ONVIF™

Advanced Security Client Test Specification

Version 16.07

July 2016

REVISION HISTORY

Vers.	Date	Description
16.07	May 11, 2016	Expected Scenarios Under of TLS Configuration was updated.
16.07	Mar 16, 2016	Docbook stylesheets were updated.
16.07	Mar 16, 2016	Feature was renamed from Advanced Security to TLS Configuration Tests prefixes were changed from ADVANCEDSECURITY to TLSCONFIGURATION TLSCONFIGURATION-13 test case was added TLSCONFIGURATION-14 test case was added tls_configuration.scenario section was updated
16.07	Mar 10, 2016	UploadCertificate added in ADVANCEDSECURITY-3 test case
16.07	Feb 25, 2016	ADD SERVER CERTIFICATE ASSIGNMENT Test Case added REMOVE SERVER CERTIFICATE ASSIGNMENT Test Case added REPLACE SERVER CERTIFICATE ASSIGNMENT Test Case added
16.07	Dec 30, 2015	Initial version: General parts added UPLOAD PASSPHRASE Test Case added DELETE PASSPHRASE Test Case 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 [http://www.onvif.org/Documents/Specifications.aspx]).

This particular document defines test cases required for testing Imaging Service features of a Client application e.g. Get Imaging Capabilities, Video Sources List, Get Imaging Settings, Imaging Settings Configuration, Focus Control. It also describes the test framework, test setup, prerequisites, test policies needed for the execution of the described test cases.

1.1 Scope

This ONVIF Advanced Security Client Test Specification defines and regulates the conformance testing procedure for the ONVIF conformant Clients in the scope of Advanced Security Service 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 Advanced Security Service features according to ONVIF Advanced Security Service Specification.

The principal intended purposes are:

- Provide self-assessment tool for implementations.
- Provide comprehensive test suite coverage for Advanced Security Service 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 TLS Configuration

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

2 Normative references

- ONVIF Conformance Process Specification:

  http://www.onvif.org/Documents/Specifications.aspx

- ONVIF Profile Policy:

  http://www.onvif.org/Documents/Specifications.aspx

- ONVIF Core Specifications:

  http://www.onvif.org/Documents/Specifications.aspx

- ONVIF Core Client Test Specification:

  http://www.onvif.org/Documents/Specifications.aspx

- ONVIF Advanced Security Specification:

  http://www.onvif.org/Documents/Specifications.aspx

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

  http://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/

- W3C Web Services Addressing 1.0 – Core:

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

3 Terms and Definitions

3.1 Conventions

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

3.2 Definitions

This section describes terms and definitions used in this document.

Profile	See ONVIF Profile Policy.
ONVIF Device	Computer appliance or software program that exposes one or multiple ONVIF Web Services.
ONVIF Client	Computer appliance or software program that uses ONVIF Web Services.
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.
Advanced Security Service	Service for keystore and a TLS server on an ONVIF device.
Valid Device Response	Device has responded to specific request with code HTTP or RTSP 200 OK and SOAP fault message has not appeared.

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.
URI	Uniform Resource Identifier.
WSDL	Web Services Description Language.

XML eXtensible Markup Language.

3.4 Namespaces

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

Table 3.1. Defined namespaces in this specification

Prefix	Namespace URI	Description
soapenv	http://www.w3.org/2003/05/soap-envelope	Envelope namespace as defined by SOAP 1.2 [SOAP 1.2, Part 1]
xs	http://www.w3.org/2001/XMLSchema	Instance namespace as defined by XS [XML-Schema, Part1] and [XMLSchema,Part 2]
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 with Advanced Security features support can provide key configuration, certificate configuration and TLS server 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 and feature requirement level for the Profiles, which will be used for Profiles conformance.

If Feature Level Requirement is defined as Mandatory for some Profile, Client shall pass Expected Scenario Under Test for each Device with this Profile support to claim this Profile Conformance.

If Feature Level Requirement is defined as Conditional, Optional for some Profile, Client shall pass Expected Scenario Under Test for at least one Device with this Profile support to claim feature as supported.

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.
- Validated Feature List - list of features ID related to this test case.

