device Features: None

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

6.1.2 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.

6.1.3 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

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.

6.1.4 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.

6.1.5 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.

6.2 Network Configuration Test Cases

6.2.1 Feature Level Requirement:

Validated Feature: Network Configuration (NetworkConfiguration)

Check Condition based on Device Features: None

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

6.2.2 Expected Scenarios Under Test:

1. Client connects to Device to configure network settings.
2. Client is considered as supporting Network Configuration if the following conditions are met:
   - Client is able to list network interfaces of Device using the GetNetworkInterfaces operation AND
   - Client is able to set network interfaces of Device using the SetNetworkInterfaces operation AND
   - Client is able to list default gateway of Device using the GetNetworkDefaultGateway operation AND
   - Client is able set default gateway of Device using the SetNetworkDefaultGateway operation.
3. Client is considered as NOT supporting Network Configuration if ANY of the following is TRUE:
   - No Valid Device Response to GetNetworkInterfaces request OR
   - No Valid Device Response to SetNetworkInterfaces request OR
   - No Valid Device Response to GetNetworkDefaultGateway request OR
   - No Valid Device Response to SetNetworkDefaultGateway request.

6.2.3 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.

6.2.4 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.

6.2.5 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.

6.2.6 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.

6.3 System Test Cases

6.3.1 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

6.3.2 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.

6.3.3 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

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.

6.4 NTP Test Cases

6.4.1 Feature Level Requirement:

Validated Feature: NTP (NTP)

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

Required Number of Devices: 1

Profile S Requirement: Conditional

Profile Q Requirement: Conditional

Profile T Requirement: Conditional

6.4.2 Expected Scenarios Under Test:

1. Client connects to Device to configure synchronization of time using NTP servers on Device.
2. Client is considered as supporting NTP if the following conditions are met:
   - Client is able to get the NTP settings from Device using the GetNTP operation AND
   - Client is able to set the NTP settings on Device using the SetNTP operation.
3. Client is considered as NOT supporting NTP if ANY of the following is TRUE:
   - No Valid Device Response to GetNTP request OR
   - No Valid Device Response to SetNTP request.

6.4.3 GET NTP

Test Label: NTP - GetNTP

Test Case ID: NTP-1

Feature Under Test: Get NTP (NTP_GetNTP)

Profile S Normative Reference: Conditional

Profile Q Normative Reference: Conditional

Profile T Normative Reference: Conditional

Test Purpose: To verify that Client is able to get the NTP settings from Device using the GetNTP operation.

Pre-Requisite:

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

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

1. Client invokes GetNTP request message to get current settings of NTP servers on Device.
2. Device responds with code HTTP 200 OK and GetNTPResponse message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.4.4 SET NTP

Test Label: NTP - SetNTP

Test Case ID: NTP-2

Feature Under Test: Set NTP (NTP_SetNTP)

Profile S Normative Reference: Conditional

Profile Q Normative Reference: Conditional

Profile T Normative Reference: Conditional

Test Purpose: To verify that Client is able to set the NTP settings on Device using the SetNTP operation.

Pre-Requisite:

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

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

1. Client invokes SetNTP request message to set the NTP servers settings on Device.

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.5 Zero Configuration Test Cases

6.5.1 Feature Level Requirement:

Validated Feature: Zero Configuration (ZeroConfiguration)

Check Condition based on Device Features: Zero Configuration is supported by Device.

Required Number of Devices: 1

Profile S Requirement: Conditional

Profile Q Requirement: Conditional

6.5.2 Expected Scenarios Under Test:

1. Client connects to Device to configure Zero Configuration settings.
2. Client is considered as supporting Zero Configuration if the following conditions are met:
   - Client is able to get the Zero Configuration settings from Device using the GetZeroConfiguration operation AND
   - Client is able to set the Zero Configuration settings on Device using the SetZeroConfiguration operation.

3. Client is considered as NOT supporting Zero Configuration if ANY of the following is TRUE:
    - No Valid Device Response to GetZeroConfiguration request OR
    - No Valid Device Response to SetZeroConfiguration request.

6.5.3 GET ZERO CONFIGURATION

Test Label: Zero Configuration - GetZeroConfiguration

Test Case ID: ZEROCONFIGURATION-1

Feature Under Test: Get Zero Configuration (ZeroConfiguration_GetZeroConfiguration)

Profile S Normative Reference: Conditional

Profile Q Normative Reference: Conditional

Test Purpose: To verify that Client is able to get the Zero Configuration settings from Device using the GetZeroConfiguration operation.

Pre-Requisite:

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

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

1. Client invokes GetZeroConfiguration request message to get the Zero Configuration settings from Device.
2. Device responds with code HTTP 200 OK and GetZeroConfigurationResponse message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.5.4 SET ZERO CONFIGURATION

Test Label: Zero Configuration - SetZeroConfiguration

Test Case ID: ZEROCONFIGURATION-2

Feature Under Test: Set Zero Configuration (ZeroConfiguration_SetZeroConfiguration)

Profile S Normative Reference: Conditional

Profile Q Normative Reference: Conditional

Test Purpose: To verify that Client is able to set the Zero Configuration settings on Device using the SetZeroConfiguration operation.

Pre-Requisite:

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

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

1. Client invokes SetZeroConfiguration request message to set the Zero Configuration settings on Device.
2. Device responds with code HTTP 200 OK and SetZeroConfigurationResponse message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.6 System Date and Time Configuration Test Cases

6.6.1 Feature Level Requirement:

Validated Feature: System Date and Time Configuration (SystemDateAndTimeConfiguration)

Check Condition based on Device Features: None

Required Number of Devices: 1

Profile A Requirement: Conditional

Profile C Requirement: Optional

Profile G Requirement: Optional

Profile Q Requirement: Conditional

Profile S Requirement: Optional

6.6.2 Expected Scenarios Under Test:

1. Client connects to Device to configure system date and time.
2. Client is considered as supporting System Date and Time Configuration if the following conditions are met:
   - Client is able to retrieve a system date and time using **GetSystemDateAndTime** operation AND
   - Client is able to configure a system date and time using EITHER **SetSystemDateAndTime** operation OR **SetNTP** operation.
3. Client is considered as NOT supporting System Date and Time Configuration if ANY of the following is TRUE:
   - No valid responses for **GetSystemDateAndTime** request OR
   - No valid responses for **SetSystemDateAndTime** request if detected AND
   - Client does not support NTP feature.

