



# ONVIF™

## Advanced Security Test Specification

Version 16.07

July 2016



© 2016 by ONVIF, Inc. All rights reserved.

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

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

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

### Revision History

Ver.	Date	Description
0.6	Feb 26, 2013	Initial version
0.7	Apr 16, 2013	Adapted document to Advanced Security Service Specification v1.0_RC1
0.7.1	May 8, 2013	Added key store tests
0.8	May 11, 2013	First draft for WG review
0.81	Jun 11, 2013	Minor changes after Singapore F2F
0.82	Jun 13, 2013	Partial fix for ticket #1079
0.9	Jun 14, 2013	Added appendices A.8 to A.11 and modified Sect. 5.3.14.
13.06	Dec, 2013	First issue of Advanced Security Test Specification
14.06	Jun, 2014	<p>New tests were added:</p> <p>Create PKCS#10 – negative test,</p> <p>Delete Certificate – CA – Preserve Public Key,</p> <p>Upload certificate – delete linked key (negative test),</p> <p>Upload certificate – Upload malformed certificate (negative test),</p> <p>Upload certificate – Upload expired certificate,</p> <p>TLS Server Certificate - self-signed,</p> <p>TLS Server Certificate – CA.</p> <p>Annex A.21, A.22 were added.</p>
14.12	Dec, 2014	<p>The following test cases were added:</p> <p>Create PKCS#10 – Subject Test,</p> <p>Create self-signed certificate – Subject Test,</p> <p>Upload Passphrase, Delete Passphrase,</p> <p>Upload PKCS8 – no key pair exists,</p>

		<p>Upload PKCS8 – decryption fails,          Upload PKCS8 – key pair without private key exists,          Upload PKCS12 – no key pair exists,          Upload PKCS12 – decryption fails,          Upload PKCS12 – key pair without private key exists.</p> <p>The following test cases were modified with ID change:</p> <p>Basic TLS Handshake,          Basic TLS Handshake after Replace Server Certificate Assignment.</p> <p>The following annexes were added:          A.23, A.24, A.25, A.26, A.27, A.28, A.29, A.30, A.31, A.32, A.33, A.34, A.35.</p>
<p>15.06</p>	<p>Jun, 2015</p>	<p>The following test cases were added:</p> <p>ADVANCED_SECURITY-3-2-5 Basic TLS Handshake with Replace Server Certification Path and PKCS#12</p> <p>ADVANCED_SECURITY-6-3-4 Upload PKCS12 - verify key and certificate</p> <p>ADVANCED_SECURITY-2-1-27 CreateSelfSignedCertificate with PKCS#12</p> <p>ADVANCED_SECURITY-2-1-28 Create PKCS#10 request with PKCS#12</p> <p>ADVANCED_SECURITY-8-1-1 Upload CRL</p> <p>ADVANCED_SECURITY-8-1-2 Delete CRL</p> <p>ADVANCED_SECURITY-8-1-3 Get CRL</p> <p>ADVANCED_SECURITY-8-1-4 Create certification path validation policy</p> <p>ADVANCED_SECURITY-8-1-5 Get certification path validation policy</p> <p>ADVANCED_SECURITY-8-1-6 Delete certification path validation policy</p> <p>ADVANCED_SECURITY-3-3-1 TLS client authentication – self-signed TLS server certificate</p>

		<p>with on-device RSA key pair</p> <p>ADVANCED_SECURITY-3-3-2 CRL processing with on-device RSA key pair</p> <p>ADVANCED_SECURITY-3-3-3 Replace certification path validation policy assignment</p> <p>The following test cases were modified:</p> <p>ADVANCED_SECURITY-1-1-3 Check private Key status for an RSA private key</p> <p>ADVANCED_SECURITY-2-1-1 Create PKCS#10 certification requests</p> <p>ADVANCED_SECURITY-2-1-2 Create self-signed certificate</p> <p>ADVANCED_SECURITY-2-1-3 Upload certificate – Keystore contains private key</p> <p>ADVANCED_SECURITY-2-1-6 Get certificate – self-signed</p> <p>ADVANCED_SECURITY-2-1-8 Get all certificates – self signed</p> <p>ADVANCED_SECURITY-2-1-10 Delete Certificate – self signed</p> <p>ADVANCED_SECURITY-2-1-12 Create Certification Path – self-signed</p> <p>ADVANCED_SECURITY-2-1-13 Create Certification Path – CA</p> <p>ADVANCED_SECURITY-2-1-14 Get Certification Path – self-signed</p> <p>ADVANCED_SECURITY-2-1-15 Get Certification Path – CA</p> <p>ADVANCED_SECURITY-2-1-16 Get All Certification Paths – self-signed</p> <p>ADVANCED_SECURITY-2-1-17 Get All Certification Paths – CA</p> <p>ADVANCED_SECURITY-2-1-18 Delete Certification Path – self-signed</p> <p>ADVANCED_SECURITY-2-1-19 Delete Certification Path - CA</p> <p>ADVANCED_SECURITY-2-1-20 Create PKCS#10</p>
--	--	---

		<p>(negative test)</p> <p>ADVANCED_SECURITY-2-1-22 Upload certificate – delete linked key (negative test)</p> <p>ADVANCED_SECURITY-3-1-1 Add Server Certificate Assignment – self-signed</p> <p>ADVANCED_SECURITY-3-1-2 Add Server Certificate Assignment – CA</p> <p>ADVANCED_SECURITY-3-1-3 Replace Server Certificate Assignment – self-signed</p> <p>ADVANCED_SECURITY-3-1-4 Replace Server Certificate Assignment – CA</p> <p>ADVANCED_SECURITY-3-1-5 Get Assigned Server Certificates – self-signed</p> <p>ADVANCED_SECURITY-3-1-6 Get Assigned Server Certificates – CA</p> <p>ADVANCED_SECURITY-3-1-7 Remove Server Certificate Assignment – self-signed</p> <p>ADVANCED_SECURITY-3-1-8 Remove Server Certificate Assignment – CA</p> <p>ADVANCED_SECURITY-3-2-3 Basic TLS Handshake</p> <p>ADVANCED_SECURITY-3-2-4 Basic TLS Handshake after Replace Server Certificate Assignment</p> <p>ADVANCED_SECURITY-4-1-1 TLS Server Certificate - self-signed</p> <p>ADVANCED_SECURITY-4-1-2 TLS Server Certificate – CA</p> <p>ADVANCED_SECURITY-5-1-1 Advanced Security Service Capabilities</p> <p>ADVANCED_SECURITY-5-1-2 Get Services and Get Advanced Security Service Capabilities Consistency</p> <p>The following annexes were added:</p> <p>A.36, A.37, A.38, A.39, A.40, A.41, A.42, A.43, A.44, A.45, A.46</p>
--	--	--



		The following annexes were modified: A.4, A.8, A.11, A.13, A.14, A.16, A.18, A.30, A.35
16.06	Mar 2016	The ADVANCED_SECURITY-3-2-4 has been updated.



Table of Contents1	Introduction
12	
1.1 Scope	12
1.1.1 Keystore	12
1.1.2 Certificate Management	13
1.1.3 TLS Server	13
1.1.4 Referential integrity	14
1.1.5 Capabilities	14
1.1.6 Off-Device Key Generation Operations	14
1.1.7 Certificate-based Client Authentication	14
2 Terms and Definitions	15
2.1 Definitions	15
2.2 Abbreviations	15
3 Test Overview	16
3.1 Test Setup	16
3.1.1 Network Configuration for DUT	16
3.2 Prerequisites	17
3.3 Test Policy	17
3.3.1 General Policy	17
3.3.2 Keystore	18
3.3.3 Certificate Management	18
3.3.4 TLS Server	19
3.3.5 Referential integrity	19
3.3.6 Capabilities	19
3.3.7 Off-Device Key Generation Operations	20
3.3.8 Certificate-based Client Authentication	20
4 Advanced Security Test Cases	22
4.1 Keystore	22
4.1.1 Create RSA Key Pair, status through polling	22
4.1.2 Create RSA Key Pair, status through event	25
4.1.3 Check private Key status for an RSA private key	29
4.1.4 Get all keys	31
4.1.5 Delete key	33
4.2 Certificate Management	36
4.2.1 Create PKCS#10 certification requests	36
4.2.2 Create self-signed certificate	38
4.2.3 Upload certificate – Keystore contains private key	40
4.2.4 Upload certificate – Keystore contains private key (negative test)	42
4.2.5 Upload certificate – Keystore does not contain private key	44





- 4.2.6 Get certificate – self-signed ..... 46
- 4.2.7 Get certificate – CA ..... 48
- 4.2.8 Get all certificates – self signed ..... 50
- 4.2.9 Get All Certificate – CA ..... 52
- 4.2.10 Delete Certificate – self signed ..... 54
- 4.2.11 Delete Certificate – CA ..... 57
- 4.2.12 Create Certification Path – self-signed ..... 60
- 4.2.13 Create Certification Path – CA ..... 62
- 4.2.14 Get Certification Path – self-signed ..... 65
- 4.2.15 Get Certification Path – CA ..... 67
- 4.2.16 Get All Certification Paths – self-signed ..... 69
- 4.2.17 Get All Certification Paths – CA ..... 72
- 4.2.18 Delete Certification Path – self-signed ..... 75
- 4.2.19 Delete Certification Path - CA ..... 78
- 4.2.20 CreatePKCS10CSR – negative test ..... 81
- 4.2.21 DeleteCertificate – CA – Preserve Public Key ..... 85
- 4.2.22 Upload certificate – delete linked key (negative test) ..... 88
- 4.2.23 Upload certificate – Upload malformed certificate (negative test) ..... 90
- 4.2.24 Upload certificate – Upload expired certificate ..... 92
- 4.2.25 CreateSelfSignedCertificate with PKCS#12 ..... 95
- 4.2.26 Create PKCS#10 request with PKCS#12 ..... 99
- 4.3 TLS Server ..... 104
  - 4.3.1 Certificate Management ..... 104
  - 4.3.2 TLS Handshaking ..... 131
  - 4.3.3 TLS client authentication ..... 148
- 4.4 Referential integrity ..... 165
  - 4.4.1 TLS Server Certificate - self-signed ..... 165
  - 4.4.2 TLS Server Certificate – CA ..... 169
- 4.5 Capabilities ..... 175
  - 4.5.1 Advanced Security Service Capabilities ..... 175
  - 4.5.2 Get Services and Get Advanced Security Service Capabilities Consistency ..... 179
- 4.6 Off-Device Key Generation Operations ..... 182
  - 4.6.1 Passphrase Management ..... 182
  - 4.6.2 Key Management ..... 186
  - 4.6.3 Certificate Management ..... 191
- 4.7 Certificate-based Client Authentication ..... 202
  - 4.7.1 Upload CRL ..... 202
  - 4.7.2 Delete CRL ..... 204
  - 4.7.3 Get CRL ..... 206



4.7.4	Create certification path validation policy .....	208
4.7.5	Get certification path validation policy .....	211
4.7.6	Delete certification path validation policy .....	213
Annex A	.....	216
A.1	Delete an RSA key pair .....	216
A.2	Subject for a server certificate.....	217
A.3	Creating a certificate from a PCKS#10 request .....	217
A.4	Provide CA certificate .....	218
A.5	Delete a certification path with corresponding certificate and RSA key pair.....	219
A.6	Determine RSA key length .....	221
A.7	Create an RSA key pair.....	222
A.8	Create a self-signed certificate.....	224
A.9	Delete a certificate with corresponding RSA key pair .....	226
A.10	Get service capabilities .....	227
A.11	Create a certification path based on self-signed certificate .....	228
A.12	Remove server certificate assignment with corresponding certification path, certificate and RSA key pair 230	
A.13	Add server certificate assignment with corresponding certification path, self-signed certificate and RSA key pair .....	232
A.14	Create a CA-signed certificate for RSA key pair .....	234
A.15	Upload a certificate without Private Key Assignment.....	236
A.16	Create and upload a CA-signed certificate for private key.....	237
A.17	Delete a certification path with corresponding two certificates and RSA key pairs .....	239
A.18	Create certification path with CA-signed certificate and associated CA certificate.....	241
A.19	Validate DER encoding .....	243
A.20	Remove server certificate assignment with corresponding certification path, certificates and RSA key pairs.....	244
A.21	Basic TLS handshake .....	246
A.22	Provide expired CA certificate .....	248
A.23	Delete a passphrase .....	249
A.24	Passphrases for test cases .....	250
A.25	Creating a PKCS#8 data structure with new public key and private key without passphrase .....	250
A.26	Generating an RSA key pair .....	251
A.27	Creating a PKCS#8 data structure with existing public key and private key without passphrase .....	252
A.28	Creating a PKCS#8 data structure with new public key and private key with passphrase .....	253
A.29	Creating a PKCS#8 data structure with existing public key and private key with passphrase .....	254
A.30	Creating a PKCS#12 data structure with new CA-signed certificate signed by new public key and private key without passphrase.....	255
A.31	Creating a PKCS#12 data structure with new CA-signed certificate signed by new public key and private key with passphrase.....	256



A.32	Creating a PKCS#12 data structure with existing CA-signed certificate and a corresponding public key and private key without passphrase .....	257
A.33	Creating a PKCS#12 data structure with existing CA-signed certificate and a corresponding public key and private key with passphrase .....	258
A.34	Subject for a server certificate (all DN-attributes) .....	260
A.35	Delete a certification path with corresponding certificate and RSA key pair when CertificateID is unknown .....	261
A.36	Upload PKCS#12 – no key pair exists .....	265
A.37	Provide CRL .....	267
A.38	Delete a CRL .....	268
A.39	Upload a CRL .....	269
A.40	Delete a certification path validation policy .....	270
A.41	Prepare certificate on the DUT .....	271
A.42	Create a certification path validation policy .....	273
A.43	Provide certificate signed by private key of other certificate .....	274
A.44	Create a certification path validation policy with provided certificate identifier .....	275
A.45	Provide CRL for specified certificate .....	277
A.46	Upload a passphrase .....	278



## 1 Introduction

The goal of the ONVIF test specification set is to make it possible to realize fully interoperable IP physical security implementation from different vendors. The set of ONVIF test specification describes the test cases need to verify the [ONVIF Core Specs] and [ONVIF Conformance] requirements. In addition, the test cases are to be basic inputs for some Profile specification requirements. It also describes the test framework, test setup, pre-requisites, test policies needed for the execution of the described test cases.

This ONVIF Advanced Security Test Specification acts as a supplementary document to the [ONVIF Core Specs], illustrating test cases need to be executed and passed. In addition, this specification acts as an input document to the development of test tool that will be used to test the ONVIF device implementation conformance towards ONVIF standard. This test tool is referred as ONVIF Client hereafter.

### 1.1 Scope

This ONVIF Advanced Security Test Specification defines and regulates the conformance testing procedure for the ONVIF conformant devices. Conformance testing is meant to be functional black-box testing. The objective of this specification is to provide test cases to test individual requirements of ONVIF devices according to the ONVIF Advanced Security Service, which is defined in [ONVIF Advanced Security Service].

The principal intended purposes are:

- To provide self-assessment tool for implementations.
- To provide comprehensive test suite coverage for [ONVIF Core Specs].

This specification does not address the following.

- Product use cases and non-functional (performance and regression) testing.
- SOAP Implementation Interoperability test i.e. Web Service Interoperability Basic Profile version 2.0 (WS-I BP 2.0).
- Full coverage of network protocol implementation test for HTTP, HTTPS, RTP, RTSP, and TLS protocols.

The set of ONVIF Test Specification will not cover the complete set of requirements as defined in [ONVIF Core Specs]; instead, it will cover its subset.

This ONVIF Advanced Security Test Specification covers the ONVIF Advanced Security Service, which is a functional block of [ONVIF Core Specs]. The following section gives a brief overview of each functional block and its scope.

#### 1.1.1 Keystore

The Keystore section covers the test cases needed for storage and management of keys on an ONVIF device.

The scope of this specification section is to cover the following functions:

- Create RSA Key Pair
- Get Key Status



- Get Private Key Status
- Get All Keys
- Delete Key

### 1.1.2 Certificate Management

The Certificate Management section covers the test cases needed for storage and management of certificates on an ONVIF device.

The scope of this specification section is to cover the following functions:

- Create PKCS#10 Certification Request
- Create Self-Signed Certificate
- Upload Certificate
- Get Certificate
- Get All Certificates
- Delete Certificate
- Create Certification Path
- Get Certification Path
- Get All Certification Paths
- Delete Certification Path

### 1.1.3 TLS Server

The TLS Server section covers the test cases needed for configuring the TLS server on an ONVIF device.

The scope of this specification section is to cover the following functions:

- Add Server Certificate Assignment
- Remove Server Certificate Assignment
- Replace Server Certificate Assignment
- Get Assigned Server Certificates
- Basic TLS Handshake
- TLS client authentication
- Add certification path validation policies assignment
- Delete certification path validation policies assignment
- Replace certification path validation policy assignment



- Get certification path validation policies assignment

#### **1.1.4 Referential integrity**

The Referential integrity section covers the test cases needed for referential integrity checks on an ONVIF device.

#### **1.1.5 Capabilities**

The Capabilities section covers the test cases needed for getting capabilities from an ONVIF device.

The scope of this specification section is to cover the following functions:

- Getting capabilities with GetServiceCapabilities command
- Getting capabilities with GetServices command

#### **1.1.6 Off-Device Key Generation Operations**

The Off-Device Key Generation Operations section covers the test cases needed for uploading keys to an ONVIF device, potentially along with a certificate for the key, based on the PKCS#8 [RFC 5859] and PKCS#12 [PKCS#12] data structures.

The scope of this specification section is to cover the following functions:

- Upload Passphrase
- Delete Passphrase
- Upload key pair in PKCS#8 data structure
- Upload certificate with private key in PKCS#12 data structure

#### **1.1.7 Certificate-based Client Authentication**

The Certificate-based Client Authentication section covers the test cases needed for CRL management on an ONVIF device.

The scope of this specification section is to cover the following functions:

- Upload CRL
- Get All CRLs
- Delete CRL
- Create certification path validation policy
- Get certification path validation policies
- Delete certification path validation policy
- Get certification path validation policy



## 2 Terms and Definitions

### 2.1 Definitions

This section defines terms that are specific to the ONVIF Advanced Security Service and tests. For a list of applicable general terms and definitions, please see [ONVIF Base Test].

