ONVIF®

TLS Configuration Add-on Client Test Specification

Version 23.06

June 2023

REVISION HISTORY

Vers.	Date	Description
23.06	Apr 18, 2023	The following was done according to #22: TLS Configuration section and TLS Configuration Test Cases section was moved from ONVIF Other Features Client Test Specifications to ONVIF TLS Configuration Add-on Client Test Specification. TLSCONFIGURATION-1 UPLOAD PASSPHRASE (removed) TLSCONFIGURATION-2 DELETE PASSPHRASE (removed) TLSCONFIGURATION-9 UPLOAD PKCS12 (removed) TLSCONFIGURATION-11 REMOVE SERVER CERTIFICATE ASSIGNMENT (removed) Expected Scenarios Under Test section was updated according to requirement in TLS Configuration add-on specification v1.0: AddServerCertificateAssignment command requirement level changed from mandatory to optional UploadPassphrase optional command requirement removed DeletePassphrase optional command requirement removed UploadCertificateWithPrivateKeyInPKCS12 mandatory command requirement removed RemoveServerCertificateAssignment mandatory command requirement removed
23.06	Mar 30, 2023	Initial version.

Table of Contents

1 Introduction

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

This particular document defines feaures and related test cases required for testing TLS Configuration Add-on features of a Client application e.g. TLS configuration. 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 TLS Configuration Add-on Client Test Specification defines and regulates the conformance testing procedure for the ONVIF conformant Clients in the scope of TLS Configuration Add-on 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 TLS Configuration Add-on features according to ONVIF Add-ons Specifications.

The principal intended purposes are:

- Provide self-assessment tool for implementations.
- Provide comprehensive test suite coverage for TLS Configuration Add-on 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 Add-on Mandatory Features

This section defines test cases which are mandatory for TLS Configuration Add-on Client conformance.

1.2.1 TLS Configuration

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

1.3 Supplementary Features and Test Cases

This section defines supplementary features and test cases which are not the part of profile, but TLS Configuration Add-on 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 TLS Configuration Add-on Specification:

  [TODO: put link to profile page] [http://TODO]

- [ONVIF Conformance] ONVIF Conformance Process Specification:

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

- [ONVIF Profile Policy] ONVIF Profile Policy:

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

- [ONVIF Network Interface Specs] ONVIF Network Interface Specification documents:

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

- [ONVIF Security Configuration Service] ONVIF Security Configuration Specifications:

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

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.
TLS Configuration Add-on	The TLS Configuration Add-on Specification.
Configuration Entity	A network video device media abstract component that produces or consumes a media stream on the network, i.e. video and/or audio stream.

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.

TLS Transport Layer Security

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 Security Configuration 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 TLS Configuration Add-on is an ONVIF Client that at least supports the following features:

- TLS Configuration.

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.
Normative References

For compatibility with the TLS Configuration Add-on, 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 TLS Configuration Test Cases

5.1.1 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: 3

Profile A Requirement: None

Profile C Requirement: None

Profile G Requirement: None

Profile S Requirement: None

5.1.2 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 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 EVENTHANDLING-1 PULLPOINT section) when **tns1:Advancedsecurity/Keystore/KeyStatus** event is supported AND
- Client may assign 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 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 NetworkProtocolsConfiguration_SetNetworkProtocols feature (please see NETWORKPROTOCOLSCONFIGURATION-2 SET NETWORK PROTOCOLS section).

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

- 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 EVENTHANDLING-1 PULLPOINT section) when Client supports **tns1:Advancedsecurity/Keystore/KeyStatus** notification if if MaximumNumberOfKeys is greater than zero on Device OR
- No valid responses for **AddServerCertificateAssignment** request if request is detected 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 NetworkProtocolsConfiguration_SetNetworkProtocols feature (please see NETWORKPROTOCOLSCONFIGURATION-2 SET NETWORK PROTOCOLS section).

5.1.3 PULLPOINT

Test Label: Event Handling - Pull Point

Test Case ID: EVENTHANDLING-1

Feature Under Test: Pull Point (EventHandling_PullPoint)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

5.1.4 SET NETWORK PROTOCOLS

Test Label: Network Protocols Configuration - Set Network Protocols

Test Case ID: NETWORKPROTOCOLSCONFIGURATION-2

Feature Under Test: Set Network Protocols (NetworkProtocolsConfiguration_SetNetworkProtocols)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

5.1.5 CREATE PKCS#10 CERTIFICATION

Test Label: Create PKCS#10 Certification

Test Case ID: TLSCONFIGURATION-3

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

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.

5.1.6 UPLOAD CERTIFICATE

Test Label: Upload Certificate

Test Case ID: TLSCONFIGURATION-4

Feature Under Test: Upload Certificate (TLSConfiguration_UploadCertificate)

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.

5.1.7 DELETE CERTIFICATE

Test Label: Delete Certificate

Test Case ID: TLSCONFIGURATION-5

Feature Under Test: Delete Certificate (TLSConfiguration_DeleteCertificate)

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.

5.1.8 DELETE CERTIFICATION PATH

Test Label: Delete Certification Path

Test Case ID: TLSCONFIGURATION-6

Feature Under Test: Delete Certification Path (TLSConfiguration_DeleteCertificationPath)

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.

5.1.9 DELETE KEY

Test Label: DeleteKey

Test Case ID: TLSCONFIGURATION-7

Feature Under Test: Delete Key (TLSConfiguration_DeleteKey)

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.

5.1.10 GET KEY STATUS

Test Label: Get Key Status

Test Case ID: TLSCONFIGURATION-8

Feature Under Test: Get Key Status (TLSConfiguration_GetKeyStatus)

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.

5.1.11 ADD SERVER CERTIFICATE ASSIGNMENT

Test Label: Add Server Certificate Assignment

Test Case ID: TLSCONFIGURATION-10

Feature Under Test: Add Server Certificate Assignment (TLSConfiguration_AddServerCertificateAssignment)

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.

5.1.12 REPLACE SERVER CERTIFICATE ASSIGNMENT

Test Label: Replace Server Certificate Assignment

Test Case ID: TLSCONFIGURATION-12

Feature Under Test: Replace Server Certificate Assignment (TLSConfiguration_ReplaceServerCertificateAssignment)

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.

5.1.13 CREATE CERTIFICATION PATH

Test Label: Create Certification Path

Test Case ID: TLSCONFIGURATION-13

Feature Under Test: Create Certification Path (TLSConfiguration_CreateCertificationPath)

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.

5.1.14 CREATE RSA KEY PAIR

Test Label: Create RSA Key Pair

Test Case ID: TLSCONFIGURATION-14

Feature Under Test: Create RSA Key Pair (TLSConfiguration_CreateRSAKeyPair)

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.

6 Supplementary Features and Test Cases

6.1 PULLPOINT

Test Label: Event Handling - Pull Point

Test Case ID: EVENTHANDLING-1

Feature Under Test: Pull Point (EventHandling_PullPoint)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

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

FAIL -

- The Client failed PASS criteria.

6.2 SET NETWORK PROTOCOLS

Test Label: Network Protocols Configuration - Set Network Protocols

Test Case ID: NETWORKPROTOCOLSCONFIGURATION-2

Feature Under Test: Set Network Protocols (NetworkProtocolsConfiguration_SetNetworkProtocols)

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

Pre-Requisite:

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

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

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

Test Result:

PASS -

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

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

FAIL -

- The Client failed PASS criteria.

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.TLSConfiguration	TLS Configuration	3	TLS Server (Security Configuration Service) is supported by Device.	TLSServerSupport