6.6.3 GET SYSTEM DATE AND TIME

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

Test Case ID: SYSTEMDATEANDTIMECONFIGURATION-1

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

Profile A Normative Reference: Conditional

Profile C Normative Reference: Optional

Profile G Normative Reference: Optional

Profile Q Normative Reference: Conditional

Profile S Normative Reference: Optional

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

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

FAIL -

- The Client failed PASS criteria.

6.6.4 SET SYSTEM DATE AND TIME

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

Test Case ID: SYSTEMDATEANDTIMECONFIGURATION-2

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

Profile A Normative Reference: Conditional

Profile C Normative Reference: Optional

Profile G Normative Reference: Optional

Profile Q Normative Reference: Conditional

Profile S Normative Reference: Optional

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

- Client **SetSystemDateAndTime** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetSystemDateAndTime** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tds:SetSystemDateAndTime** AND
  - [S2] If **tds:DateTimeType** element value is equal to "Manual" THEN **tds:SetSystemDateAndTime** contains **tds:UTCDateTime** element AND
- Device response on the **SetSystemDateAndTime** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tds:SetSystemDateAndTimeResponse**.

FAIL -

- The Client failed PASS criteria.

6.7 HTTP Firmware Upgrade Test Cases

6.7.1 Feature Level Requirement:

Validated Feature: Firmware Upgrade via HTTP (HTTPFirmwareUpgrade)

Check Condition based on Device Features: HTTP Firmware Upgrade is supported by Device.

Required Number of Devices: 1

Profile Q Requirement: Conditional

6.7.2 Expected Scenarios Under Test:

1. Client connects to the Device to instruct it to prepare for upgrade using the StartFirmwareUpgrade operation.
2. Client sends the firmware image using HTTP POST to the upload URI provided by the Device in StartFirmwareUpgradeResponse.
3. Client is considered as supporting HTTP Firmware Upgrade if the following conditions are met:
   - Client is able to instruct the Device to prepare for upgrade using **StartFirmwareUpgrade** operation if Device supports HTTP Firmware Upgrade AND

- Client is able to send the firmware image using **HTTP POST** if Device supports HTTP Firmware Upgrade.

4. Client is considered as NOT supporting HTTP Firmware Upgrade if ANY of the following is TRUE:
   - No valid responses for **StartFirmwareUpgrade** request if Device supports HTTP Firmware Upgrade OR
   - No valid **HTTP POST** request to the upload URI if Device supports HTTP Firmware Upgrade.
   - No valid responses for **HTTP POST** request to the upload URI with firmware image if Device supports HTTP Firmware Upgrade.

6.7.3 FIRMWARE UPGRADE VIA HTTP

Test Label: Firmware Upgrade via HTTP - Start Firmware Upgrade

Test Case ID: HTTPFIRMWAREUPGRADE-1

Feature Under Test: Start Firmware Upgrade (HTTPFirmwareUpgrade_StartFirmwareUpgrade)

Profile Q Normative Reference: Conditional

Test Purpose: To verify that Client is able to upgrade the Device firmware via HTTP using the **StartFirmwareUpgrade** operation and **HTTP POST**.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **StartFirmwareUpgrade** operation present.
- Device supports Http Firmware Upgrade.

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

1. Client invokes **StartFirmwareUpgrade** request message to instruct the Device to prepare for upgrade.
2. Device responds with code HTTP 200 OK and **StartFirmwareUpgradeResponse** message.
3. Client sends the firmware image using **HTTP POST** to the upload URI provided by the Device in StartFirmwareUpgradeResponse.

4. Device responds with code HTTP 200 OK message.

Test Result:

PASS -

- Client **StartFirmwareUpgrade** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **StartFirmwareUpgrade** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tds:StartFirmwareUpgrade** AND
- Device response on the **StartFirmwareUpgrade** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tds:StartFirmwareUpgradeResponse**.
- There is **HTTP POST** request in Test Procedure fulfills the following requirements:
  - [S4] It invoked to address which equal to **tds:StartFirmwareUpgradeResponse/ tds:UploadUri** value from the Device response to **StartFirmwareUpgrade** request AND
  - [S5] It invoked after the Client **StartFirmwareUpgrade** request AND
  - [S6] It contains HTTP Content-Type Header with value is equal to “application/octet-stream” AND
- Device response on the **HTTP POST** request fulfills the following requirements:
  - [S7] It has HTTP 200 response code.

FAIL -

- The Client failed PASS criteria.

6.8 HTTP System Backup Test Cases

6.8.1 Feature Level Requirement:

Validated Feature: System Backup via HTTP (HTTPSystemBackup)

Check Condition based on Device Features: HTTP System Backup is supported by Device.

Required Number of Devices: 1

Profile Q Requirement: Conditional

6.8.2 Expected Scenarios Under Test:

1. Client connects to the Device to retrieve URI from which a system backup may be downloaded using the GetSystemUris operation.

   Client gets the backup system configurations using HTTP GET sent to the System Backup Uri provided by the Device in GetSystemUrisResponse.

2. Client is considered as supporting HTTP System Backup if the following conditions are met:

   - Client is able to retrieve URI from Device for system backup using **GetSystemUris** operation if Device supports HTTP System Backup AND
   - Client is able to to backup system configurations using **HTTP GET** if Device supports HTTP System Backup AND

3. Client is considered as NOT supporting HTTP System Backup if ANY of the following is TRUE:

   - No valid responses for **GetSystemUris** request if Device supports HTTP System Backup OR
   - No valid responses for **HTTP GET** request to the System Backup Uri if Device supports HTTP System Backup.

6.8.3 GET SYSTEM URIS

Test Label: System Backup via HTTP - Get System Uris

Test Case ID: HTTPSYSTEMBACKUP-1

Feature Under Test: Get System Uris (HTTPSystemBackup_GetSystemUris)

Profile Q Normative Reference: Conditional

Test Purpose: To verify that Client is able to backup system configurations via HTTP using the **GetSystemUris** operation and **HTTP GET**.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetSystemUris** operation present.
- Device supports HTTP System Backup.

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

1. Client invokes **GetSystemUris** request message to retrieve URI from which a system backup file may be downloaded.
2. Device responds with code HTTP 200 OK and **GetSystemUrisResponse** message.
3. Client retrieves the backup file using **HTTP GET** to the System Backup Uri provided by the Device in GetSystemUrisResponse.
4. Device responds with code HTTP 200 OK message and with backup file.