<b>Key</b>	A key is an input to a cryptographic algorithm. Sufficient randomness of the key is usually a necessary condition for the security of the algorithm. This specification supports RSA key pairs as keys.
<b>Key Pair</b>	A key that consists of a public key and (optionally) a private key.
<b>RSA key pair</b>	A key pair that is accepted as input by the RSA algorithm.
<b>Digital Signature</b>	A digital signature for an object allows to verify the object's authenticity, i.e., to check whether the object has in fact been created by the signer and has not been modified afterwards. A digital signature is based on a key pair, where the private key is used to create the signature and the public key is used for verification of the signature.
<b>Certificate</b>	A certificate as used in this specification binds a public key to a subject entity. The certificate is digitally signed by the certificate issuer (the certification authority) to allow for verifying its authenticity.
<b>Certification Path</b>	A certification path is a sequence of certificates in which the signature of each certificate except for the last certificate can be verified with the subject public key in the next certificate in the sequence.
<b>Certification Authority</b>	A certification authority is an entity that issues certificates to subject entities.
<b>Alias</b>	An alias is a name for an object on the device that is chosen by the client and treated transparently by the device.

### 2.2 Abbreviations

This section describes abbreviations used in this document.

<b>CA</b>	Certification Authority
<b>CSR</b>	Certificate Signing Request (also called Certification Request)
<b>SHA</b>	Secure Hashing Algorithm
<b>TLS</b>	Transport Layer Security

### 3 Test Overview

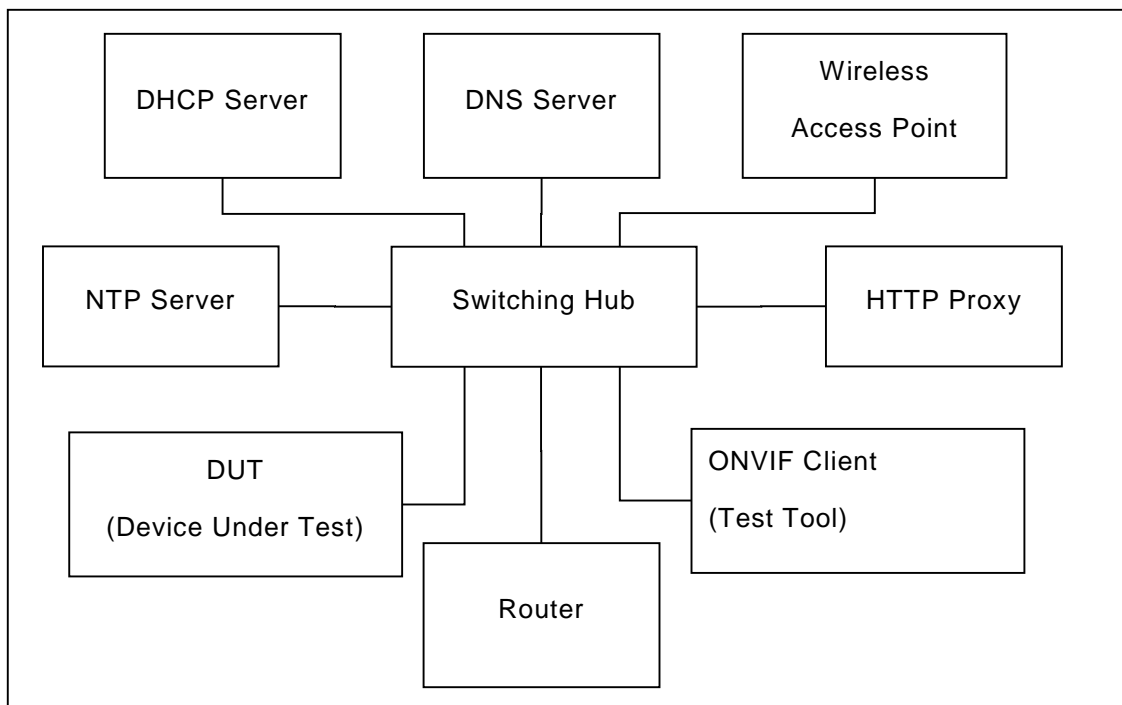
This section provides information the test setup procedure and required prerequisites, and the test policies that should be followed for test case execution.

#### 3.1 Test Setup

##### 3.1.1 Network Configuration for DUT

The generic test configuration for the execution of test cases defined in this document is as shown below (Figure 1).

Based on the individual test case requirements, some of the entities in the below setup may not be needed for the execution of those corresponding test cases.



**Figure 1: Test Configuration for DUT**

**DUT:** ONVIF device to be tested. Hereafter, this is referred to as DUT (Device Under Test).

**ONVIF Client (Test Tool):** Tests are executed by this system, and it controls the behaviour of the DUT. It handles both expected and unexpected behaviour.

**HTTP Proxy:** provides facilitation in case of RTP and RTSP tunnelling over HTTP.

**Wireless Access Point:** provides wireless connectivity to the devices that support wireless connection.

**DNS Server:** provides DNS related information to the connected devices.

**DHCP Server:** provides IPv4 Address to the connected devices.

**NTP Server:** provides time synchronization between ONVIF Client and DUT.





### **3.2 Prerequisites**

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

- The DUT shall be configured with an IPv4 address.
- The DUT shall be IP reachable in the test configuration.
- The DUT shall be able to be discovered by the Test Tool.
- The DUT shall be configured with the time, i.e. manual configuration of UTC time and if NTP is supported by the DUT then NTP time shall be synchronized with NTP Server.
- The DUT time and Test tool time shall be synchronized with each other either manually or by a common NTP server.
- The ONVIF Client supports both WS-Security Username Token profile and HTTP digest authentication as authentication functionalities and selects the authentication method to use based on the procedure defined in Sect. 3.3.6 (Authentication method selection as a testing framework) of [ONVIF Base Test Spec].
- The user account that is used by the ONVIF Client for issuing commands to the DUT has administrative rights.
- The ONVIF Client shall have access to a certification authority.
- The DUT shall have enough free storage capacity for RSA key pairs that is required for test cases (see test cases pre-requisites for more information).
- The DUT shall have enough free storage capacity for certificates that is required for test cases (see test cases pre-requisites for more information).
- The DUT shall have enough free storage capacity for certification paths that is required for test cases (see test cases pre-requisites for more information).
- The DUT shall have enough free storage capacity for server certificate assignment that is required for test cases (see test cases pre-requisites for more information).

### **3.3 Test Policy**

This section describes the test policies specific to the test case execution of each functional block.

The DUT shall adhere to the test policies defined in this section.

#### **3.3.1 General Policy**

The test policies specific to the test case execution of all functional blocks:

- If a DUT method produces a fault that is not explicitly stated as expected in the test procedure of a test case, the result of the test case shall be FAIL.
- Assertions in a test procedure are defined using the verb verify, e.g., “ONVIF Client verifies that list l contains ID x”, with the following semantics:
  - If the assertion holds, the test proceeds with the next step in the test procedure.
  - If the assertion does not hold, the test result shall be FAIL.



### 3.3.2 Keystore

The test policies specific to the test case execution of Keystore functional block:

- DUT shall give the Advanced Security Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.
- The DUT shall support on-board generation of an RSA key pair.
- The following tests are performed about key management
  - The DUT generates an RSA key pair status handling is done with polling.
  - The DUT generates an RSA key pair status handling is done with event.
  - The DUT returns whether a key pair in the keystore contains a private key.
  - The status of a key in the DUT's keystore is returned correctly.
  - A key is deleted correctly from the keystore on the DUT.

Please, refer to Section 4.1 for Keystore Test Cases.

### 3.3.3 Certificate Management

The test policies specific to the test case execution of Certificate Management functional block:

- DUT shall give the Advanced Security Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.
- The DUT shall support generating a PKCS#10 certification request.
- The DUT shall support creating a self-signed certificate.
- The following tests are performed about certificate management
  - The DUT correctly supports external certification for a key pair in the keystore.
  - The DUT correctly generates a self-signed certificate for a key pair in the keystore.
  - The ONVIF Client can upload a certificate to the DUT.
  - A certificate from the keystore on the DUT is correctly returned to the ONVIF client.
  - All certificates in the keystore on the DUT are correctly returned to the ONVIF client.
  - The ONVIF Client can delete a certificate from the keystore on the DUT.
  - Certificates in the keystore on the DUT can be correctly combined to a certification path.
  - A certification path stored in the keystore on the DUT can be correctly deleted.

Please, refer to Section 4.2 for Certificate Management Test Cases.



### 3.3.4 TLS Server

The test policies specific to the test case execution of TLS Server functional block:

- DUT shall give the Advanced Security Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.
- The DUT shall implement a TLS server.
- The following tests are performed for the TLS server
  - A certification path is assigned to the TLS server.
  - A certification path is received from the TLS server.
  - A certification path assignment is removed from the TLS server.
  - A certification path assignment to the TLS server is replaced by another certification path assignment.
  - Basic TLS Handshake
  - Basic TLS Handshake after Replace Server Certificate Assignment
- The following tests are performed for the TLS server in case certificate along with an RSA private key in a PKCS#12 data structure upload is supported by the DUT
  - Basic TLS Handshake with Replace Server Certification Path and PKCS#12
- The following tests are performed for the TLS server in case TLS client authentication is supported by the DUT
  - TLS client authentication – self-signed TLS server certificate with on-device RSA key pair
  - CRL processing with on-device RSA key pair
  - Replace certification path validation policy assignment

Please, refer to Section 4.3 for TLS Server Test Cases.

### 3.3.5 Referential integrity

The test policies specific to the test case execution of Referential integrity functional block:

- DUT shall give the Advanced Security Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.
- The DUT shall implement a TLS server.
- The following tests are performed for the TLS server
  - Referential integrity of certificate assigned to a TLS server.

Please, refer to Section 4.4 for Referential integrity Test Cases.

### 3.3.6 Capabilities

The test policies specific to the test case execution of Capabilities functional block:



- DUT shall give the Advanced Security Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.
- The following tests are performed
  - Getting capabilities with GetServiceCapabilities command
  - Getting capabilities with GetServices command

Please, refer to Section 4.5 for Capabilities Test Cases.

### 3.3.7 Off-Device Key Generation Operations

The test policies specific to the test case execution of Off-Device Key Generation Operations functional block:

- DUT shall give the Advanced Security Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.
- The following tests are performed
  - Uploading passphrase with UploadPassphrase command
  - Deleting passphrase with DeletePassphrase command
  - Upload key pair in PKCS#8 data structure with UploadKeyPairInPKCS8 command
  - Upload certificate with private key in PKCS#12 data structure with UploadCertificateWithPrivateKeyInPKCS12 command

Please, refer to Section 4.6 for Off-Device Key Generation Operations Test Cases.

### 3.3.8 Certificate-based Client Authentication

The test policies specific to the test case execution of Certificate-based Client Authentication functional block:

- DUT shall give the Advanced Security Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.
- The DUT shall support upload of CRLs.
- The following tests are performed about CRL management
  - The ONVIF Client can upload a CRL to the DUT.
  - A CRL from the storage on the DUT is correctly returned to the ONVIF client.
  - All CRLs in the storage on the DUT are correctly returned to the ONVIF client.
  - The ONVIF Client can delete a CRL from the storage on the DUT.
- The following tests are performed about certification path validation policy management
  - The ONVIF Client can create a certification path validation policy on the DUT.
  - A certification path validation policy from the storage on the DUT is correctly returned to the ONVIF client.



- All certification path validation policies in the storage on the DUT are correctly returned to the ONVIF client.
- The ONVIF Client can delete a certification path validation policy from the storage on the DUT.

Please, refer to Section 4.7 for Certificate-based Client Authentication Test Cases.



## **4 Advanced Security Test Cases**

### **4.1 Keystore**

#### **4.1.1 Create RSA Key Pair, status through polling**

**Test Label:** Create RSA Key Pair - Status Using Polling

**Test Case ID:** ADVANCED\_SECURITY-1-1-1

**ONVIF Core Specification Coverage:** Advanced Security, Keystore – Key Management

**Command Under Test:** CreateRSAKeyPair, GetKeyStatus

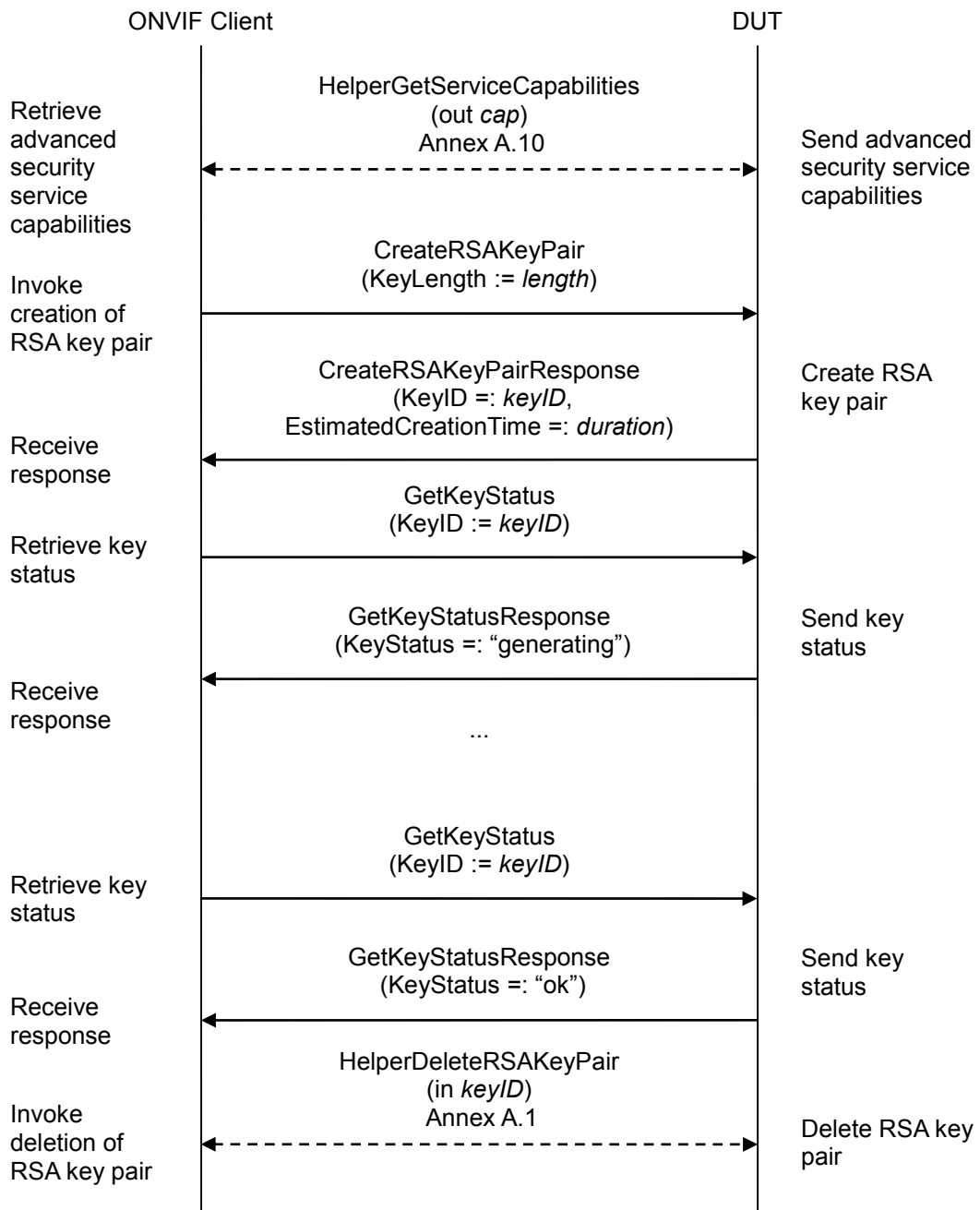
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test RSA key pair generation with key status retrieval through polling.

**Pre-requisite:** Advanced Security Service is received from the DUT. On-board RSA key pair generation is supported by the DUT as indicated by the RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional RSA key pair.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start ONVIF Client.
2. Start DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.10.



4. For each key length *keyLength* in the **RSASKeyLengths** capability contained in *cap.KeystoreCapabilities* repeat the following steps:
  - 4.1. ONVIF Client invokes **CreateRSAKeyPair** with parameter
    - KeyLength := *keyLength*
  - 4.2. The DUT responds with **CreateRSAKeyPairResponse** message with parameters
    - KeyID =: *keyID*
    - EstimatedCreationTime =: *duration*
  - 4.3. Until *timeout1* expires repeat the following steps:
    - 4.3.1. ONVIF Client waits for time *duration*.
    - 4.3.2. ONVIF Client invokes **GetKeyStatus** with parameters
      - KeyID := *keyID*
    - 4.3.3. The DUT responds with **GetKeyStatusResponse** message with parameters
      - KeyStatus =: *keyStatus*
    - 4.3.4. If *keyStatus* is equal to “ok”, go to the step 4.5.
    - 4.3.5. If *keyStatus* is equal to “corrupt”, FAIL the test, delete the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.1 to restore DUT configuration, and skip other steps.
  - 4.4. If *timeout1* timeout expires for step 4.3 and the last *keyStatus* is other than “ok”, FAIL the test, delete the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.1 to restore DUT configuration and skip other steps.
  - 4.5. ONVIF Client deletes the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.1 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **CreateRSAKeyPairResponse** message(s).

The DUT did not send **GetKeyStatusResponse** message(s).

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.