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 ONVIF compatibility, 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 TLS Configuration Test Cases

Validated Feature: tls_configuration

Profile A Requirement: None

Profile C Requirement: None

Profile G Requirement: None

Profile Q Requirement: Conditional

Profile S Requirement: None

5.1 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).

5.2 UPLOAD PASSPHRASE

Test Label: Upload Passphrase

Test Case ID: TLSCONFIGURATION-1

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

Feature Under Test: Upload Passphrase

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 Advanced Security 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.

Validated Feature List: tls_configuration.upload_passphrase

5.3 DELETE PASSPHRASE

Test Label: Delete Passphrase

Test Case ID: TLSCONFIGURATION-2

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

Feature Under Test: Delete Passphrase

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 Advanced Security 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.

Validated Feature List: tls_configuration.delete_passphrase

5.4 CREATE PKCS#10 CERTIFICATION

Test Label: Create PKCS#10 Certification

Test Case ID: TLSCONFIGURATION-3

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

Feature Under Test: Create PKCS#10 Certification

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 Advanced Security 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.

Validated Feature List: tls_configuration.create_pkcs10

5.5 UPLOAD CERTIFICATE

Test Label: Upload Certificate

Test Case ID: TLSCONFIGURATION-4

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

Feature Under Test: Upload Certificate

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 Advanced Security 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.

Validated Feature List: tls_configuration.upload_certificate

5.6 DELETE CERTIFICATE

Test Label: Delete Certificate

Test Case ID: TLSCONFIGURATION-5

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

Feature Under Test: Delete Certificate

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 Advanced Security 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.

Validated Feature List: tls_configuration.delete_certificate

5.7 DELETE CERTIFICATION PATH

Test Label: Delete Certification Path

Test Case ID: TLSCONFIGURATION-6

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

Feature Under Test: Delete Certification Path

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 Advanced Security 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.

Validated Feature List: tls_configuration.delete_certification_path

5.8 DELETE KEY

Test Label: DeleteKey

Test Case ID: TLSCONFIGURATION-7

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

Feature Under Test: Delete Key

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 Advanced Security 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.

Validated Feature List: tls_configuration.delete_key

5.9 GET KEY STATUS

Test Label: Get Key Status

Test Case ID: TLSCONFIGURATION-8

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

Feature Under Test: Delete Key

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 Advanced Security 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.

Validated Feature List: tls_configuration.get_key_status

5.10 UPLOAD PKCS12

Test Label: Upload PKCS12

Test Case ID: TLSCONFIGURATION-9

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

Feature Under Test: Delete Key

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 Advanced Security 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.

Validated Feature List: tls_configuration.upload_pkcs12

5.11 ADD SERVER CERTIFICATE ASSIGNMENT

Test Label: Add Server Certificate Assignment

Test Case ID: TLSCONFIGURATION-10

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

Feature Under Test: Add Server Certificate Assignment

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 Advanced Security 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.

Validated Feature List: tls_configuration.add_server_certificate_assignment

5.12 REMOVE SERVER CERTIFICATE ASSIGNMENT

Test Label: Remove Server Certificate Assignment

Test Case ID: TLSCONFIGURATION-11

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

Feature Under Test: Remove Server Certificate Assignment

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 Advanced Security 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.

Validated Feature List: tls_configuration.remove_server_certificate_assignment

5.13 REPLACE SERVER CERTIFICATE ASSIGNMENT

Test Label: Replace Server Certificate Assignment

Test Case ID: TLSCONFIGURATION-12

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

Feature Under Test: Replace Server Certificate Assignment

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 Advanced Security 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.

Validated Feature List: tls_configuration.replace_server_certificate_assignment

5.14 CREATE CERTIFICATION PATH

Test Label: Create Certification Path

Test Case ID: TLSCONFIGURATION-13

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

Feature Under Test: Create Certification Path

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 Advanced Security 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.

Validated Feature List: tls_configuration.create_certification_path

5.15 CREATE RSA KEY PAIR

Test Label: Create RSA Key Pair

Test Case ID: TLSCONFIGURATION-14

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

Feature Under Test: Create RSA Key Pair

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 Advanced Security 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.

Validated Feature List: tls_configuration.create_rsa_key_pair