Test Result:

PASS -

- Client **GetSystemUris** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetSystemUris** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tds:GetSystemUris** AND
- Device response on the **GetSystemUris** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tds:GetSystemUrisResponse**.
- There is **HTTP GET** request in Test Procedure that fulfills the following requirements:
  - [S4] It invoked to address which equal to **tds:GetSystemUrisResponse/ tds:SystemBackupUri** value from the Device response to **GetSystemUris** request AND
  - [S5] It invoked after the Client **GetSystemUris** request AND
- Device response on the **HTTP GET** request fulfills the following requirements:
  - [S6] It has HTTP 200 response code.

FAIL -

- The Client failed PASS criteria.

6.9 HTTP System Restore Test Cases

6.9.1 Feature Level Requirement:

Validated Feature: System Restore via HTTP (HTTPSystemRestore)

Check Condition based on Device Features: HTTP System Backup is supported by Device.

Required Number of Devices: 1

Profile Q Requirement: Conditional

6.9.2 Expected Scenarios Under Test:

1. Client connects to the Device to retrieve URI to which the backuped data may be uploaded using the StartSystemRestore operation.

   Client uploads the backuped configuration data using HTTP POST to the Upload Uri provided by the Device in StartSystemRestoreResponse.

2. Client is considered as supporting HTTP System Restore if the following conditions are met:

   - Client is able to retrieve URI from Device for restore system configurations using **StartSystemRestore** operation if Device supports HTTP System Backup AND
   - Client is able to send the backuped data to the Device using **HTTP POST** if Device supports HTTP System Backup.

3. Client is considered as NOT supporting HTTP System Restore if ANY of the following is TRUE:

   - No valid responses for **StartSystemRestore** request if Device supports HTTP System Backup OR
   - No valid **HTTP POST** request to the Upload Uri if Device supports HTTP System Backup.
   - No valid responses for **HTTP POST** request to the Upload Uri if Device supports HTTP System Backup.

6.9.3 HTTP SYSTEM RESTORE

Test Label: System Restore via HTTP - Start System Restore

Test Case ID: HTTPSYSTEMRESTORE-1

Feature Under Test: Start System Restore (HTTPSystemRestore_StartSystemRestore)

Profile Q Normative Reference: Conditional

Test Purpose: To verify that Client is able to restore system configurations via HTTP using the **StartSystemRestore** operation and **HTTP POST**.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **StartSystemRestore** operation present.
- Device supports HTTP System Backup.

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

1. Client invokes **StartSystemRestore** request message to retrieve upload URI from the Device.
2. Device responds with code HTTP 200 OK and **StartSystemRestoreResponse** message.
3. Client transmits the configuration data to the upload URI using **HTTP POST**.
4. Device responds with code HTTP 200 OK message.

Test Result:

PASS -

- Client **StartSystemRestore** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **StartSystemRestore** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tds:StartSystemRestore** AND
- Device response on the **StartSystemRestore** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tds:StartSystemRestoreResponse**.
- There is **HTTP POST** request in Test Procedure that fulfills the following requirements:
  - [S4] It invoked to address which equal to **tds:StartSystemRestore/tds:UploadUri** value from the Device response to **StartSystemRestore** request AND
  - [S5] It invoked after the Client **StartSystemRestore** request AND
  - [S6] It contains HTTP Content-Type Header with value is equal to “application/octet-stream” AND
- Device response on the **HTTP POST** request fulfills the following requirements:
  - [S7] It has HTTP 200 response code.

FAIL -

- The Client failed PASS criteria.

6.10 Monitoring Notifications Test Cases

6.10.1 Feature Level Requirement:

Validated Feature: Monitoring Notifications (MonitoringNotifications)

Check Condition based on Device Features: Monitoring/ProcessorUsage or Monitoring/OperatingTime/LastReset or Monitoring/OperatingTime/LastReboot or Monitoring/OperatingTime/LastClockSynchronization is supported by Device.

Required Number of Devices: 1

Profile Q Requirement: Conditional

6.10.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation to get monitoring notifications.
2. Client uses Pull Point event mechanism to retrieve notification events from Device.
3. Client is considered as supporting Monitoring Notifications if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature AND
   - Client is able to retrieve at least one of the following notifications:
     - tns1:Monitoring/ProcessorUsage notification about processor usage if Device supports MonitoringProcessorUsageEvent feature
     - tns1:Monitoring/OperatingTime/LastReset notification about last reset if Device supports MonitoringOperatingTimeLastResetEvent feature
     - tns1:Monitoring/OperatingTime/LastReboot notification about last reboot if Device supports MonitoringOperatingTimeLastRebootEvent feature
     - tns1:Monitoring/OperatingTime/LastClockSynchronization notification about last clock synchronization if Device supports MonitoringOperatingTimeLastClockSynchronizationEvent feature
4. Client is considered as NOT supporting Monitoring Notifications if ANY of the following is TRUE:

- Client does not support EventHandling_Pullpoint feature OR
- Client is not able to retrieve the following notifications:
  - tns1:Monitoring/ProcessorUsage notification about processor usage if Device supports MonitoringProcessorUsageEvent feature
  - tns1:Monitoring/OperatingTime/LastReset notification about last reset if Device supports MonitoringOperatingTimeLastResetEvent feature
  - tns1:Monitoring/OperatingTime/LastReboot notification about last reboot if Device supports MonitoringOperatingTimeLastRebootEvent feature
  - tns1:Monitoring/OperatingTime/LastClockSynchronization notifications about last clock synchronization if Device supports MonitoringOperatingTimeLastClockSynchronizationEvent feature.

6.11 Device Management Notifications Test Cases

6.11.1 Feature Level Requirement:

Validated Feature: Device Management Notifications (DeviceManagementNotifications)

Check Condition based on Device Features: Device/HardwareFailure/FanFailure or Device/HardwareFailure/PowerSupplyFailure or Device/HardwareFailure/StorageFailure or Device/HardwareFailure/TemperatureCritical or Monitoring/Backup/Last is supported by Device.