#### **4.1.2 Create RSA Key Pair, status through event**

**Test Label:** Create RSA Key Pair – Status Using Event

**Test Case ID:** ADVANCED\_SECURITY-1-1-2

**ONVIF Core Specification Coverage:** Advanced Security, Keystore – Key Management

**Command Under Test:** CreateRSAKeyPair

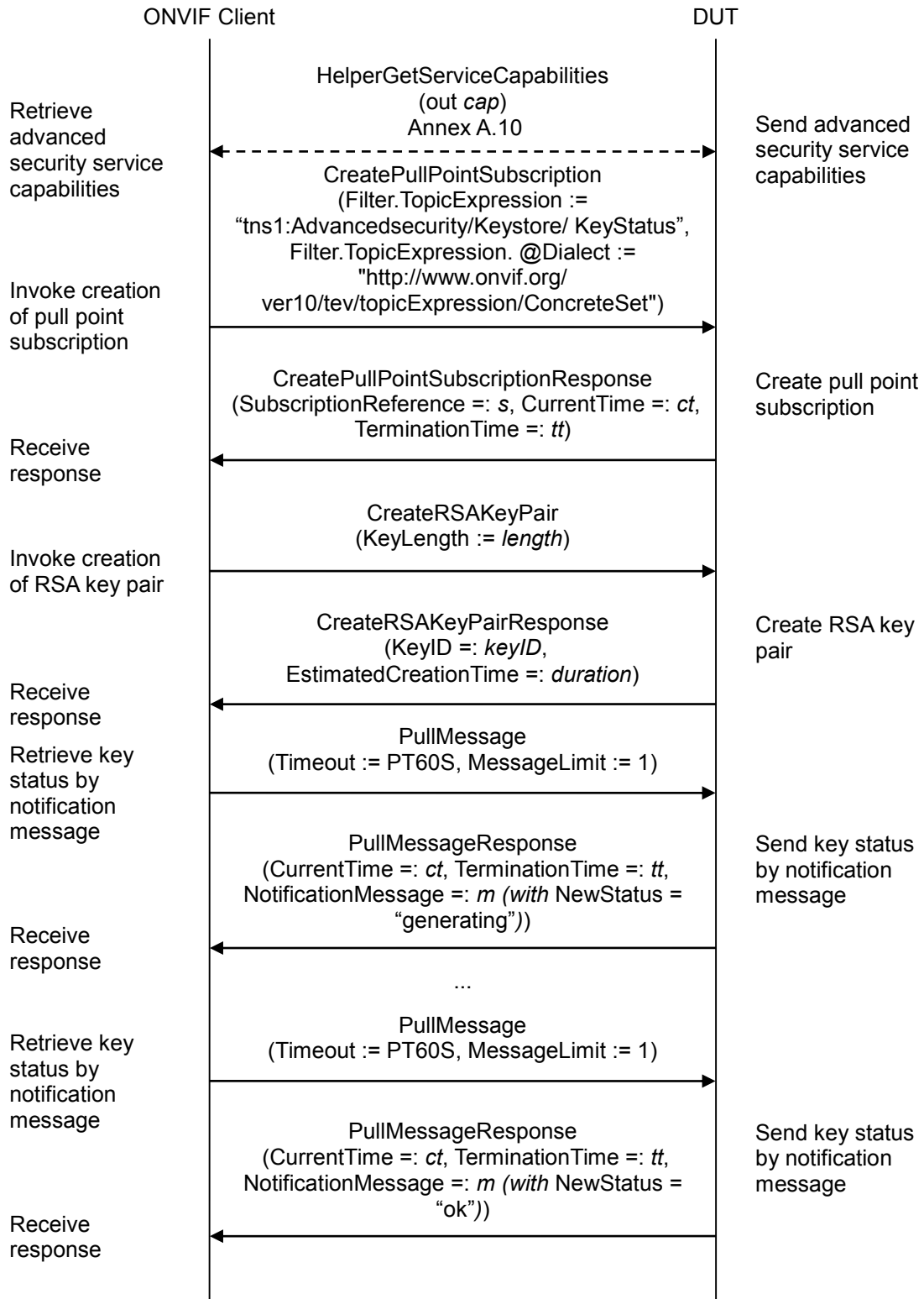
**WSDL Reference:** advancedsecurity.wsdl and event.wsdl

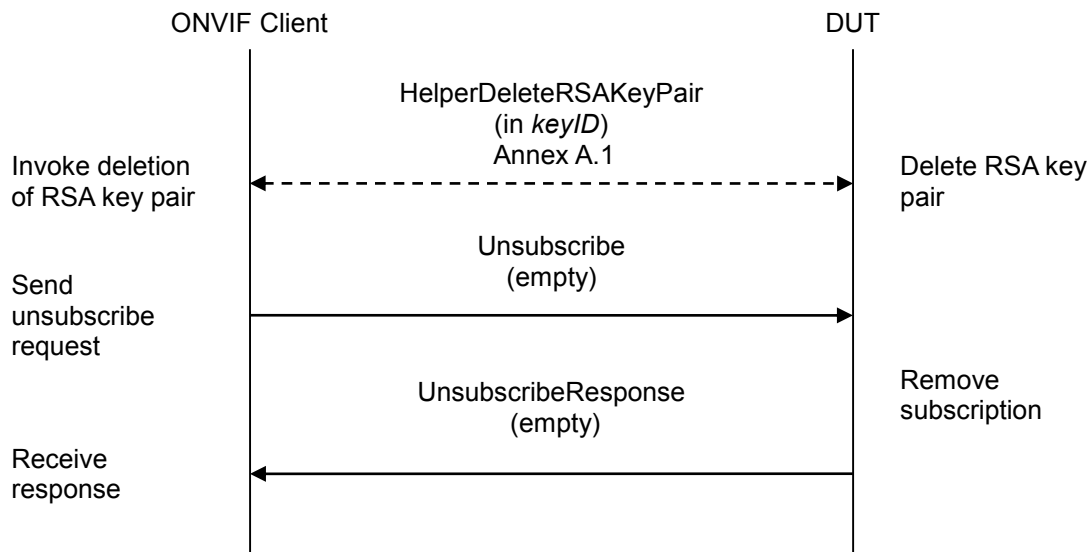
**Test Purpose:** To test RSA key pair generation with key status retrieval through events.

**Pre-requisite:** Advanced Security Service is received from the DUT. Event Service was received from the DUT. On-board RSA key pair generation is supported by the DUT as indicated by the RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional RSA key pair.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by the following the procedure mentioned in Annex A.10.
4. ONVIF Client invokes **CreatePullPointSubscription** with parameters
  - Filter.TopicExpression := "tns1:Advancedsecurity/Keystore/KeyStatus"
  - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
5. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
  - SubscriptionReference =: *s*
  - CurrentTime =: *ct*
  - TerminationTime =: *tt*
6. For each key length *keyLength* in the RSAKeyLengths capability contained in *cap.KeystoreCapabilities* repeat the following steps:
  - 6.1. ONVIF Client invokes **CreateRSAKeyPair** with parameter
    - KeyLength := *keyLength*
  - 6.2. The DUT responds with **CreateRSAKeyPairResponse** message with parameters
    - KeyID =: *keyID*
    - EstimatedCreationTime =: *duration*



6.3. Until  $timeout1 + 2 * duration$  timeout expires repeat the following steps:

6.3.1. ONVIF Client waits for time  $t := \min\{(tt-ct)/2, 1 \text{ second}\}$ .

6.3.2. ONVIF Client invokes **PullMessages** to the subscription endpoint  $s$  with parameters

- Timeout := PT60S
- MessageLimit := 1

6.3.3. The DUT responds with **PullMessagesResponse** message with parameters

- CurrentTime =:  $ct$
- TerminationTime =:  $tt$
- NotificationMessage =:  $m$

6.3.4. If  $m$  is not null and the KeyID source simple item in  $m$  is equal to  $keyID$  and the NewStatus data simple item in  $m$  is equal to "ok", go to the step 6.5.

6.3.5. If  $m$  is not null and the KeyID source simple item in  $m$  is equal to  $keyID$  and the NewStatus data simple item in  $m$  is equal to "corrupt", FAIL the test, delete the RSA key pair (in  $keyID$ ) by following the procedure mentioned in Annex A.1 to restore DUT configuration and go to the step 7.

6.4. If  $timeout1$  timeout expires for step 6.3 without Notification with KeyID source simple item equal to  $keyID$  and the NewStatus data simple item equal to "ok", FAIL the test, delete the RSA key pair (in  $keyID$ ) by following the procedure mentioned in Annex A.1 to restore DUT configuration and go to the step 7.

6.5. ONVIF Client deletes the RSA key pair (in  $keyID$ ) by following the procedure mentioned in Annex A.1 to restore DUT configuration.

7. ONVIF Client sends an **Unsubscribe** to the subscription endpoint  $s$ .

8. The DUT responds with **UnsubscribeResponse** message.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **CreatePullPointSubscriptionResponse** message.

The DUT did not send **CreateRSAKeyPairResponse** message(s).

The DUT did not send **PullMessagesResponse** message(s).

The DUT did not send **UnsubscribeResponse** message.

**Note:**  $timeout1$  will be taken from Operation Delay field of ONVIF Device Test Tool.



**4.1.3 Check private Key status for an RSA private key**

**Test Label:** Check RSA key pair private key status

**Test Case ID:** ADVANCED\_SECURITY-1-1-3

**ONVIF Core Specification Coverage:** Advanced Security, Keystore – Key Management

**Command Under Test:** GetAllKeys

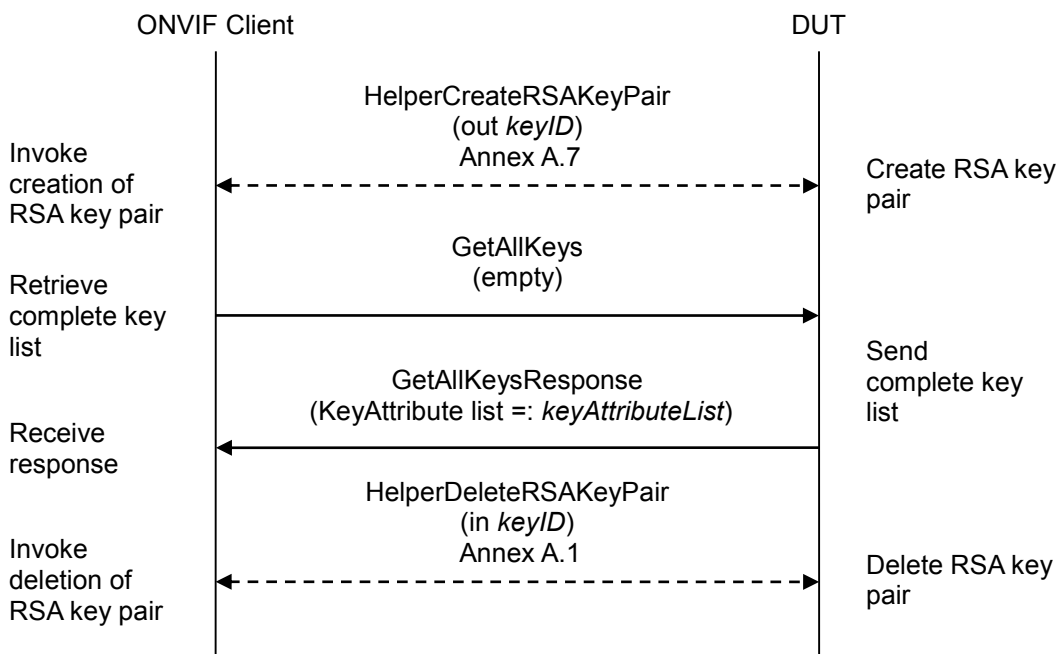
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test whether the private key status is correctly returned for a key pair with private key.

**Pre-requisite:** Advanced Security Service is received from the DUT. On-board RSA key pair generation is supported by the DUT as indicated by the RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional RSA key pair.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates an RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.7.



4. ONVIF Client invokes **GetAllKeys**.
5. The DUT responds with a **GetAllKeysResponse** message with parameters
  - KeyAttribute list =: *keyAttributeList*
6. If *keyAttributeList*[KeyID = *keyID*].hasPrivateKey is not equal to true, FAIL the test and go to the next step.
7. ONVIF Client deletes the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.1 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetAllKeysResponse** message.



**4.1.4 Get all keys**

**Test Label:** Get All Keys

**Test Case ID:** ADVANCED\_SECURITY-1-1-4

**ONVIF Core Specification Coverage:** Advanced Security, Keystore – Key Management

**Command Under Test:** GetAllKeys

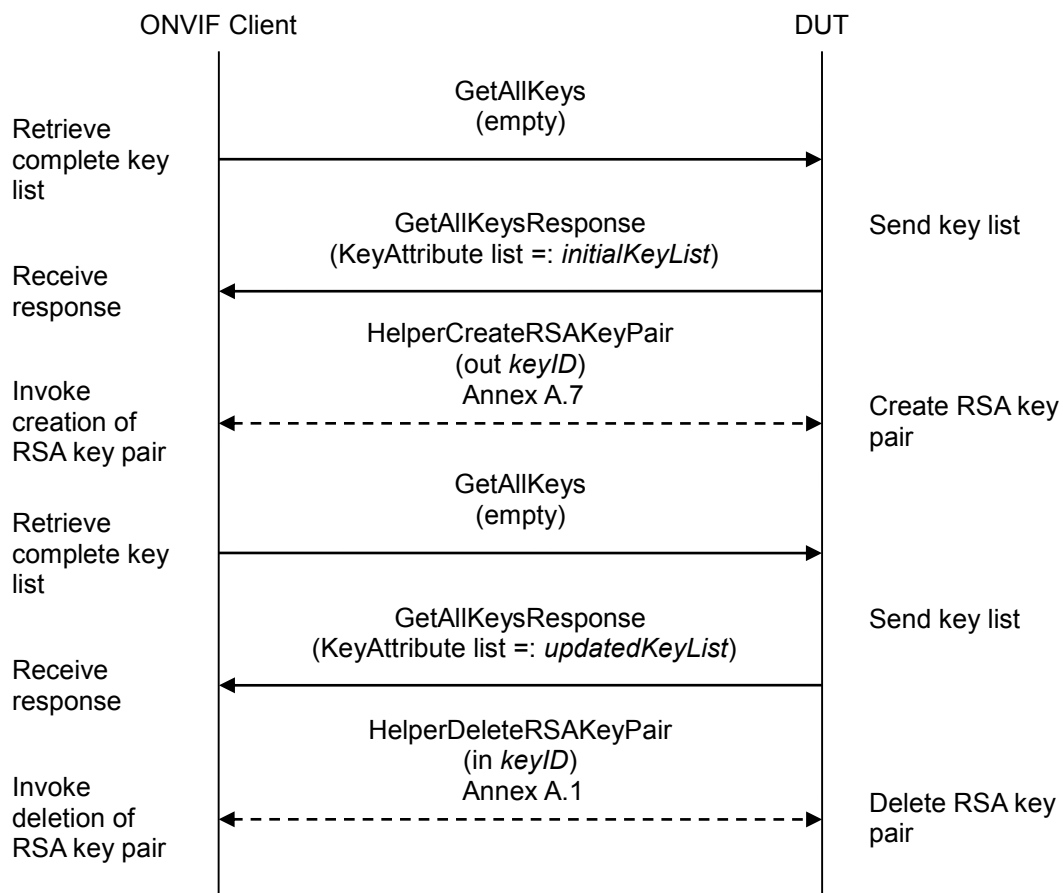
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test listing of RSA key pairs and appearing of new created RSA key pairs in the list.

**Pre-requisite:** Advanced Security Service is received from the DUT. On-board RSA key pair generation is supported by the DUT as indicated by the RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional RSA key pair.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**



1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAllKeys**.
4. The DUT responds with a **GetAllKeysResponse** message with parameters
  - KeyAttribute list =: *initialKeyList*
5. ONVIF Client creates an RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.7.
6. ONVIF Client invokes **GetAllKeys**.
7. The DUT responds with a **GetAllKeysResponse** message with parameters
  - KeyAttribute list =: *updatedKeyList*
8. If *updatedKeyList* does not contain *keyID* and all keys from *initialKeyList*, FAIL the test, and go to the step 10.
9. If *updatedCertificateList* contains keys other than *keyID* or keys from *initialCertificateList*, FAIL the test, and go to the step 10.
10. ONVIF Client deletes the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.1 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetAllKeysResponse** message(s).

**Note:** The DUT may return an empty list at step 4.





#### **4.1.5 Delete key**

**Test Label:** Delete Key

**Test Case ID:** ADVANCED\_SECURITY-1-1-6

**ONVIF Core Specification Coverage:** Advanced Security, Keystore – Key Management

**Command Under Test:** DeleteKey

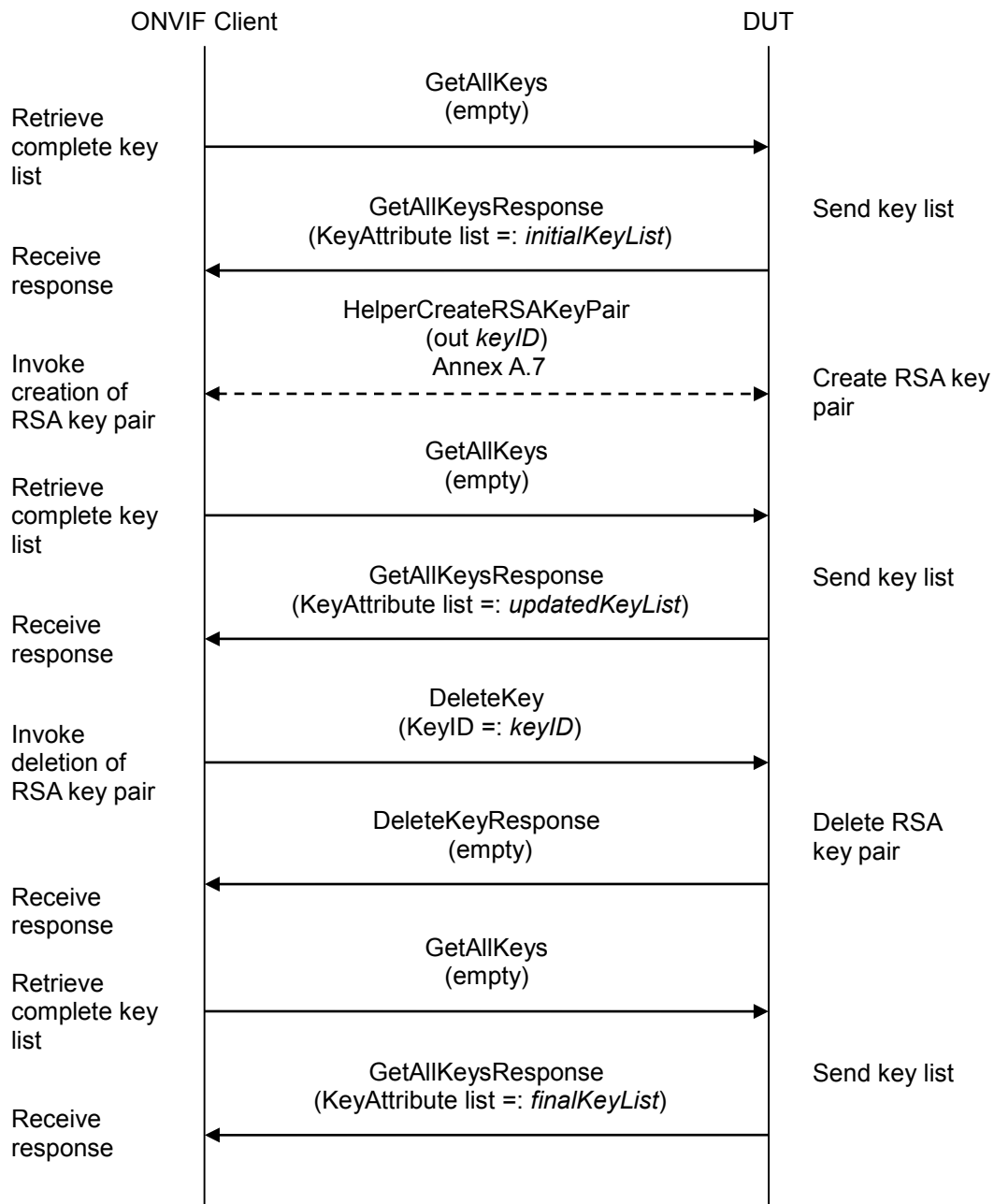
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test deletion of RSA key pairs

**Pre-requisite:** Advanced Security Service is received from the DUT. On-board RSA key pair generation is supported by the DUT as indicated by the RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional RSA key pair.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAllKeys**.
4. The DUT responds with a **GetAllKeysResponse** message with parameters



- KeyAttribute list =: *initialKeyList*
5. ONVIF Client creates an RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.7.
  6. ONVIF Client invokes **GetAllKeys**.
  7. The DUT responds with a **GetAllKeysResponse** message with parameters
    - KeyAttribute list =: *updatedKeyList*
  8. If *updatedKeyList* does not contain *keyID* and all keys from *initialKeyList*, FAIL the test, delete the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.1 to restore DUT configuration, and skip other steps.
  9. If *updatedKeyList* contains keys other than *keyID* or keys from *initialKeyList*, FAIL the test, and delete the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.1 to restore DUT configuration, and skip other steps.
  10. ONVIF Client invokes **DeleteKey** with parameters
    - KeyID =: *keyID*
  11. The DUT responds with a **DeleteKeyResponse** message.
  12. ONVIF Client invokes **GetAllKeys**.
  13. The DUT responds with a **GetAllKeysResponse** message with parameters
    - KeyAttribute list =: *finalKeyList*
  14. If *finalKeyList* is not equal *initialKeyList*, FAIL the test.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **DeleteKeyResponse** message.

The DUT did not send **GetAllKeysResponse** message(s).

**Note:** The DUT may return an empty list at step 4.



## 4.2 Certificate Management

### 4.2.1 Create PKCS#10 certification requests

**Test Label:** Create PKCS#10

**Test Case ID:** ADVANCED\_SECURITY-2-1-1

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** CreatePKCS10CSR

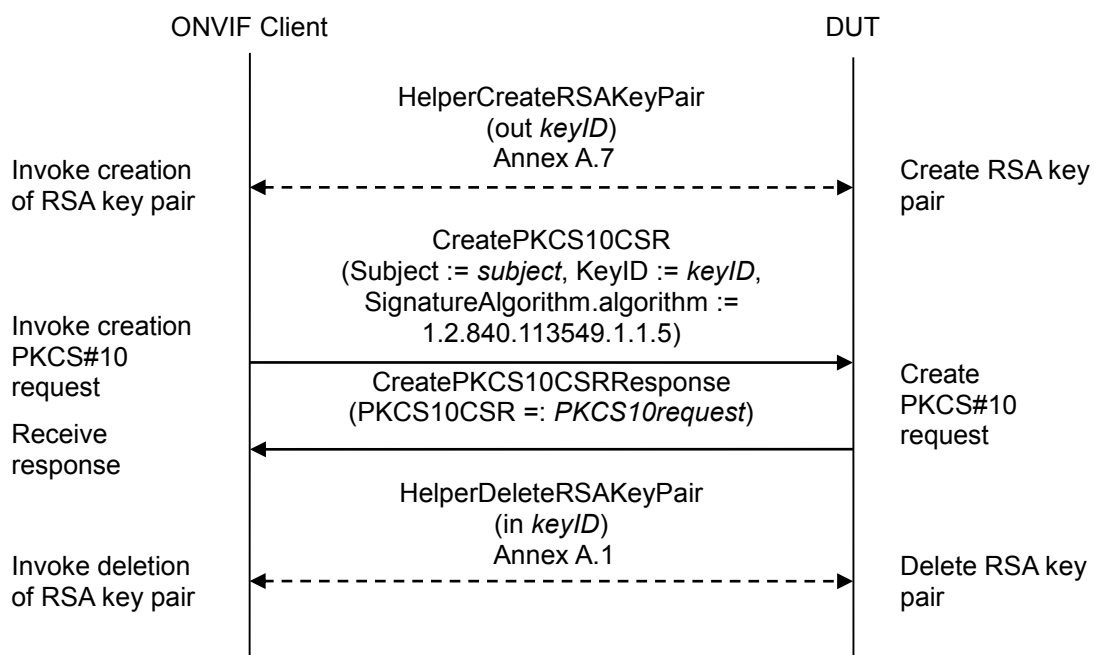
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the creation of a PKCS#10 certification requests.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional RSA key pair.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.



3. ONVIF Client creates an RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.7.
4. ONVIF Client invokes **CreatePKCS10CSR** with parameter
  - Subject := *subject* (see Annex A.2)
  - KeyID := *keyID*
  - CSRAttribute skipped
  - SignatureAlgorithm.algorithm := 1.2.840.113549.1.1.5 (OID of SHA-1 with RSA Encryption algorithm)
5. The DUT responds with **CreatePKCS10CSRResponse** message with parameters
  - PKCS10CSR =: *PKCS10request*
6. ONVIF Client validates that *PKCS10request* is correctly DER encoded (see Annex A.19).
7. If *PKCS10request* is incorrectly DER encoded, FAIL the test and go to the step 10.
8. ONVIF Client validates that *PKCS10request* contains the correct subject equals to *subject*.
9. If *PKCS10request* contains a wrong subject, FAIL the test and go to the step 10.
10. ONVIF Client deletes the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.1 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **CreatePKCS10CSRResponse** message.



**4.2.2 Create self-signed certificate**

**Test Label:** Create Self-Signed Certificate

**Test Case ID:** ADVANCED\_SECURITY-2-1-2

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** CreateSelfSignedCertificate

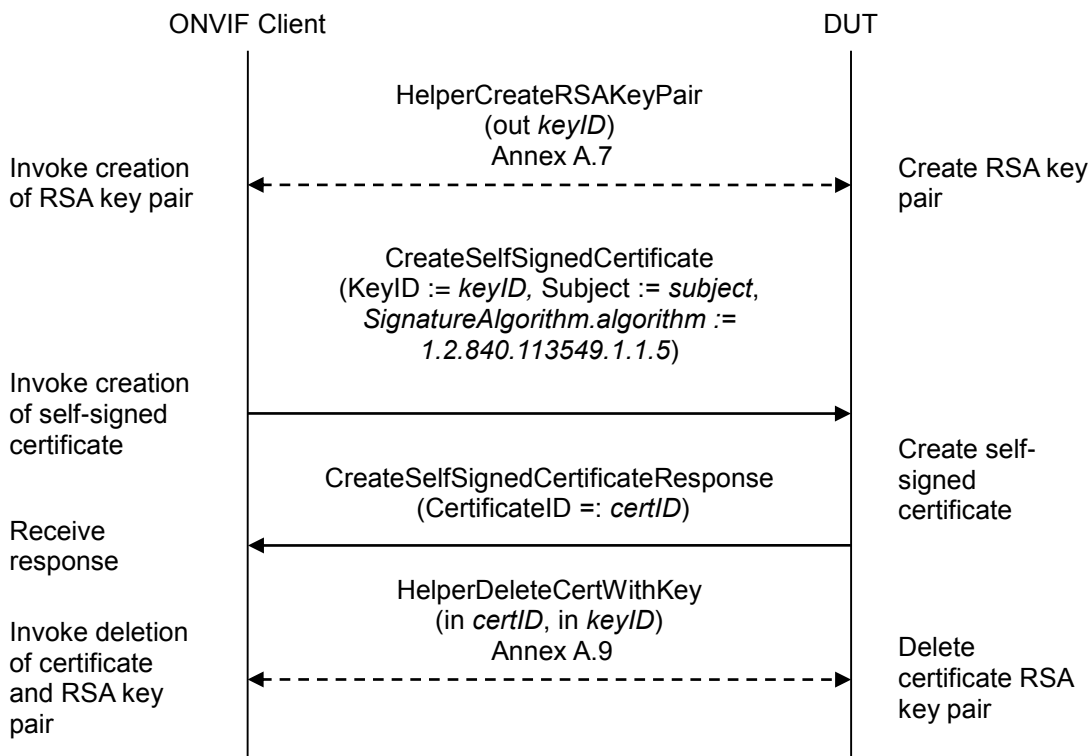
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the creation of self-signed certificates.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.



3. ONVIF Client creates an RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.7.
4. ONVIF Client invokes **CreateSelfSignedCertificate** with parameters
  - X509Version skipped
  - KeyID := *keyID*
  - Subject := *subject* (see Annex A.2)
  - Alias skipped
  - notValidBefore skipped
  - notValidAfter skipped
  - SignatureAlgorithm.algorithm := 1.2.840.113549.1.1.5 (OID of SHA-1 with RSA Encryption algorithm)
  - SignatureAlgorithm.parameters skipped
  - SignatureAlgorithm.anyParameters skipped
  - Extension skipped
5. The DUT responds with a **CreateSelfSignedCertificateResponse** message with parameters
  - CertificateID =: *certID*
6. ONVIF Client deletes the self-signed certificate (in *certID*) and related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.9 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **CreateSelfSignedCertificateResponse** message.



**4.2.3 Upload certificate – Keystore contains private key**

**Test Label:** Upload certificate – Keystore contains private key

**Test Case ID:** ADVANCED\_SECURITY-2-1-3

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** UploadCertificate

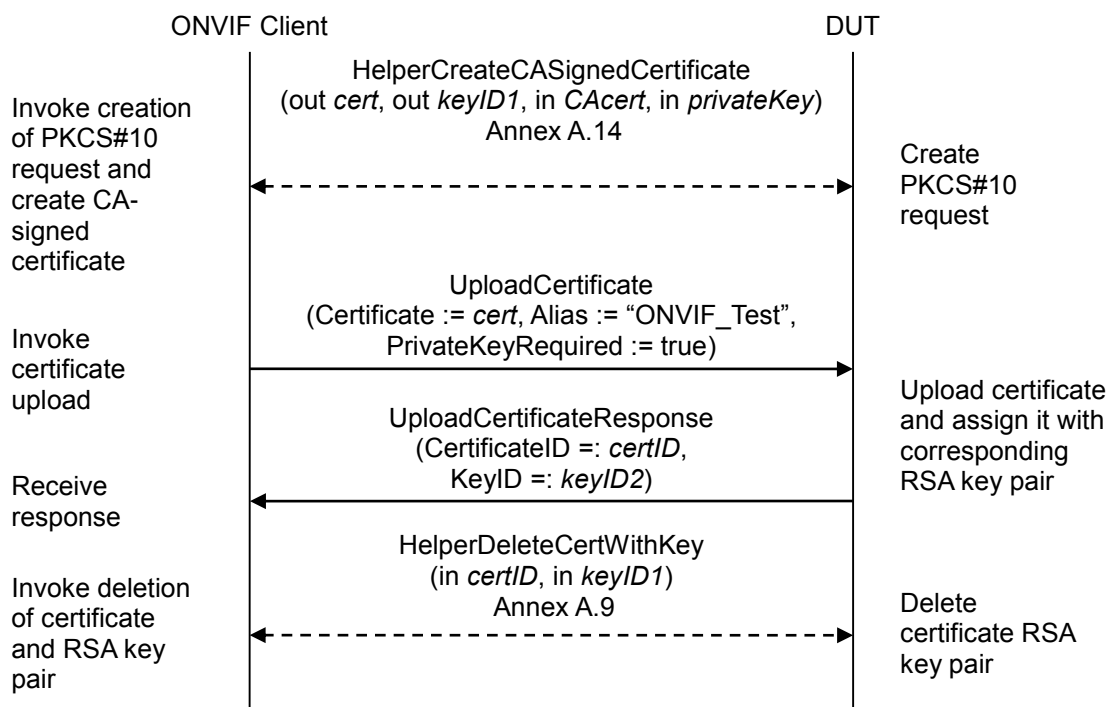
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the upload of a certificate in case the keystore in the DUT contains a private key for the public key in the certificate.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.





2. Start the DUT.
3. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding private key (out *privateKey*) by following the procedure described in Annex A.4.
4. ONVIF Client creates a certificate (out *cert*) from the PKCS#10 request with RSA key pair (out *keyID1*) and associated CA certificate (in *CAcert*) and a corresponding private key (in *privateKey*) by following the procedure described in Annex A.14.
5. ONVIF Client invokes **UploadCertificate** with parameters
  - Certificate := *cert*
  - Alias := "ONVIF\_Test"
  - PrivateKeyRequired := true
6. The DUT responds with a **UploadCertificateResponse** message with parameters
  - CertificateID =: *certID*
  - KeyID =: *keyID2*
7. ONVIF Client validates that *keyID2* equal to *keyID1*.
8. If *keyID2* is not equal to *keyID1*, FAIL the test and go to the step 9.
9. ONVIF Client deletes the CA certificate (in *certID*) and related RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.9 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **UploadCertificateResponse** message.



#### 4.2.4 Upload certificate – Keystore contains private key (negative test)

**Test Label:** Upload Certificate – Keystore Contains Private Key (Negative Test)

**Test Case ID:** ADVANCED\_SECURITY-2-1-4

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** UploadCertificate

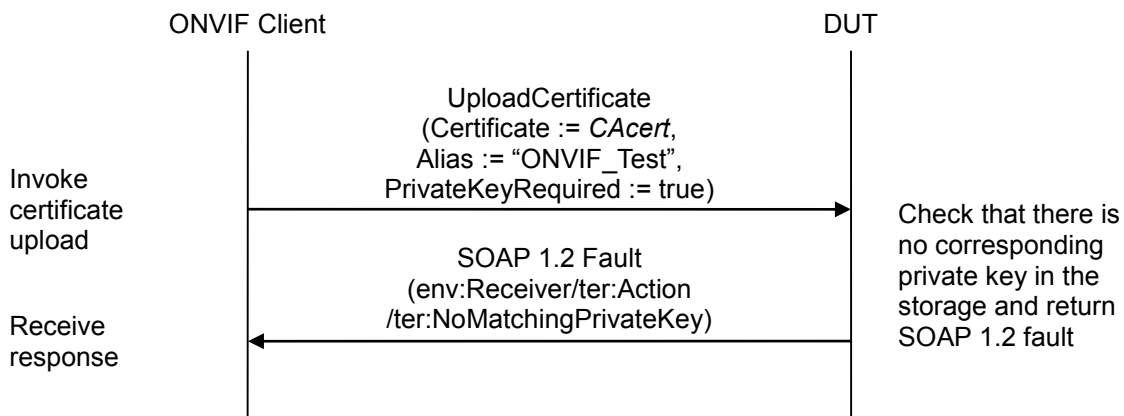
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the upload of a certificate in case the keystore in the DUT does not contain a private key for the public key in the certificate (negative test).

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. The DUT shall have enough free storage capacity for one additional certificate. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



#### Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding private key (out *privateKey*) by following the procedure described in Annex A.4.
4. ONVIF Client invokes **UploadCertificate** with parameters
  - Certificate := *CAcert*
  - Alias := "ONVIF\_Test"
  - PrivateKeyRequired := true



5. The DUT returns env:Receiver/ter:Action/ter:NoMatchingPrivateKey SOAP 1.2 fault.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send the env:Receiver/ter:Action/ter:NoMatchingPrivateKey SOAP 1.2 fault message.



**4.2.5 Upload certificate – Keystore does not contain private key**

**Test Label:** Upload certificate – Keystore does not contain private key

**Test Case ID:** ADVANCED\_SECURITY-2-1-5

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** UploadCertificate

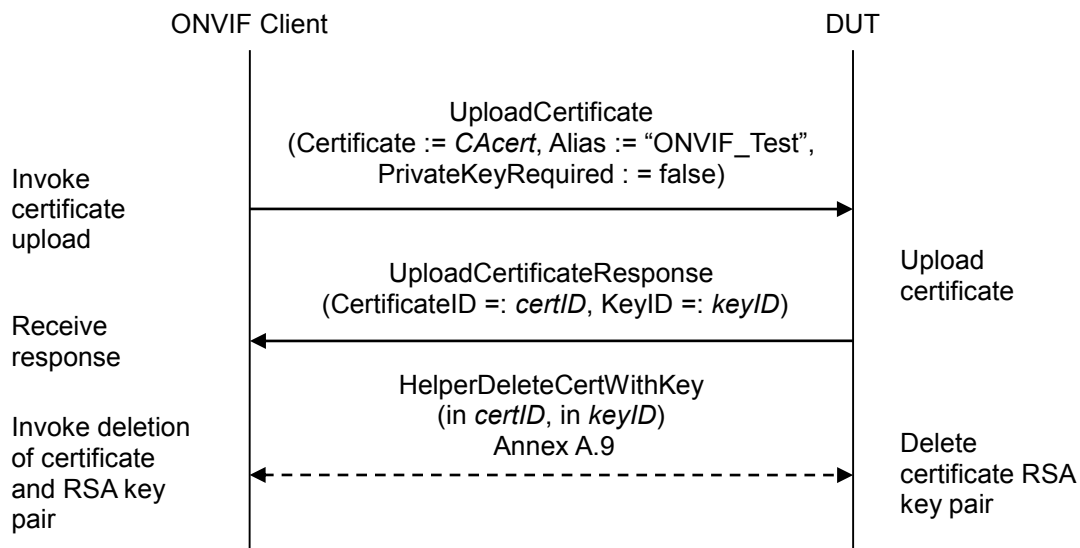
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the upload of a certificate in case the keystore in the DUT does not contain a private key for the public key in the certificate.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding private key (out *privateKey*) by following the procedure described in Annex A.4.
4. ONVIF Client invokes **UploadCertificate** with parameters



- Certificate := *CAcert*
  - Alias := "ONVIF\_Test"
  - PrivateKeyRequired : = false
5. The DUT responds with a **UploadCertificateResponse** message with parameters
- CertificateID =: *certID*
  - KeyID =: *keyID*
6. ONVIF Client deletes the CA certificate (in *certID*) and related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.9 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **UploadCertificateResponse** message.