Required Number of Devices: 1

Profile Q Requirement: Conditional

6.11.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation to get device management notifications.
2. Client uses Pull Point event mechanism to retrieve notification events from Device.
3. Client is considered as supporting Device Management Notifications if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature AND
   - Client is able to retrieve at least one of the following notifications:

- tns1:Device/HardwareFailure/FanFailure notification about fan failure if Device supports DeviceHardwareFailureFanFailureEvent feature
- tns1:Device/HardwareFailure/PowerSupplyFailure notification about power supply failure if Device supports DeviceHardwareFailurePowerSupplyFailureEvent feature
- tns1:Device/HardwareFailure/StorageFailure notification about storage failure if Device supports DeviceHardwareFailureStorageFailureEvent feature
- tns1:Device/HardwareFailure/TemperatureCritical notification about temperature critical if Device supports DeviceHardwareFailureTemperatureCriticalEvent feature
- tns1:Monitoring/Backup/Last notification about last backup if Device supports MonitoringBackupLastEvent feature

4. Client is considered as NOT supporting Device Management Notifications if ANY of the following is TRUE:
   - Client does not support EventHandling_Pullpoint feature OR
   - Client is not able to retrieve the following notifications:
     - tns1:Device/HardwareFailure/FanFailure notification about fan failure if Device supports DeviceHardwareFailureFanFailureEvent feature
     - tns1:Device/HardwareFailure/PowerSupplyFailure notification about power supply failure if Device supports DeviceHardwareFailurePowerSupplyFailureEvent feature
     - tns1:Device/HardwareFailure/StorageFailure notification about storage failure if Device supports DeviceHardwareFailureStorageFailureEvent feature
     - tns1:Device/HardwareFailure/TemperatureCritical notification about temperature critical if Device supports DeviceHardwareFailureTemperatureCriticalEvent feature
     - tns1:Monitoring/Backup/Last notification about last backup if Device supports MonitoringBackupLastEvent feature

6.12 Hostname Configuration Test Cases

6.12.1 Feature Level Requirement:

Validated Feature: Hostname Configuration (HostnameConfiguration)

Check Condition based on Device Features: None

Required Number of Devices: 1

Profile A Requirement: Optional

Profile C Requirement: Optional

Profile G Requirement: Optional

Profile Q Requirement: Conditional

Profile S Requirement: Optional

6.12.2 Expected Scenarios Under Test:

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

6.12.3 GET HOSTNAME

Test Label: Hostname Configuration - Get Hostname

Test Case ID: HOSTNAMECONFIGURATION-1

Feature Under Test: Get Hostname (HostnameConfiguration_GetHostname)

Profile A Normative Reference: Optional

Profile C Normative Reference: Optional

Profile G Normative Reference: Optional

Profile Q Normative Reference: Conditional

Profile S Normative Reference: Optional

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.12.4 SET HOSTNAME

Test Label: Hostname Configuration - Set Hostname

Test Case ID: HOSTNAMECONFIGURATION-2

Feature Under Test: Set Hostname (HostnameConfiguration_SetHostname)

Profile A Normative Reference: Optional

Profile C Normative Reference: Optional

Profile G Normative Reference: Optional

Profile Q Normative Reference: Conditional

Profile S Normative Reference: Optional

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.13 DNS Configuration Test Cases

6.13.1 Feature Level Requirement:

Validated Feature: DNS Configuration (DNSConfiguration)

Check Condition based on Device Features: None

Required Number of Devices: 1

Profile A Requirement: Optional

Profile C Requirement: Optional

Profile G Requirement: Optional

Profile Q Requirement: Conditional

Profile S Requirement: Optional

6.13.2 Expected Scenarios Under Test:

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

6.13.3 GET DNS

Test Label: DNS Configuration - Get DNS

Test Case ID: DNSCONFIGURATION-1

Feature Under Test: Get DNS (DNSConfiguration_GetDNS)

Profile A Normative Reference: Optional

Profile C Normative Reference: Optional

Profile G Normative Reference: Optional

Profile Q Normative Reference: Conditional

Profile S Normative Reference: Optional

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.13.4 SET DNS

Test Label: DNS Configuration - Set DNS

Test Case ID: DNSCONFIGURATION-2

Feature Under Test: Set DNS (DNSConfiguration_SetDNS)

Profile A Normative Reference: Optional

Profile C Normative Reference: Optional

Profile G Normative Reference: Optional

Profile Q Normative Reference: Conditional

Profile S Normative Reference: Optional

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.14 Network Protocols Configuration Test Cases

6.14.1 Feature Level Requirement:

Validated Feature: Network Protocols Configuration (NetworkProtocolsConfiguration)

Check Condition based on Device Features: None

Required Number of Devices: 1

Profile A Requirement: Optional

Profile C Requirement: Optional

Profile G Requirement: Optional

Profile Q Requirement: Conditional

Profile S Requirement: Optional

6.14.2 Expected Scenarios Under Test:

1. Client connects to Device to configure a network protocols.
2. Client is considered as supporting Network Protocols Configuration if the following conditions are met:
   - Client is able to get defined network protocols from the Device using **GetNetworkProtocols** operation AND
   - Client is able configures defined network protocols on the Device using **SetNetworkProtocols** operation.
3. Client is considered as NOT supporting Network Protocols Configuration if ANY of the following is TRUE:
   - No valid responses for **GetNetworkProtocols** request OR
   - No valid responses for **SetNetworkProtocols** request.

6.14.3 GET NETWORK PROTOCOLS

Test Label: Network Protocols Configuration - Get Network Protocols

Test Case ID: NETWORKPROTOCOLSCONFIGURATION-1

Feature Under Test: Get Network Protocols (NetworkProtocolsConfiguration_GetNetworkProtocols)

Profile A Normative Reference: Optional

Profile C Normative Reference: Optional

Profile G Normative Reference: Optional

Profile Q Normative Reference: Conditional

Profile S Normative Reference: Optional

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.14.4 SET NETWORK PROTOCOLS

Test Label: Network Protocols Configuration - Set Network Protocols

Test Case ID: NETWORKPROTOCOLSCONFIGURATION-2

Feature Under Test: Set Network Protocols (NetworkProtocolsConfiguration_SetNetworkProtocols)

Profile A Normative Reference: Optional

Profile C Normative Reference: Optional

Profile G Normative Reference: Optional

Profile Q Normative Reference: Conditional

Profile S Normative Reference: Optional

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.15 TLS Configuration Test Cases

6.15.1 Feature Level Requirement:

Validated Feature: TLS Configuration (TLSConfiguration)

Check Condition based on Device Features: TLS Server (Security Configuration Service) is supported by Device.