**4.2.6 Get certificate – self-signed**

**Test Label:** Get Certificate - Self-Signed

**Test Case ID:** ADVANCED\_SECURITY-2-1-6

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** GetCertificate

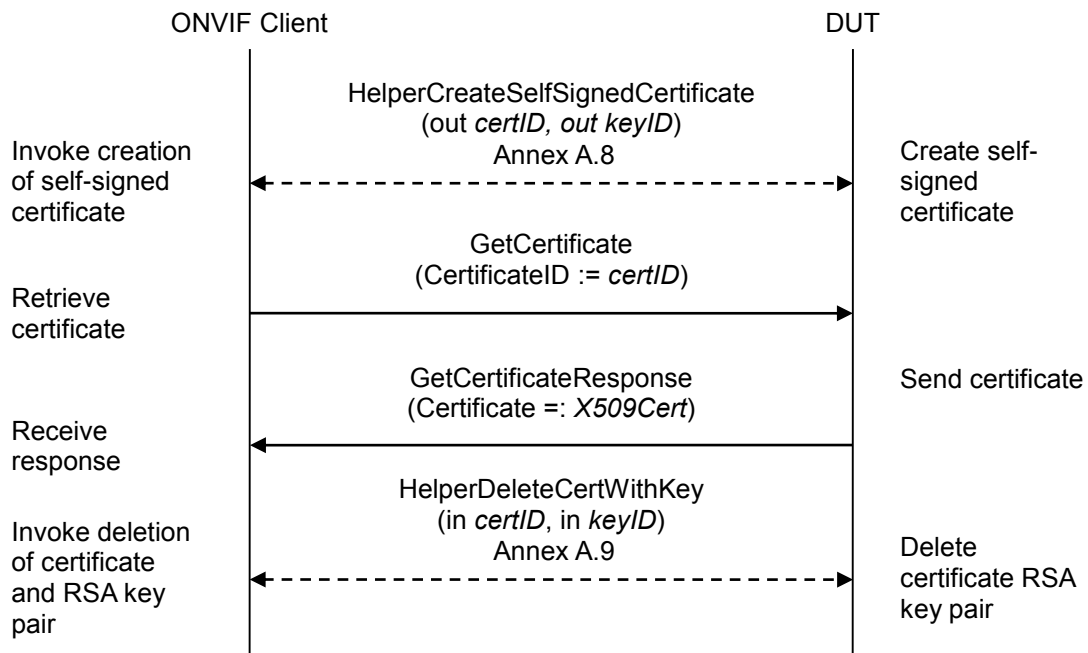
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the retrieval of a self-signed certificate.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a self-signed certificate (out *certID*) and related RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.8.



4. ONVIF Client invokes **GetCertificate** message with parameters
  - CertificateID := *certID*
5. The DUT responds with a **GetCertificateResponse** message with parameters
  - Certificate =: *X509Cert*
6. ONVIF Client validates that *X509Cert.CertificateContent* is correctly DER encoded (see Annex A.19).
7. If *X509Cert.CertificateContent* is incorrectly DER encoded, FAIL the test and go to the step 10.
8. ONVIF Client validates that *X509Cert.CertificateContent* contains the correct subject equals to subject defined in Annex A.2.
9. If *X509Cert.CertificateContent* contains wrong subject, FAIL the test and go to the step 10.
10. ONVIF Client deletes the self-signed certificate (in *certID*) and related the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.9 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetCertificateResponse** message.



**4.2.7 Get certificate – CA**

**Test Label:** Get Certificate – CA

**Test Case ID:** ADVANCED\_SECURITY-2-1-7

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** GetCertificate

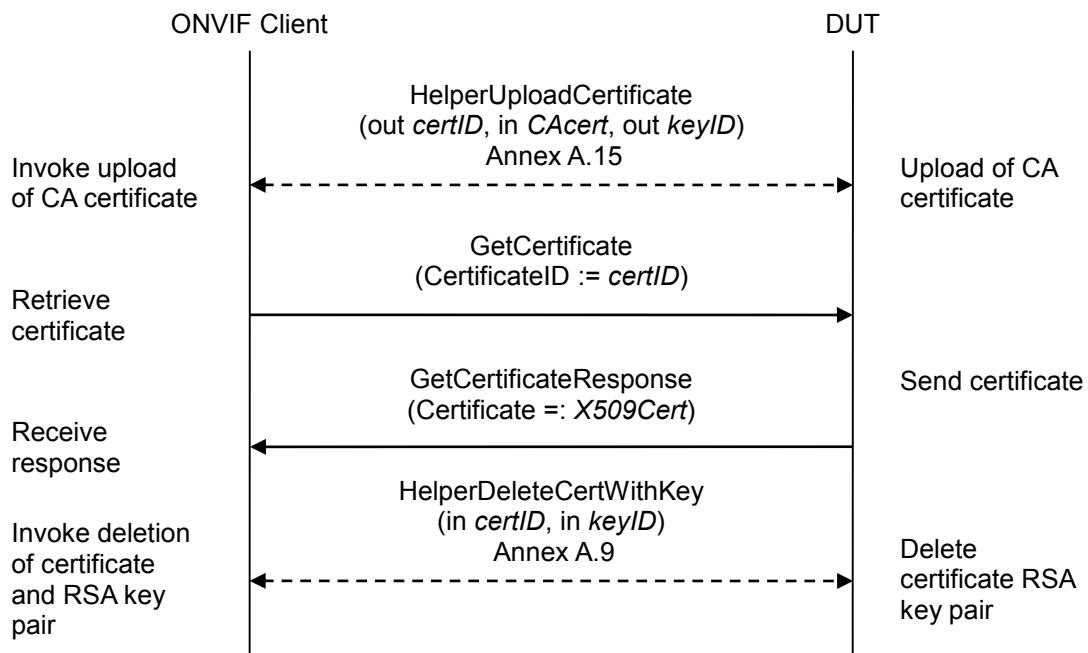
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the retrieval of a CA certificate.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding private key (out *privateKey*) by following the procedure described in Annex A.4.





4. ONVIF Client uploads a CA certificate (out *certID*, in *CAcert*) and new RSA key pair with the public key from the CA certificate (out *keyID*) by following the procedure described in Annex A.15.
5. ONVIF Client invokes **GetCertificate** message with parameters
  - CertificateID := *certID*
6. The DUT responds with a **GetCertificateResponse** message with parameters
  - Certificate =: *X509Cert*
7. ONVIF Client validates that *X509Cert.CertificateContent* is correctly DER encoded (see Annex A.19).
8. If *X509Cert.CertificateContent* is incorrectly DER encoded, FAIL the test and go to the step 10.
9. If *X509Cert.CertificateContent* contains wrong subject, FAIL the test and go to the step 10.
10. ONVIF Client deletes the CA certificate (in *certID*) and related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.9 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetCertificateResponse** message.



**4.2.8 Get all certificates – self signed**

**Test Label:** Get All Certificates - Self-Signed

**Test Case ID:** ADVANCED\_SECURITY-2-1-8

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** GetAllCertificates

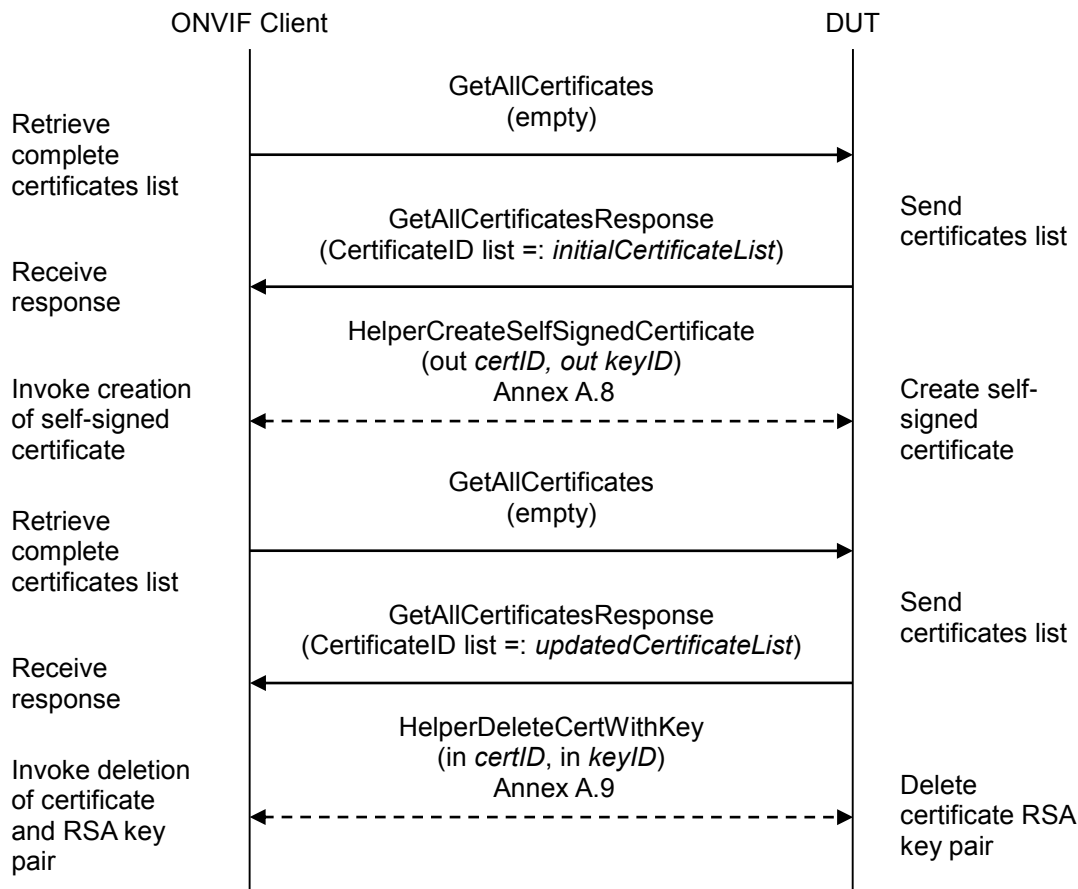
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the retrieval of all certificates tested with self-signed certificates.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAllCertificates**.
4. The DUT responds with a **GetAllCertificatesResponse** message with parameters
  - CertificateID list =: *initialCertificateList*
5. ONVIF Client creates a self-signed certificate (out *certID*) and related RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.8.
6. ONVIF Client invokes **GetAllCertificates**.
7. The DUT responds with a **GetAllCertificatesResponse** message with parameters
  - CertificateID list =: *updatedCertificateList*
8. If *updatedCertificateList* does not contain *certID* and all certificates from *initialCertificateList*, FAIL the test, and go to the step 10.
9. If *updatedCertificateList* contains certificates other than *certID* or certificates from *initialCertificateList*, FAIL the test, and go to the step 10.
10. ONVIF Client deletes the self-signed certificate (in *certID*) and related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.9 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetAllCertificatesResponse** message(s).

**Note:** The DUT may return an empty list at step 4.



**4.2.9 Get All Certificate – CA**

**Test Label:** Get All Certificate – CA

**Test Case ID:** ADVANCED\_SECURITY-2-1-9

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** GetAllCertificates

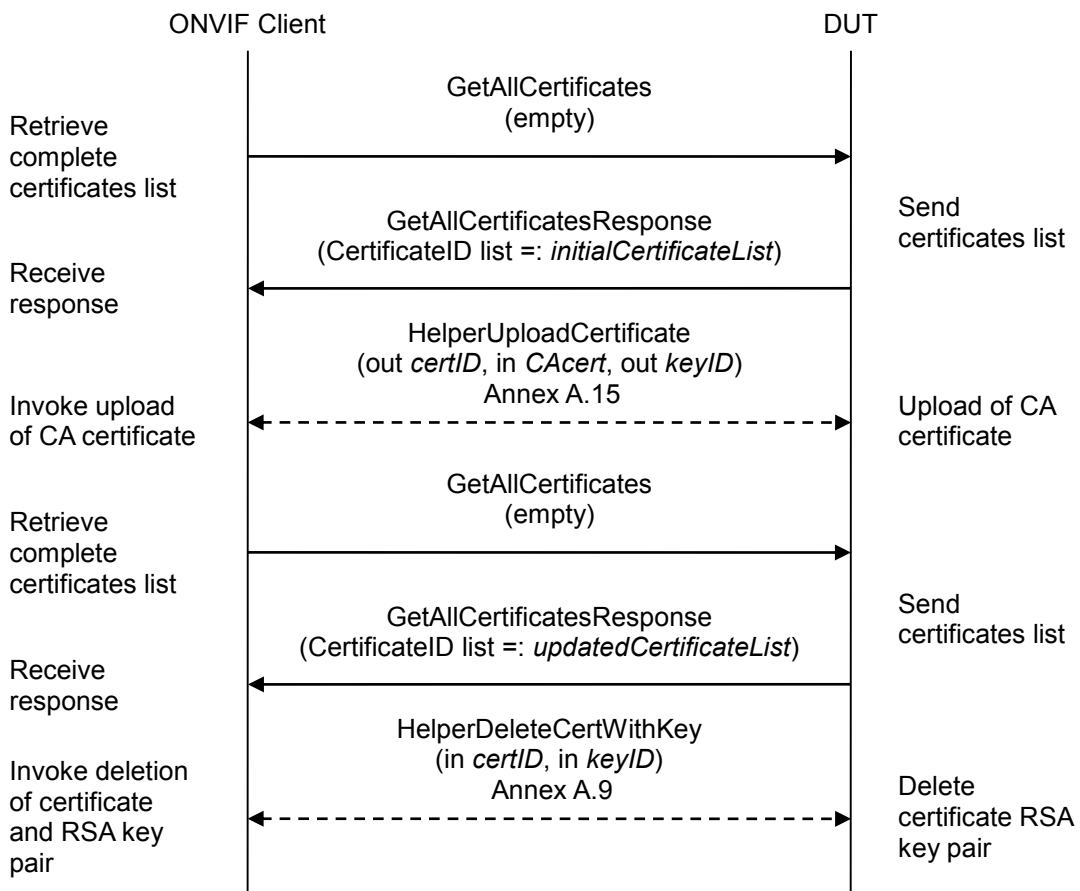
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the retrieval of all certificates tested with CA certificates.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAllCertificates**.
4. DUT responds with a **GetAllCertificatesResponse** message with parameters
  - CertificateID list =: *initialCertificateList*
5. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding private key (out *privateKey*) by following the procedure described in Annex A.4.
6. ONVIF Client uploads a CA certificate (out *certID*, in *CAcert*) and new RSA key pair with the public key from the CA certificate (out *keyID*) by following the procedure described in Annex A.15.
7. ONVIF Client invokes **GetAllCertificates**.
8. The DUT responds with a **GetAllCertificatesResponse** message with parameters
  - CertificateID list =: *updatedCertificateList*
9. If *updatedCertificateList* does not contain *certID* and all certificates from *initialCertificateList*, FAIL the test, and go to the step 10.
10. If *updatedCertificateList* contains certificates other than *certID* or certificates from *initialCertificateList*, FAIL the test, and go to the step 10.
11. ONVIF Client deletes the CA certificate (in *certID*) and related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.9 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetAllCertificatesResponse** message(s).