Required Number of Devices: 1

Profile A Requirement: None

Profile C Requirement: None

Profile G Requirement: None

Profile Q Requirement: Conditional

Profile S Requirement: None

6.15.2 Expected Scenarios Under Test:

1. Client connects to Device to manage the associations between certification paths and the TLS server.
2. Client is considered as supporting TLS Configuration if the following conditions are met:
   - Client may upload a passphrase from the keystore of the Device using **UploadPassphrase** operation if Device supports Passphrase handling AND
   - Client may delete a passphrase to the keystore of the Device using **DeletePassphrase** operation if Device supports Passphrase handling AND
   - Client is able to generates a DER-encoded PKCS#10 using **CreatePKCS10CSR** operation and upload created certificate using **UploadCertificate** operation if Device supports PKCS10ExternalCertificationWithRSA AND
   - Client is able to upload a certificate using **UploadCertificate** operation if Device supports PKCS10ExternalCertificationWithRSA AND
   - Client is able to delete a certificate to the keystore of the Device using **DeleteCertificate** operation if Device supports

PKCS10ExternalCertificationWithRSA or SelfSignedCertificateCreationWithRSA or PKCS12CertificateWithRSAPrivateKeyUpload AND

- Client is able to delete a certification path using **DeleteCertificationPath** operation if Device supports TLSServerSupport or PKCS12CertificateWithRSAPrivateKeyUpload AND
- Client is able to delete a key using **DeleteKey** operation if MaximumNumberOfKeys is greater than zero on Device AND
- Client is able to get key status using EITHER **GetKeyStatus** operation OR using **tns1:Advancedsecurity/Keystore/KeyStatus** event if MaximumNumberOfKeys is greater than zero on Device AND
- Client supports EventHandling_Pullpoint feature (please, see ONVIF Core Client Test Specification) when **tns1:Advancedsecurity/Keystore/KeyStatus** event is supported AND
- Client is able to upload a certification path consisting of X.509 certificates using **UploadCertificateWithPrivateKeyInPKCS12** operation if Device supports PKCS12CertificateWithRSAPrivateKeyUpload AND
- Client is able to assigns a key pair and certificate along with a certification path to the TLS server on the Device using **AddServerCertificateAssignment** operation if Device supports TLSServerSupport AND
- Client is able to remove key pair and certificate assignment to the TLS server on the Device using **RemoveServerCertificateAssignment** operation if Device supports TLSServerSupport AND
- Client is able to replace an existing key pair and certificate assignment to the TLS server on the Device by a new key pair and certificate assignment using **ReplaceServerCertificateAssignment** operation if Device supports TLSServerSupport AND
- Client is able to create certification path using **CreateCertificationPath** operation if Device supports TLSServerSupport AND
- Client is able to generate RSA key pair using **CreateRSAKeyPair** operation if Device supports RSAKeyPairGeneration AND
- Client supports network_protocols_configuration.set_network_protocols feature (see ONVIF Core Client Test Specification).

3. Client is considered as NOT supporting TLS Configuration if ANY of the following is TRUE:

- No valid responses for **UploadPassphrase** request if detected if Device supports Passphrase handling OR
- No valid responses for **DeletePassphrase** request if detected if Device supports Passphrase handling OR
- No valid responses for **CreatePKCS10CSR** request if Device supports Passphrase handling OR
- No valid responses for **UploadCertificate** request if Device supports Passphrase handling OR
- No valid responses for **DeleteCertificate** request if Device supports PKCS10ExternalCertificationWithRSA or SelfSignedCertificateCreationWithRSA or PKCS12CertificateWithRSAPrivateKeyUpload OR
- No valid responses for **DeleteCertificationPath** request if Device supports TLSServerSupport or PKCS12CertificateWithRSAPrivateKeyUpload OR
- No valid responses for **DeleteKey** request if MaximumNumberOfKeys is greater than zero on Device OR
- No valid responses for **GetKeyStatus** request if detected if MaximumNumberOfKeys is greater than zero on Device OR
- Client unable to get key status using **GetKeyStatus** request OR using **tns1:Advancedsecurity/Keystore/KeyStatus** event if MaximumNumberOfKeys is greater than zero on Device OR
- Client does not support EventHandling_Pullpoint feature (please, see ONVIF Core Client Test Specification) when Client supports **tns1:Advancedsecurity/Keystore/KeyStatus** notification if if MaximumNumberOfKeys is greater than zero on Device OR
- No valid responses for **UploadCertificateWithPrivateKeyInPKCS12** request if Device supports PKCS12CertificateWithRSAPrivateKeyUpload OR
- No valid responses for **AddServerCertificateAssignment** request if Device supports TLSServerSupport OR
- No valid responses for **RemoveServerCertificateAssignment** request if Device supports TLSServerSupport OR
- No valid responses for **ReplaceServerCertificateAssignment** request if Device supports TLSServerSupport OR

- No valid responses for **CreateCertificationPath** request if Device supports TLSServerSupport OR
- No valid responses for **CreateRSAKeyPair** request if Device supports RSAKeyPairGeneration OR
- Client does not support network_protocols_configuration.set_network_protocols feature (see ONVIF Core Client Test Specification).

6.15.3 UPLOAD PASSPHRASE

Test Label: Upload Passphrase

Test Case ID: TLSCONFIGURATION-1

Feature Under Test: Upload Passphrase (TLSConfiguration_UploadPassphrase)

Profile A Normative Reference: None

Profile C Normative Reference: None

Profile G Normative Reference: None

Profile Q Normative Reference: Optional

Profile S Normative Reference: None

Test Purpose: To verify that Client is able to upload a passphrase to the keystore of the Device using **UploadPassphrase** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **UploadPassphrase** operation present.
- Device supports Security Configuration Service.
- Device supports Passphrase handling.

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

1. Client invokes **UploadPassphrase** request message to upload a passphrase to the Device.
2. Device responds with code HTTP 200 OK and **UploadPassphraseResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.15.4 DELETE PASSPHRASE

Test Label: Delete Passphrase

Test Case ID: TLSCONFIGURATION-2

Feature Under Test: Delete Passphrase (TLSConfiguration_DeletePassphrase)

Profile A Normative Reference: None

Profile C Normative Reference: None

Profile G Normative Reference: None

Profile Q Normative Reference: Optional

Profile S Normative Reference: None

Test Purpose: To verify that Client is able to delete a passphrase from the keystore of the Device using **DeletePassphrase** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **DeletePassphrase** operation present.
- Device supports Security Configuration Service.
- Device supports Passphrase handling.