**Note:** The DUT may return an empty list at step 4.



**4.2.10 Delete Certificate – self signed**

**Test Label:** Delete Certificate - Self-Signed

**Test Case ID:** ADVANCED\_SECURITY-2-1-10

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** DeleteCertificate

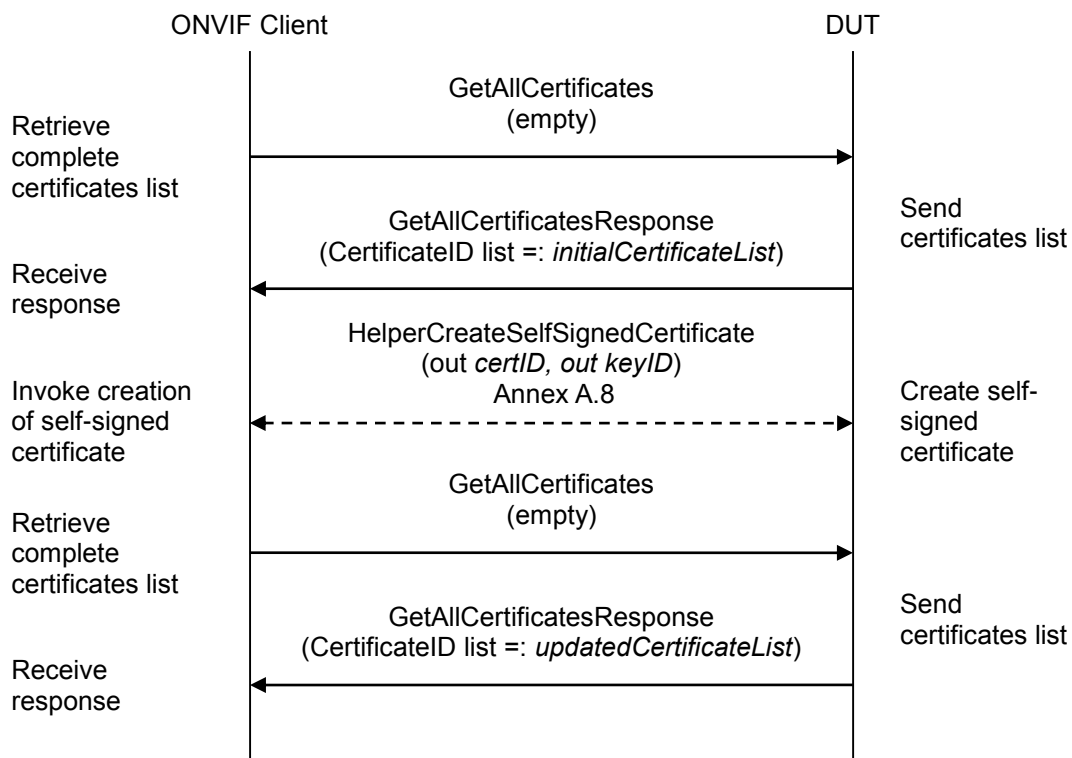
**WSDL Reference:** advancedsecurity.wsdl

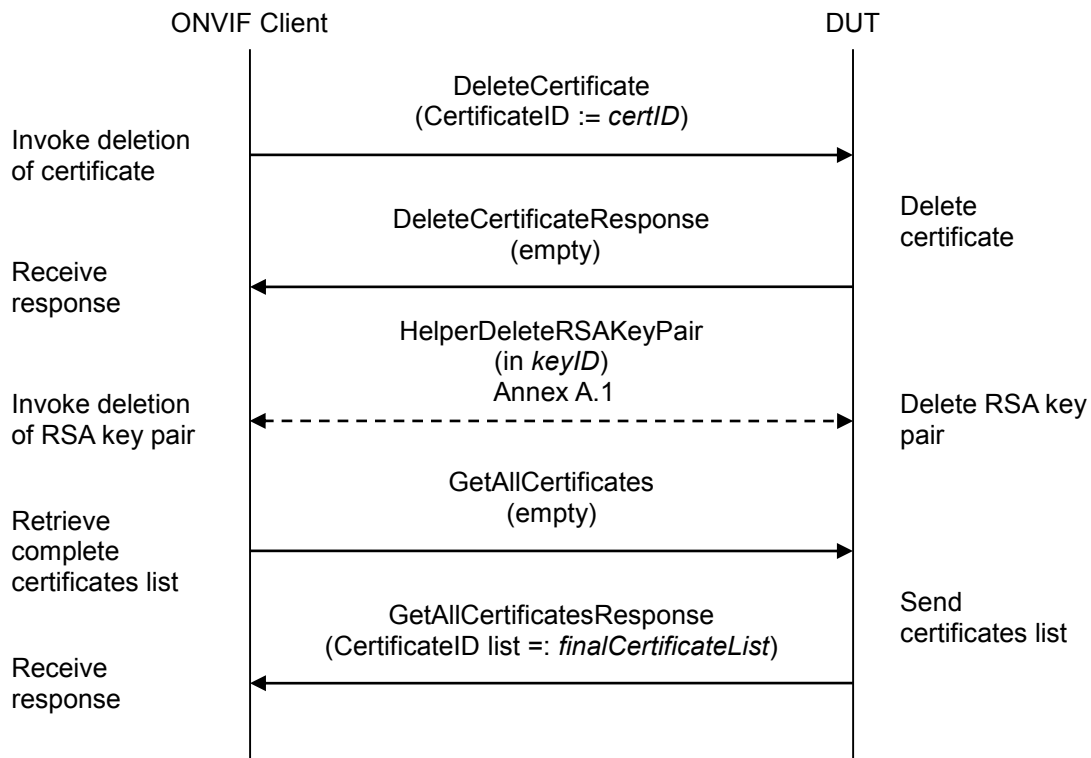
**Test Purpose:** To test the deletion of a certificate tested with self-signed certificates.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAllCertificates**.
4. The DUT responds with a **GetAllCertificatesResponse** message with parameters
  - CertificateID list =: *initialCertificateList*
5. ONVIF Client creates a self-signed certificate (out *certID*) and related RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.8.
6. ONVIF Client invokes **GetAllCertificates**.
7. The DUT responds with a **GetAllCertificatesResponse** message with parameters
  - CertificateID list =: *updatedCertificateList*
8. If *updatedCertificateList* does not contain *certID* and all certificates from *initialCertificateList*, FAIL the test, delete the self-signed certificate (in *certID*) and related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.9 to restore DUT configuration, and skip other steps.
9. If *updatedCertificateList* contains certificates other than *certID* or certificates from *initialCertificateList*, FAIL the test, delete the self-signed certificate (in *certID*) and related RSA



key pair (in *keyID*) by following the procedure mentioned in Annex A.9 to restore DUT configuration, and skip other steps.

10. ONVIF Client invokes **DeleteCertificate** with parameters

- CertificateID := *certID*

11. The DUT responds with a **DeleteCertificateResponse** message.

12. ONVIF Client deletes the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.1 to restore DUT configuration.

13. ONVIF Client invokes **GetAllCertificates**.

14. The DUT responds with a **GetAllCertificatesResponse** message with parameters

- CertificateID list := *finalCertificateList*

15. If *finalCertificateList* is not equal *initialCertificateList*, FAIL the test.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **DeleteCertificateResponse** message.

The DUT did not send **GetAllCertificatesResponse** message(s).

**Note:** The DUT may return an empty list at step 4.





**4.2.11 Delete Certificate – CA**

**Test Label:** Delete Certificate – CA

**Test Case ID:** ADVANCED\_SECURITY-2-1-11

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** DeleteCertificate

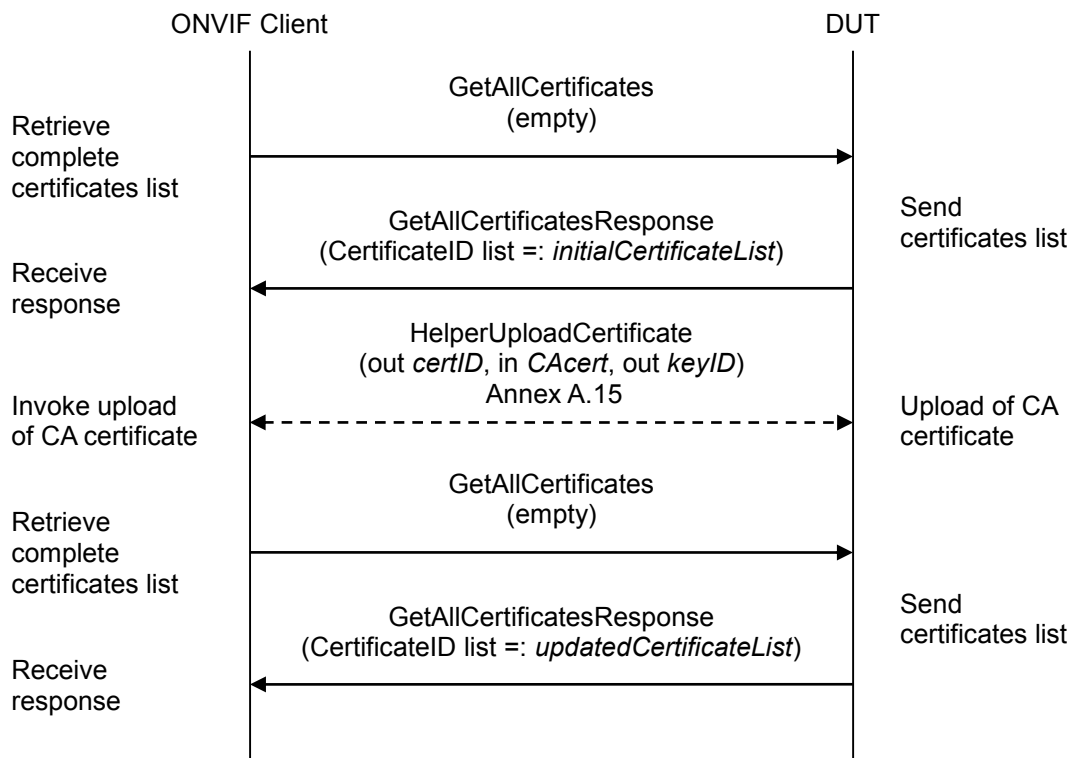
**WSDL Reference:** advancedsecurity.wsdl

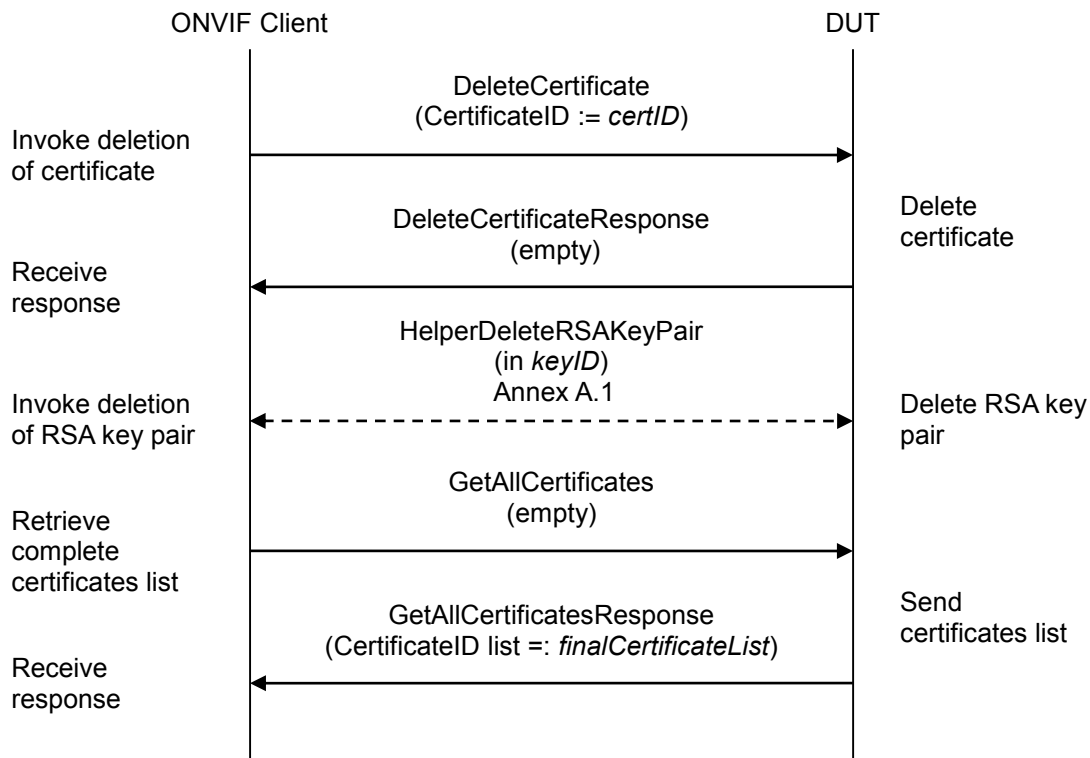
**Test Purpose:** To test the deletion of a certificate tested with CA certificates.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAllCertificates**.
4. The DUT responds with a **GetAllCertificatesResponse** message with parameters
  - CertificateID list =: *initialCertificateList*
5. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding private key (out *privateKey*) by following the procedure described in Annex A.4.
6. ONVIF Client uploads a CA certificate (out *certID*, in *CAcert*) and new RSA key pair with the public key from the CA certificate (out *keyID*) by following the procedure described in Annex A.15.
7. ONVIF Client invokes **GetAllCertificates**.
8. The DUT responds with a **GetAllCertificatesResponse** message with parameters
  - CertificateID list =: *updatedCertificateList*
9. If *updatedCertificateList* does not contain *certID* and all certificates from *initialCertificateList*, FAIL the test, delete the CA certificate (in *certID*) and related the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.9 to restore DUT configuration, and skip other



steps.

10. If *updatedCertificateList* contains certificates other than *certID* or certificates from *initialCertificateList*, FAIL the test, delete the CA certificate (in *certID*) and related the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.9 to restore DUT configuration, and skip other steps.
11. ONVIF Client invokes **DeleteCertificate** with parameters
  - CertificateID := *certID*
12. The DUT responds with a **DeleteCertificateResponse** message.
13. ONVIF Client deletes the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.1 to restore DUT configuration.
14. ONVIF Client invokes **GetAllCertificates**.
15. The DUT responds with a **GetAllCertificatesResponse** message with parameters
  - CertificateID list =: *finalCertificateList*
16. If *finalCertificateList* is not equal *initialCertificateList*, FAIL the test.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **DeleteCertificateResponse** message.

The DUT did not send **GetAllCertificatesResponse** message(s).

**Note:** The DUT may return an empty list at step 4.



**4.2.12 Create Certification Path – self-signed**

**Test Label:** Create Certification Path - Self-Signed

**Test Case ID:** ADVANCED\_SECURITY-2-1-12

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** CreateCertificationPath

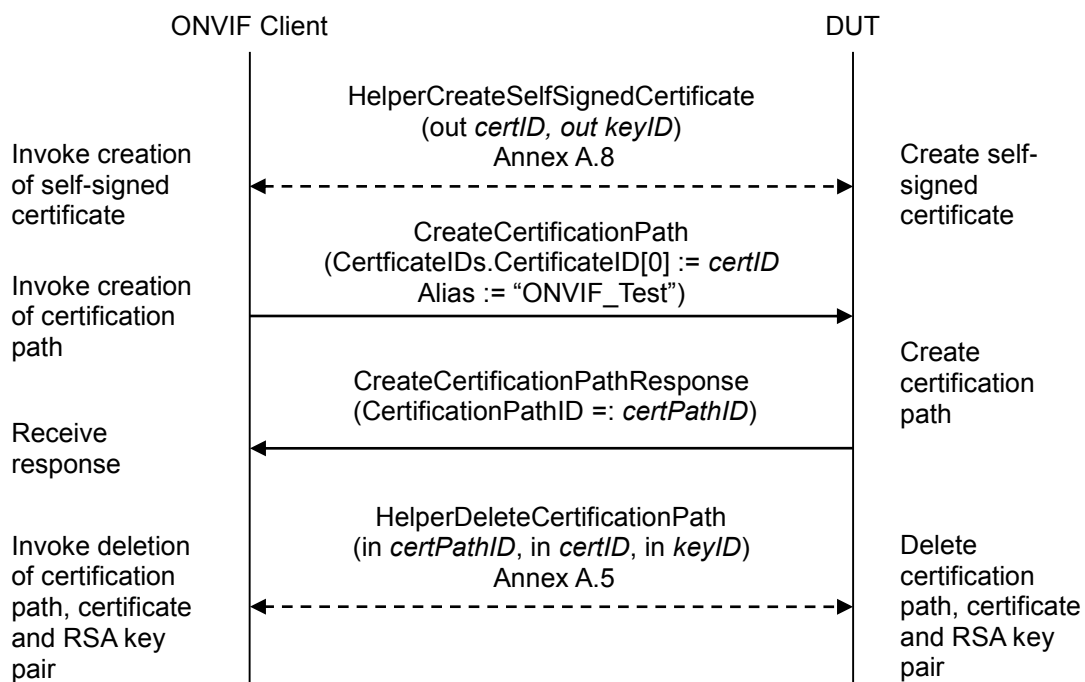
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the creation of a certification path containing a self-signed certificate.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certification path.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.



3. ONVIF Client creates a self-signed certificate (out *certID*) and related RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.8.
4. ONVIF Client invokes **CreateCertificationPath** with parameters
  - CertificateIDs.CertificateID[0] := *certID*
  - Alias := "ONVIF\_Test"
5. DUT responds with a **CreateCertificationPathResponse** message with parameters
  - CertificationPathID =: *certPathID*
6. ONVIF Client deletes the certification path (in *certPathID*) and related the self-signed certificate (in *certID*) and the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.5 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **CreateCertificationPathResponse** message.



#### **4.2.13 Create Certification Path – CA**

**Test Label:** Create Certification Path – CA

**Test Case ID:** ADVANCED\_SECURITY-2-1-13

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** CreateCertificationPath

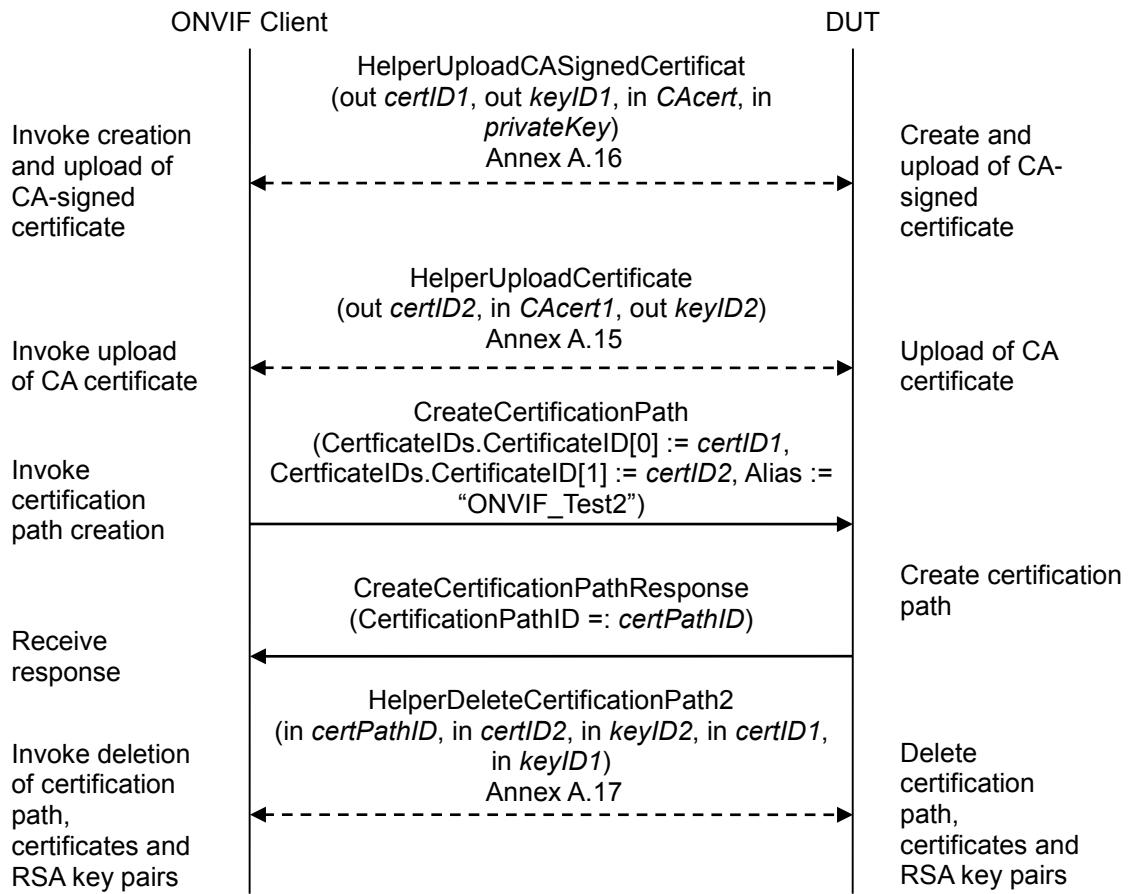
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the creation of a certification path (signed server + CA certificate).