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

1. Client invokes **DeletePassphrase** request message to delete a passphrase from the Device.
2. Device responds with code HTTP 200 OK and **DeletePassphraseResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.15.5 CREATE PKCS#10 CERTIFICATION

Test Label: Create PKCS#10 Certification

Test Case ID: TLSCONFIGURATION-3

Feature Under Test: Create PKCS#10 Certification (TLSConfiguration_CreatePKCS10Certification)

Profile A Normative Reference: None

Profile C Normative Reference: None

Profile G Normative Reference: None

Profile Q Normative Reference: Conditional

Profile S Normative Reference: None

Test Purpose: To verify that Client is able to generates a DER-encoded PKCS#10 using **CreatePKCS10CSR** operation, create an X.509 certificate from a PKCS#10 certification request and upload created certificate using **UploadCertificate** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **CreatePKCS10CSR** operation present.
- Device supports Security Configuration Service.
- Device supports PKCS10ExternalCertificationWithRSA.

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

1. Client invokes **CreatePKCS10CSR** request message to generate PKCS#10 on the Device.
2. Device responds with code HTTP 200 OK and **CreatePKCS10CSRResponse** message.
3. Client creates a certificate from the PKCS#10 request with RSAkey pair and associated CA certificate and a corresponding private key
4. Client invokes **UploadCertificate** request message to upload a certificate on the Device.
5. Device responds with code HTTP 200 OK and **UploadCertificateResponse** message.

Test Result:

PASS -

- Client **CreatePKCS10CSR** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **CreatePKCS10CSR** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tas:CreatePKCS10CSR** AND
- Device response on the **CreatePKCS10CSR** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tas:CreatePKCS10CSRResponse**.
- There is Client **UploadCertificate** request in Test Procedure that fulfills the following requirements:
  - [S4] It is invoked after the Client **CreatePKCS10CSR** request AND
  - **tas:UploadCertificate/tas:Certificate** element value fulfills the following requirements:
    - [S5] It contains Subject element with value equals to Subject element value from **tas:CreatePKCS10CSRResponse/tas:PKCS10CSR** AND
    - [S6] It contains Public Key element with value equals to Public Key element value from **tas:CreatePKCS10CSRResponse/tas:PKCS10CSR** AND
- Device response to the **UploadCertificate** request fulfills the following requirements:

- [S7] It has RTSP 200 response code AND
- [S8] **soapenv:Body** element has child element **tas:UploadCertificateResponse**.

FAIL -

- The Client failed PASS criteria.

6.15.6 UPLOAD CERTIFICATE

Test Label: Upload Certificate

Test Case ID: TLSCONFIGURATION-4

Feature Under Test: Upload Certificate (TLSConfiguration_UploadCertificate)

Profile A Normative Reference: None

Profile C Normative Reference: None

Profile G Normative Reference: None

Profile Q Normative Reference: Conditional

Profile S Normative Reference: None

Test Purpose: To verify that Client is able to upload a certificate using **UploadCertificate** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **UploadCertificate** operation present.
- Device supports Security Configuration Service.
- Device supports PKCS10ExternalCertificationWithRSA.

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

1. Client invokes **UploadCertificate** request message to upload a certificate on the Device.
2. Device responds with code HTTP 200 OK and **UploadCertificateResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.15.7 DELETE CERTIFICATE

Test Label: Delete Certificate

Test Case ID: TLSCONFIGURATION-5

Feature Under Test: Delete Certificate (TLSConfiguration_DeleteCertificate)

Profile A Normative Reference: None

Profile C Normative Reference: None

Profile G Normative Reference: None

Profile Q Normative Reference: Conditional

Profile S Normative Reference: None

Test Purpose: To verify that Client is able to delete a certificate using **DeleteCertificate** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **DeleteCertificate** operation present.
- Device supports Security Configuration Service.
- Device supports PKCS10ExternalCertificationWithRSA or SelfSignedCertificateCreationWithRSA or PKCS12CertificateWithRSAPrivateKeyUpload.

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

1. Client invokes **DeleteCertificate** request message to delete a certificate from the Device.

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.15.8 DELETE CERTIFICATION PATH

Test Label: Delete Certification Path

Test Case ID: TLSCONFIGURATION-6

Feature Under Test: Delete Certification Path (TLSConfiguration_DeleteCertificationPath)

Profile A Normative Reference: None

Profile C Normative Reference: None

Profile G Normative Reference: None

Profile Q Normative Reference: Conditional

Profile S Normative Reference: None

Test Purpose: To verify that Client is able to delete a certification path using **DeleteCertificationPath** operation.

Pre-Requisite:

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

- Device supports TLSServerSupport or PKCS12CertificateWithRSAPrivateKeyUpload.

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

1. Client invokes **DeleteCertificationPath** request message to delete a certification path from the Device.
2. Device responds with code HTTP 200 OK and **DeleteCertificationPathResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.15.9 DELETE KEY

Test Label: DeleteKey

Test Case ID: TLSCONFIGURATION-7

Feature Under Test: Delete Key (TLSConfiguration_DeleteKey)

Profile A Normative Reference: None

Profile C Normative Reference: None

Profile G Normative Reference: None

Profile Q Normative Reference: Conditional

Profile S Normative Reference: None

Test Purpose: To verify that Client is able to delete a key using **DeleteKey** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **DeleteKey** operation present.
- Device supports Security Configuration Service.
- MaximumNumberOfKeys is greater than zero.

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

1. Client invokes **DeleteKey** request message to delete a key from the keystore of Device.
2. Device responds with code HTTP 200 OK and **DeleteKeyResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.15.10 GET KEY STATUS

Test Label: Get Key Status

Test Case ID: TLSCONFIGURATION-8

Feature Under Test: Get Key Status (TLSConfiguration_GetKeyStatus)

Profile A Normative Reference: None

Profile C Normative Reference: None

Profile G Normative Reference: None

Profile Q Normative Reference: Optional

Profile S Normative Reference: None

Test Purpose: To verify that Client is able to get key status using **GetKeyStatus** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetKeyStatus** operation present.
- Device supports Security Configuration Service.
- MaximumNumberOfKeys is greater than zero.