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for two additional RSA key pairs. The DUT shall have enough free storage capacity for two additional certificates. The DUT shall have enough free storage capacity for one additional certification path. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding private key (out *privateKey*) by following the procedure described in Annex A.4.
4. ONVIF Client creates and uploads a CA-signed certificate (out *certID1*) for RSA key pair (out *keyID1*) and associated CA certificate (in *CAcert*) and a corresponding private key (in *privateKey*) by following the procedure described in Annex A.16.
5. ONVIF Client uploads a CA certificate (out *certID2*, in *CAcert*) and new RSA key pair with the public key from the CA certificate (out *keyID2*) by following the procedure described in Annex A.15.
6. ONVIF Client invokes **CreateCertificationPath** with parameters
  - CertificateIDs.CertificateID[0] := *certID1*
  - CertificateIDs.CertificateID[1] := *certID2*



- Alias := “ONVIF\_Test2”
7. The DUT responds with a **CreateCertificationPathResponse** message with parameters
    - CertificationPathID =: *certPathID*
  8. ONVIF Client deletes the certification path (in *certPathID*), related CA certificate (in *certID2*) and the RSA key pair (in *keyID2*) and related the CA-signed certificate (in *certID1*) and the RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.17 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **CreateCertificationPathResponse** message.





**4.2.14 Get Certification Path – self-signed**

**Test Label:** Get certification path - self-signed

**Test Case ID:** ADVANCED\_SECURITY-2-1-14

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** GetCertificationPath

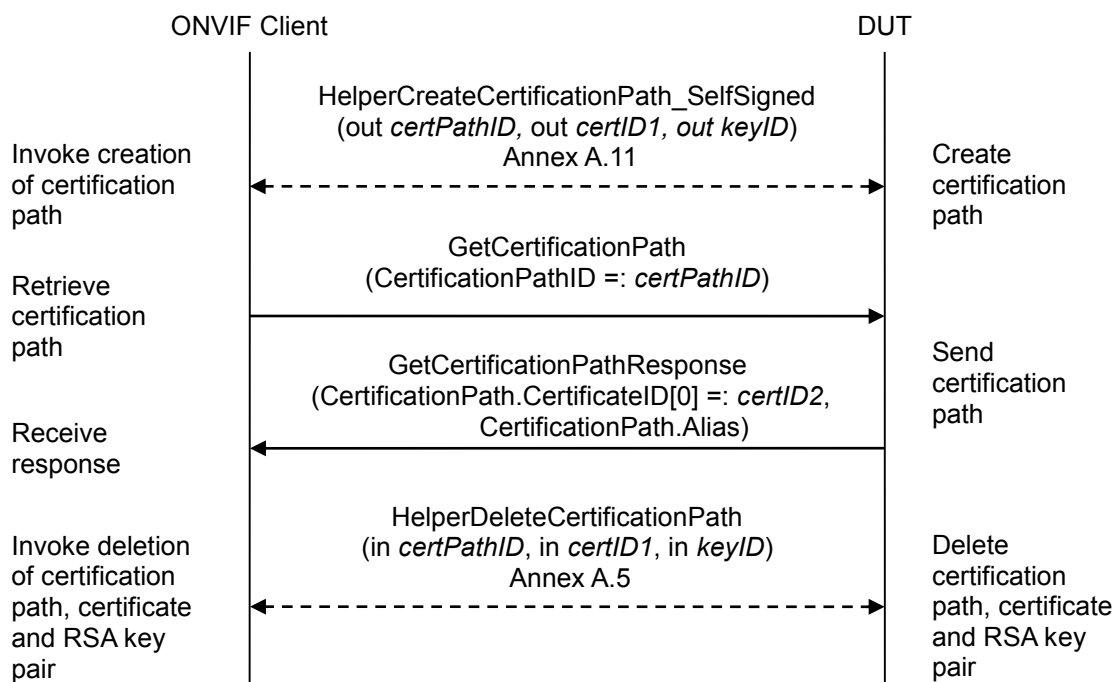
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the retrieval of a certification path containing a self-signed certificate.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certification path.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.



3. ONVIF Client creates a certification path (out *certPathID*) based on self-signed certificate (out *certID1*) and related RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.11.
4. ONVIF Client invokes **GetCertificationPath** message with parameters
  - CertificationPathID =: *certPathID*
5. The DUT responds with a **GetCertificationPathResponse** message with parameters
  - CertificationPath.CertificateID[0] =: *certID2*
  - CertificationPath.Alias
6. If *certID1* is not equal to *certID2*, FAIL the test and go to the step 7.
7. ONVIF Client deletes the certification path (in *certPathID1*) and related the self-signed certificate (in *certID1*) and the RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.5 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passes all assertions.

**FAIL –**

The DUT did not send **GetCertificationPathResponse** message.



**4.2.15 Get Certification Path – CA**

**Test Label:** Get Certification Path – CA

**Test Case ID:** ADVANCED\_SECURITY-2-1-15

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** GetCertificationPath

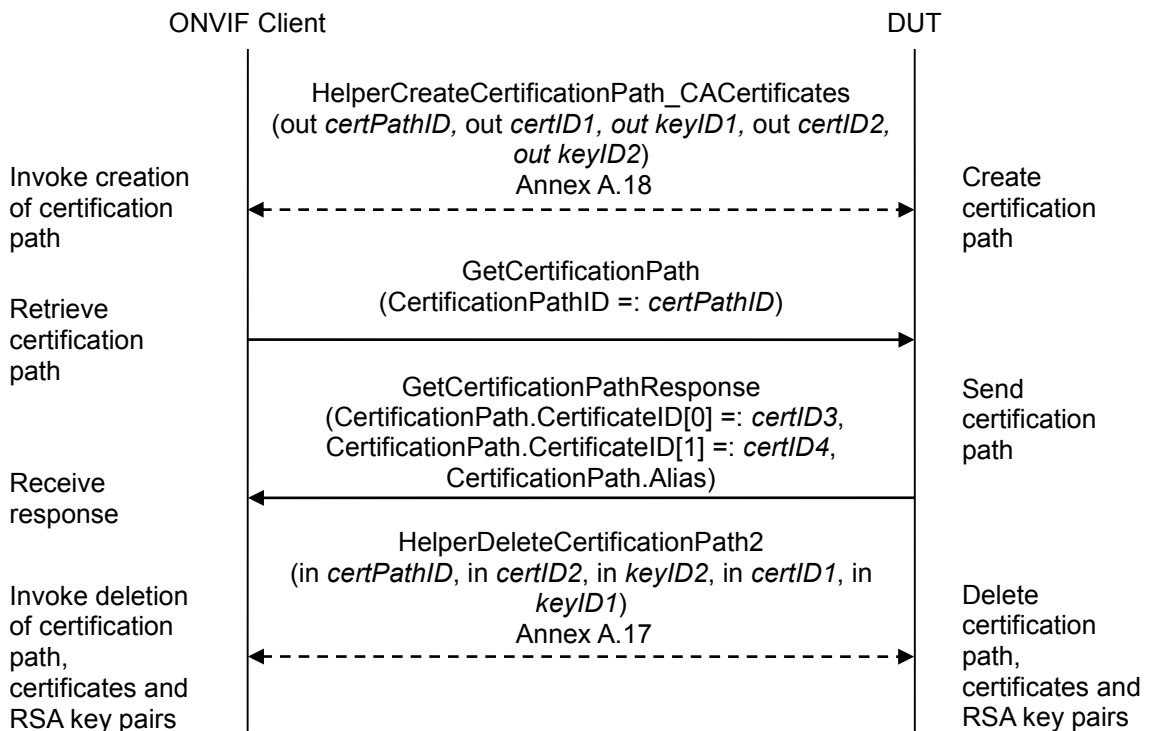
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the retrieval of a certification path containing a signed server and a CA certificate.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for two additional RSA key pairs. The DUT shall have enough free storage capacity for one additional certification path. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**



1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a certification path (out *certPathID*) based on CA-signed certificate (out *certID1*) and related RSA key pair (out *keyID1*) and a corresponding CA certificate (out *certID2*) and related RSA key pair (out *keyID2*) by following the procedure mentioned in Annex A.18.
4. ONVIF Client invokes **GetCertificationPath** message with parameters
  - CertificationPathID =: *certPathID*
5. The DUT responds with a **GetCertificationPathResponse** message with parameters
  - CertificationPath.CertificateID[0] =: *certID3*
  - CertificationPath.CertificateID[1] =: *certID4*
  - CertificationPath.Alias
6. If *certID1* is not equal to *certID3*, FAIL the test and go to the step 8.
7. If *certID2* is not equal to *certID4*, FAIL the test and go to the step 8.
8. ONVIF Client deletes the certification path (in *certPathID*), related the CA certificate (in *certID2*) and the RSA key pair (in *keyID2*) and related CA-signed certificate (in *certID1*) and the RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.17 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **CreateCertificationPathResponse** message.

The DUT did not send **GetCertificationPathResponse** message.



#### **4.2.16 Get All Certification Paths – self-signed**

**Test Label:** Get All Certifications Path - Self-Signed

**Test Case ID:** ADVANCED\_SECURITY-2-1-16

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** GetAllCertificationPaths

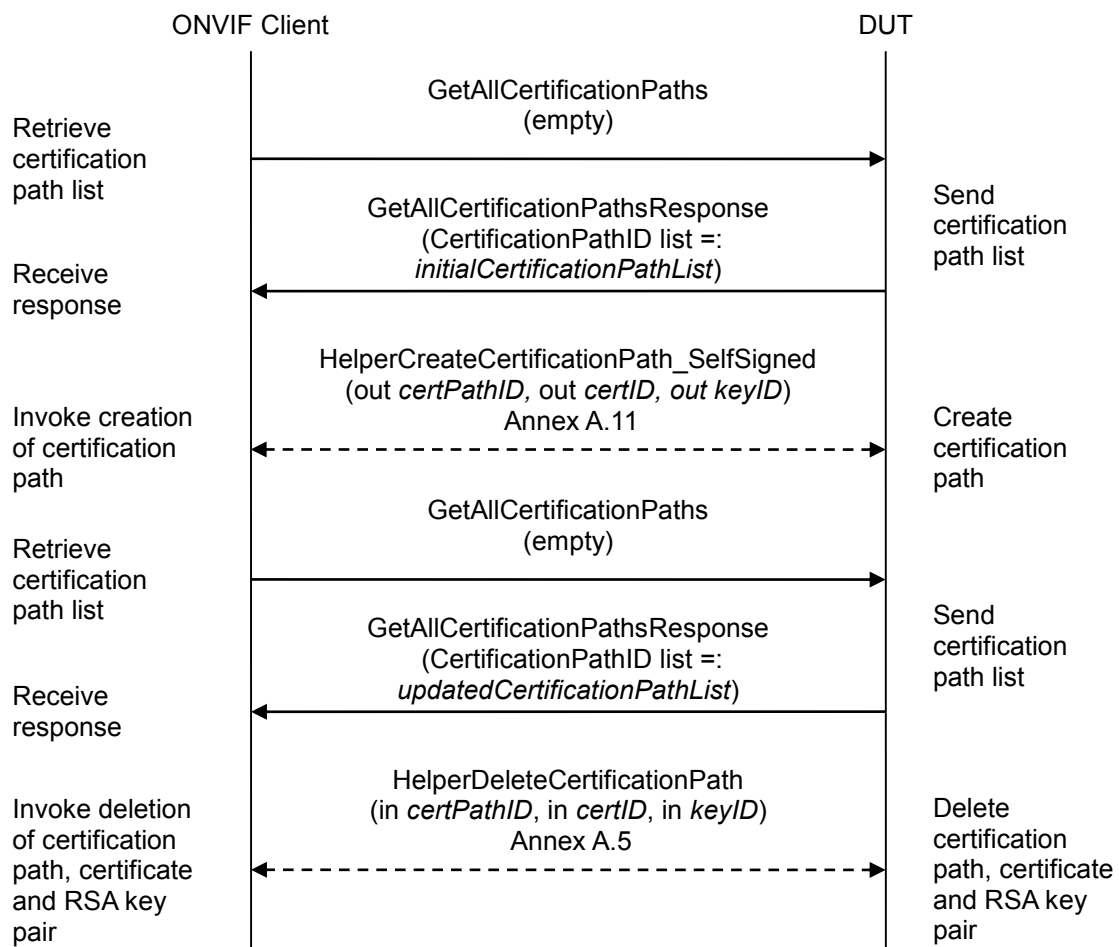
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the retrieval off all certification paths (self-signed certificate).

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. The DUT shall have enough free storage capacity for one additional certification path.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAllCertificationPaths**.
4. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters
  - CertificationPathID list =: *initialCertificationPathList*
5. ONVIF Client creates a certification path (out *certPathID*) based on self-signed certificate (out *certID*) and related RSA key pair (put *keyID*) by following the procedure mentioned in Annex A.11.
6. ONVIF Client invokes **GetAllCertificationPaths**.
7. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters
  - CertificationPathID list =: *updatedCertificationPathList*



8. If *updatedCertificationPathList* does not contain *certPathID* and all certification paths from *initialCertificationPathList*, FAIL the test, and go to the step 10.
9. If *updatedCertificationPathList* contains certification paths other than *certPathID* or certification paths from *initialCertificationPathList*, FAIL the test, and go to the step 10.
10. ONVIF Client deletes the certification path (in *certPathID*) and related self-signed certificate (in *certID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.5 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetAllCertificationPathsResponse** messages(s).

**Note:** The DUT may return an empty list at step 4.



#### **4.2.17 Get All Certification Paths – CA**

**Test Label:** Get All Certification Paths – CA

**Test Case ID:** ADVANCED\_SECURITY-2-1-17

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** GetAllCertificationPaths

**WSDL Reference:** advancedsecurity.wsdl

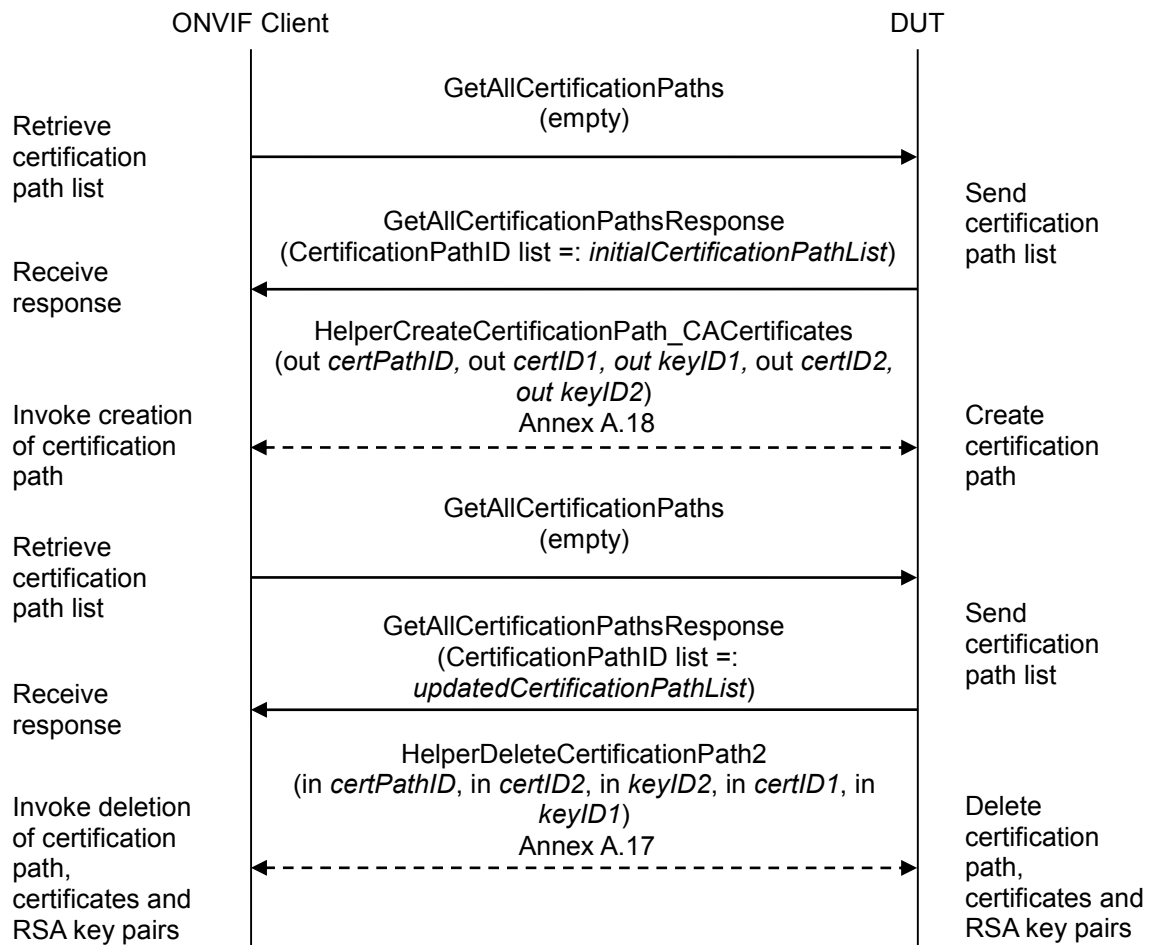
**Test Purpose:** To test the retrieval off all certification paths (CA plus signed certificate).