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

1. Client invokes **GetKeyStatus** request message to get a key status from the Device.
2. Device responds with code HTTP 200 OK and **GetKeyStatusResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.15.11 UPLOAD PKCS12

Test Label: Upload PKCS12

Test Case ID: TLSCONFIGURATION-9

Feature Under Test: Upload PKCS12 (TLSConfiguration_UploadPKCS12)

Profile A Normative Reference: None

Profile C Normative Reference: None

Profile G Normative Reference: None

Profile Q Normative Reference: Conditional

Profile S Normative Reference: None

Test Purpose: To verify that Client is able to upload a certification path consisting of X.509 certificates using **UploadCertificateWithPrivateKeyInPKCS12** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **UploadCertificateWithPrivateKeyInPKCS12** operation present.
- Device supports Security Configuration Service.
- Device supports PKCS12CertificateWithRSAPrivateKeyUpload.

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

1. Client invokes **UploadCertificateWithPrivateKeyInPKCS12** request message to upload a PKCS12 to the Device.
2. Device responds with code HTTP 200 OK and **UploadCertificateWithPrivateKeyInPKCS12Response** message.

Test Result:

PASS -

- Client **UploadCertificateWithPrivateKeyInPKCS12** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **UploadCertificateWithPrivateKeyInPKCS12** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tas:UploadCertificateWithPrivateKeyInPKCS12** AND
- Device response on the **UploadCertificateWithPrivateKeyInPKCS12** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tas:UploadCertificateWithPrivateKeyInPKCS12Response**.

FAIL -

- The Client failed PASS criteria.

6.15.12 ADD SERVER CERTIFICATE ASSIGNMENT

Test Label: Add Server Certificate Assignment

Test Case ID: TLSCONFIGURATION-10

Feature Under Test: Add Server Certificate Assignment (TLSConfiguration_AddServerCertificateAssignment)

Profile A Normative Reference: None

Profile C Normative Reference: None

Profile G Normative Reference: None

Profile Q Normative Reference: Conditional

Profile S Normative Reference: None

Test Purpose: To verify that Client is able to assigns a key pair and certificate along with a certification path to the TLS server on the Device using **AddServerCertificateAssignment** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **AddServerCertificateAssignment** operation present.
- Device supports Security Configuration Service.
- Device supports TLSServerSupport.

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

1. Client invokes **AddServerCertificateAssignment** request message to assign of a certificate to a TLS server.
2. Device responds with code HTTP 200 OK and **AddServerCertificateAssignmentResponse** message.

Test Result:

PASS -

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

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

FAIL -

- The Client failed PASS criteria.

6.15.13 REMOVE SERVER CERTIFICATE ASSIGNMENT

Test Label: Remove Server Certificate Assignment

Test Case ID: TLSCONFIGURATION-11

Feature Under Test: Remove Server Certificate Assignment (TLSConfiguration_RemoveServerCertificateAssignment)

Profile A Normative Reference: None

Profile C Normative Reference: None

Profile G Normative Reference: None

Profile Q Normative Reference: Conditional

Profile S Normative Reference: None

Test Purpose: To verify that Client is able to remove key pair and certificate assignment to the TLS server on the Device using **RemoveServerCertificateAssignment** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **RemoveServerCertificateAssignment** operation present.
- Device supports Security Configuration Service.
- Device supports TLSServerSupport.

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

1. Client invokes **RemoveServerCertificateAssignment** request message to remove server certification assignment.
2. Device responds with code HTTP 200 OK and **RemoveServerCertificateAssignmentResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.15.14 REPLACE SERVER CERTIFICATE ASSIGNMENT

Test Label: Replace Server Certificate Assignment

Test Case ID: TLSCONFIGURATION-12

Feature Under Test: Replace Server Certificate Assignment (TLSConfiguration_ReplaceServerCertificateAssignment)

Profile A Normative Reference: None

Profile C Normative Reference: None

Profile G Normative Reference: None

Profile Q Normative Reference: Conditional

Profile S Normative Reference: None

Test Purpose: To verify that Client is able to replace an existing key pair and certificate assignment to the TLS server on the Device by a new key pair and certificate assignment using **ReplaceServerCertificateAssignment** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **ReplaceServerCertificateAssignment** operation present.
- Device supports Security Configuration Service.
- Device supports TLSServerSupport.

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

1. Client invokes **ReplaceServerCertificateAssignment** request message to replace certificate assignment to a TLS server.
2. Device responds with code HTTP 200 OK and **ReplaceServerCertificateAssignmentResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.15.15 CREATE CERTIFICATION PATH

Test Label: Create Certification Path

Test Case ID: TLSCONFIGURATION-13

Feature Under Test: Create Certification Path (TLSConfiguration_CreateCertificationPath)

Profile A Normative Reference: None

Profile C Normative Reference: None

Profile G Normative Reference: None

Profile Q Normative Reference: Conditional

Profile S Normative Reference: None

Test Purpose: To verify that Client is able to create certification path using **CreateCertificationPath** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **CreateCertificationPath** operation present.
- Device supports Security Configuration Service.
- Device supports TLSServerSupport.

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

1. Client invokes **CreateCertificationPath** request message to create certification path.
2. Device responds with code HTTP 200 OK and **CreateCertificationPathResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

6.15.16 CREATE RSA KEY PAIR

Test Label: Create RSA Key Pair

Test Case ID: TLSCONFIGURATION-14

Feature Under Test: Create RSA Key Pair (TLSConfiguration_CreateRSAKeyPair)

Profile A Normative Reference: None

Profile C Normative Reference: None

Profile G Normative Reference: None

Profile Q Normative Reference: Conditional

Profile S Normative Reference: None

Test Purpose: To verify that Client is able to generate RSA key pair using **CreateRSAKeyPair** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **CreateRSAKeyPair** operation present.
- Device supports Security Configuration Service.
- Device supports RSAKeyPairGeneration.

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

1. Client invokes **CreateRSAKeyPair** request message to create RSA key pair.
2. Device responds with code HTTP 200 OK and **CreateRSAKeyPairResponse** message.

Test Result:

PASS -

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

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

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.

7.2 Set Synchronization Point Test Cases

7.2.1 Feature Level Requirement:

Validated Feature: Set Synchronization Point (SetSynchronizationPoint)

Check Condition based on Device Features: None

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

7.2.2 Expected Scenarios Under Test:

1. Client connects to Device to synchronize property states.
2. Client is considered as supporting Set Synchronization Point 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 if the following is TRUE:
   - No valid responses for **SetSynchronizationPoint** request OR
   - **SetSynchronizationPoint** request does not contains valid **wsa:Action** header.

7.2.3 SET SYNCHRONIZATION POINT

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

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

- [S3] It has HTTP 200 response code AND
- [S4] **soapenv:Body** element has child element **tev:SetSynchronizationPointResponse**

FAIL -

- The Client failed PASS criteria.

7.3 Unsubscribe Test Cases

Validated Feature: Unsubscribe (Unsubscribe)

Check Condition based on Device Features: None

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: Optional

7.3.1 Expected Scenarios Under Test:

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

7.3.2 UNSUBSCRIBE

Test Label: Unsubscribe - Unsubscribe

Test Case ID: UNSUBSCRIBE-1

Feature Under Test: Unsubscribe (Unsubscribe_UnsubscribeAction)

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: Optional

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

7.4 Keep Alive for Pull Point Event Handling Test Cases

7.4.1 Feature Level Requirement:

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

Check Condition based on Device Features: None

Required Number of Devices: 3

Profile A Requirement: Mandatory

Profile C Requirement: Mandatory

Profile S Requirement: Conditional

Profile Q Requirement: Optional

Profile G Requirement: Conditional

Profile T Requirement: Optional

7.4.2 Expected Scenarios Under Test:

1. Client connects to Device to initiate Pull Point Event Handling.
2. Client is considered as supporting Keep Alive for Pull Point Event Handling if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature AND
   - Client is able to renew pull point subscribtion using **Renew** operation OR **PullMessages** operation mechanism.
3. Client is considered as NOT supporting Keep Alive for Pull Point Event Handling if the following is TRUE:
   - No valid responses for **Renew** request AND for **CreatePullPointSubscription** request in the case if **PullMessages** used for keep alive OR
   - No valid responses for **Renew** request if detected OR
   - No valid responses for **CreatePullPointSubscription** request in the case if **PullMessages** used for keep alive if detected OR

- **Renew** request was invoked to address which was not specified in **tev:SubscriptionReference\wsa:Address** element of corresponding **CreatePullPointSubscriptionResponse** message.

7.4.3 RENEW

Test Label: Advanced Pull Point Event Handling - Renew

Test Case ID: KEEPALIVEFORPULLPOINTEVENTHANDLING-1

Feature Under Test: Renew (KeepAliveForPullPointEventHandling_Renew)

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: Optional

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

7.4.4 PULL MESSAGES AS KEEP ALIVE

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

Test Case ID: KEEPALIVEFORPULLPOINTEVENTHANDLING-2

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

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: Optional

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

8 Supplementary Features and Test Cases

8.1 METADATA STREAMING USING MEDIA2

Test Label: Metadata Streaming Using Media2

Test Case ID: MEDIA2_METADATASTREAMING-1

Feature Under Test: Metadata Streaming (Media2_MetadataStreaming_MetadataStreamingUsingMedia2)

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.

Annex A Test for Appendix A

A.1 Required Number of Devices Summary

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

Table A.1. Required Number of Devices Summary

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
tc.HTTPDigest	HTTP Digest	3	Digest	Digest
tc.Capabilities	Capabilities	3	None	All
tc.GetServices	Get Services	3	GetServices is supported by Device.	GetServices
tc.Discovery	Discovery	3	None	All
tc.DeviceDiscoveryTypeFilter	Device Discovery Type Filter	3	Device Discovery Type is supported by Device.	DiscoveryTypesTdsDevice
tc.UserHandling	User Handling	3	None	All
tc.TransitionToOperationalState	Transition to Operational State	3	Profile Q is supported by Device.	ProfileQSupported
tc.EventHandling	Event Handling	3	None	All
tc.NetworkConfiguration	Network Configuration	3	None	All
tc.System	System	3	None	All
tc.NTP	NTP	1	NTP is supported by Device.	NTP
tc.ZeroConfiguration	Zero Configuration	1	Zero Configuration is supported by Device.	ZeroConfiguration
tc.SystemDateAndTimeConfiguration	System Date and Time Configuration	1	None	All

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
tc.HTTPFirmwareUpgrade	HTTP Firmware Upgrade	1	HTTP Firmware Upgrade is supported by Device.	HttpFirmware Upgrade
tc.HTTPSystemBackup	HTTP System Backup	1	HTTP System Backup is supported by Device.	HttpSystemBackup
tc.HTTPSystemRestore	HTTP System Restore	1	HTTP System Backup is supported by Device.	HttpSystemBackup
tc.MonitoringNotifications	Monitoring Notifications	1	Monitoring/ProcessorUsage or Monitoring/ OperatingTime/ LastReset or Monitoring/OperatingTime/ LastReboot or Monitoring/OperatingTime/ LastClockSynchronization is supported by Device.	MonitoringProcessorUsageEvent OR MonitoringOperatingTimeLastResetEvent OR MonitoringOperatingTimeLastRebootEvent OR MonitoringOperatingTimeLastClockSynchronizationEvent
tc.DeviceManagementNotifications	Device Management Notifications	1	Check Condition based on Device Features: Device/ HardwareFailure/ FanFailure or Device/HardwareFailure/PowerSupplyFailure or Device/ HardwareFailure/ StorageFailure or Device/Hardw	MonitoringBackupLastEvent OR DeviceHardwareFailureFanFailureEvent OR DeviceHardwareFailurePowerSupplyFailureEvent OR DeviceHardwareFailureStor

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
			areFailure/TemperatureCritical or Monitoring/Backup/Last is supported by Device.	ageFailureEvent OR DeviceHardwareFailureTemperatureCriticalEvent
tc.HostnameConfiguration	Hostname Configuration	1	None	All
tc.DNSConfiguration	DNS Configuration	1	None	All
tc.NetworkProtocolsConfiguration	Network Protocols Configuration	1	None	All
tc.TLSConfiguration	TLS Configuration	1	TLS Server (Security Configuration Service) is supported by Device.	TLSServerSupport
tc.GetServicesWithCapabilities	Get Services with Capabilities	1	GetServices is supported by Device.	GetServices
tc.SetSynchronizationPoint	Set Synchronization Point	1	None	All
tc.KeepAliveForPullPointEventHandling	Keep Alive for Pull Point Event Handling	3	None	All