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for two additional RSA key pairs. The DUT shall have enough free storage capacity for two additional certificates. The DUT shall have enough free storage capacity for one additional certification path. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAllCertificationPaths**.
4. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters
  - CertificationPathID list =: *initialCertificationPathList*
5. ONVIF Client creates a certification path (out *certPathID*) based on CA-signed certificate (out *certID1*) and related RSA key pair (out *keyID1*) and a corresponding CA certificate (out *certID2*) and related RSA key pair (out *keyID2*) by following the procedure mentioned in Annex A.18.
6. ONVIF Client invokes GetAllCertificationPaths.
7. The DUT responds with a GetAllCertificationPathsResponse message with parameters
  - CertificationPathID list =: *updatedCertificationPathList*



8. If *updatedCertificationPathList* does not contain *certPathID* and all certification paths from *initialCertificationPathList*, FAIL the test, and go to the step 9.
9. If *updatedCertificationPathList* contains certification paths other than *certPathID* or certification paths from *initialCertificationPathList*, FAIL the test, and go to the step 9.
10. ONVIF Client deletes the certification path (in *certPathID*), related CA certificate (in *certID2*) and RSA key pair (in *keyID2*) and related the CA-signed certificate (in *certID1*) and the RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.17 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetAllCertificationPathsResponse** messages(s).

**Note:** The DUT may return an empty list at step 4.



**4.2.18 Delete Certification Path – self-signed**

**Test Label:** Delete Certifications Path - Self-Signed

**Test Case ID:** ADVANCED\_SECURITY-2-1-18

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** DeleteCertificationPath

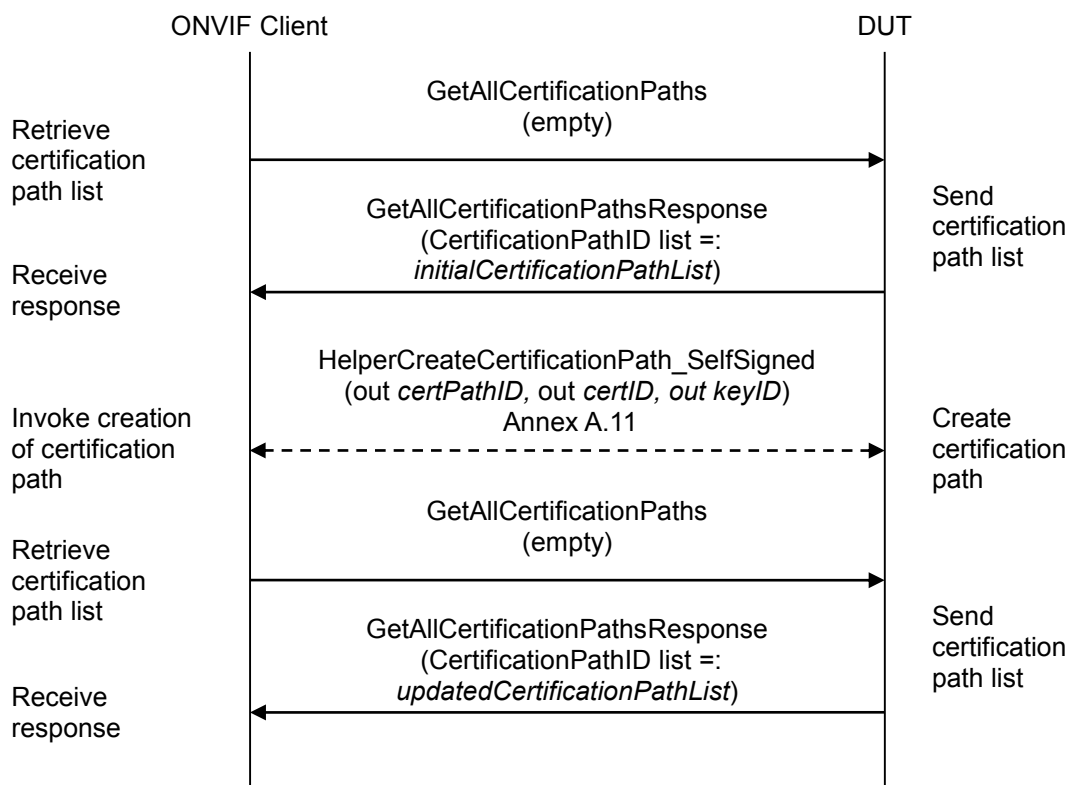
**WSDL Reference:** advancedsecurity.wsdl

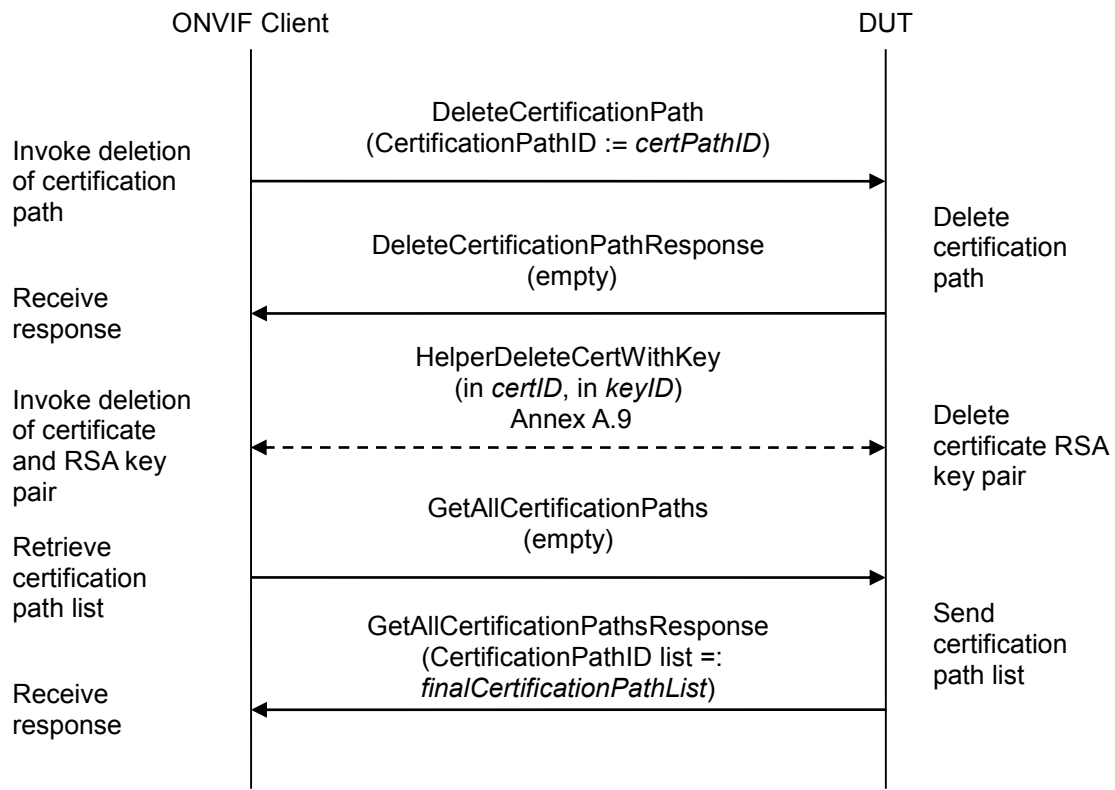
**Test Purpose:** To test the deletion of a certification path (self-signed certificate).

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. The DUT shall have enough free storage capacity for one additional certification path.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAllCertificationPaths**.
4. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters
  - CertificationPathID list =: *initialCertificationPathList*
5. ONVIF Client creates a certification path (out *certPathID*) based on self-signed certificate (out *certID*) and related RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.11.
6. ONVIF Client invokes **GetAllCertificationPaths**.
7. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters
  - CertificationPathID list =: *updatedCertificationPathList*
8. If *updatedCertificationPathList* does not contain *certPathID* and all certification paths from *initialCertificationPathList*, FAIL the test, delete the certification path (in *certPathID*) and related self-signed certificate (in *certID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.5 to restore DUT configuration, and skip other steps.



9. If *updatedCertificationPathList* contains certification paths other than *certPathID* or certification paths from *initialCertificationPathList*, FAIL the test, delete the certification path (in *certPathID*) and related self-signed certificate (in *certID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.5 to restore DUT configuration, and skip other steps.
10. ONVIF Client invokes **DeleteCertificationPath** with parameters
  - CertificationPathID := *certPathID*
11. The DUT responds with a **DeleteCertificationPathResponse** message.
12. ONVIF Client deletes the self-signed certificate (in *certID*) and related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.9 to restore DUT configuration.
13. ONVIF Client invokes **GetAllCertificationPaths**.
14. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters
  - CertificationPathID list =: *finalCertificationPathList*
15. If *finalCertificationPathList* is not equal *initialCertificationPathList*, FAIL the test.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **DeleteCertificationResponse** message.

The DUT did not send **GetAllCertificationPathsResponse** messages(s).

**Note:** The DUT may return an empty list at step 4.



**4.2.19 Delete Certification Path - CA**

**Test Label:** Delete Certifications Path – CA

**Test Case ID:** ADVANCED\_SECURITY-2-1-19

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** DeleteCertification

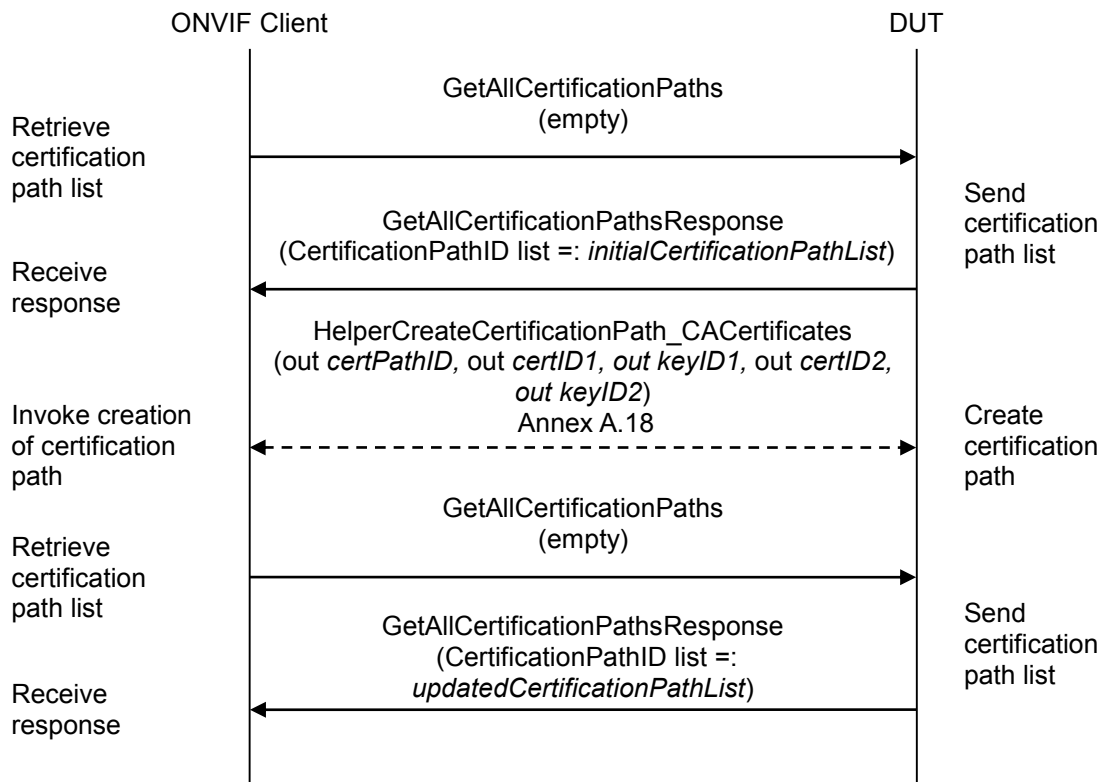
**WSDL Reference:** advancedsecurity.wsdl

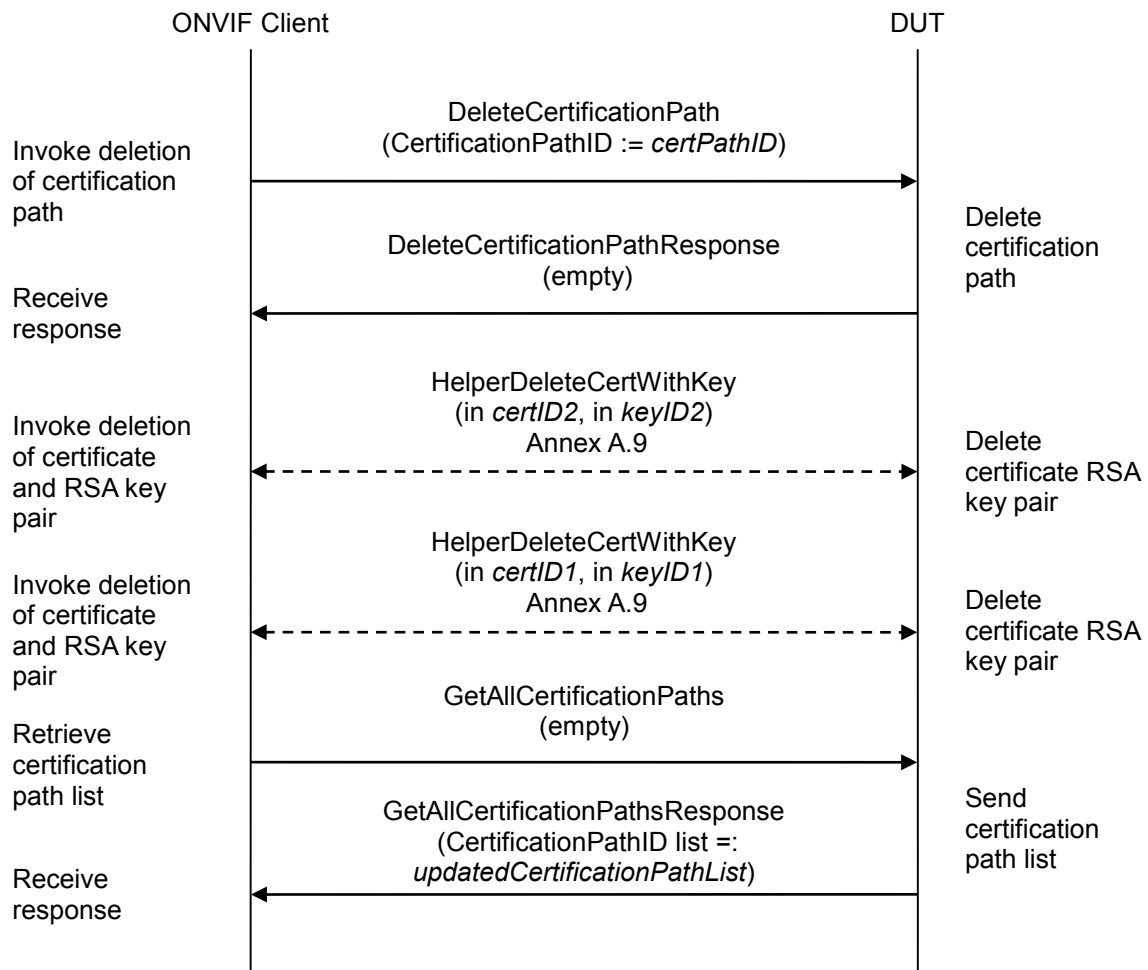
**Test Purpose:** To test the deletion of a certification path (CA plus signed certificate).

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for two additional RSA key pairs. The DUT shall have enough free storage capacity for two additional certificates. The DUT shall have enough free storage capacity for one additional certification path. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAllCertificationPaths**.
4. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters
  - CertificationPathID list =: *initialCertificationPathList*
5. ONVIF Client creates a certification path (out *certPathID*) based on CA-signed certificate (out *certID1*) and related RSA key pair (out *keyID1*) and a corresponding CA certificate (out *certID2*) and related RSA key pair (out *keyID2*) by following the procedure mentioned in Annex A.18.
6. ONVIF Client invokes **GetAllCertificationPaths**.
7. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters
  - CertificationPathID list =: *updatedCertificationPathList*



8. If *updatedCertificationPathList* does not contain *certPathID* and all certification paths from *initialCertificationPathList*, FAIL the test, perform steps 9-12 to restore DUT settings, and skip other steps.
9. If *updatedCertificationPathList* contains certification paths other than *certPathID* or certification paths from *initialCertificationPathList*, FAIL the test, perform steps 9-12 to restore DUT settings, and skip other steps.
10. ONVIF Client invokes **DeleteCertificationPath** with parameters
  - CertificationPathID := *certPathID*
11. The DUT responds with a **DeleteCertificationPathResponse** message.
12. ONVIF Client deletes the CA certificate (*certID2*) and related RSA key pair (*keyID2*) by following the procedure mentioned in Annex A.9 to restore DUT configuration.
13. ONVIF Client deletes the CA certificate (*certID1*) and related RSA key pair (*keyID1*) by following the procedure mentioned in Annex A.9 to restore DUT configuration.
14. ONVIF Client invokes **GetAllCertificationPaths**.
15. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters
  - CertificationPathID list =: *finalCertificationPathList*
16. If *finalCertificationPathList* is not equal *initialCertificationPathList*, FAIL the test.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetAllCertificationPathsResponse** messages(s).

The DUT did not send **DeleteCertificationPathResponse** messages.

**Note:** The DUT may return an empty list at step 4.





**4.2.20 CreatePKCS10CSR – negative test**

**Test Label:** Create PKCS#10 – negative test

**Test Case ID:** ADVANCED\_SECURITY-2-1-20

**ONVIF Core Specification Coverage:** Advanced Security, Keystore – Certificate Management

**Command Under Test:** CreatePKCS10CSR

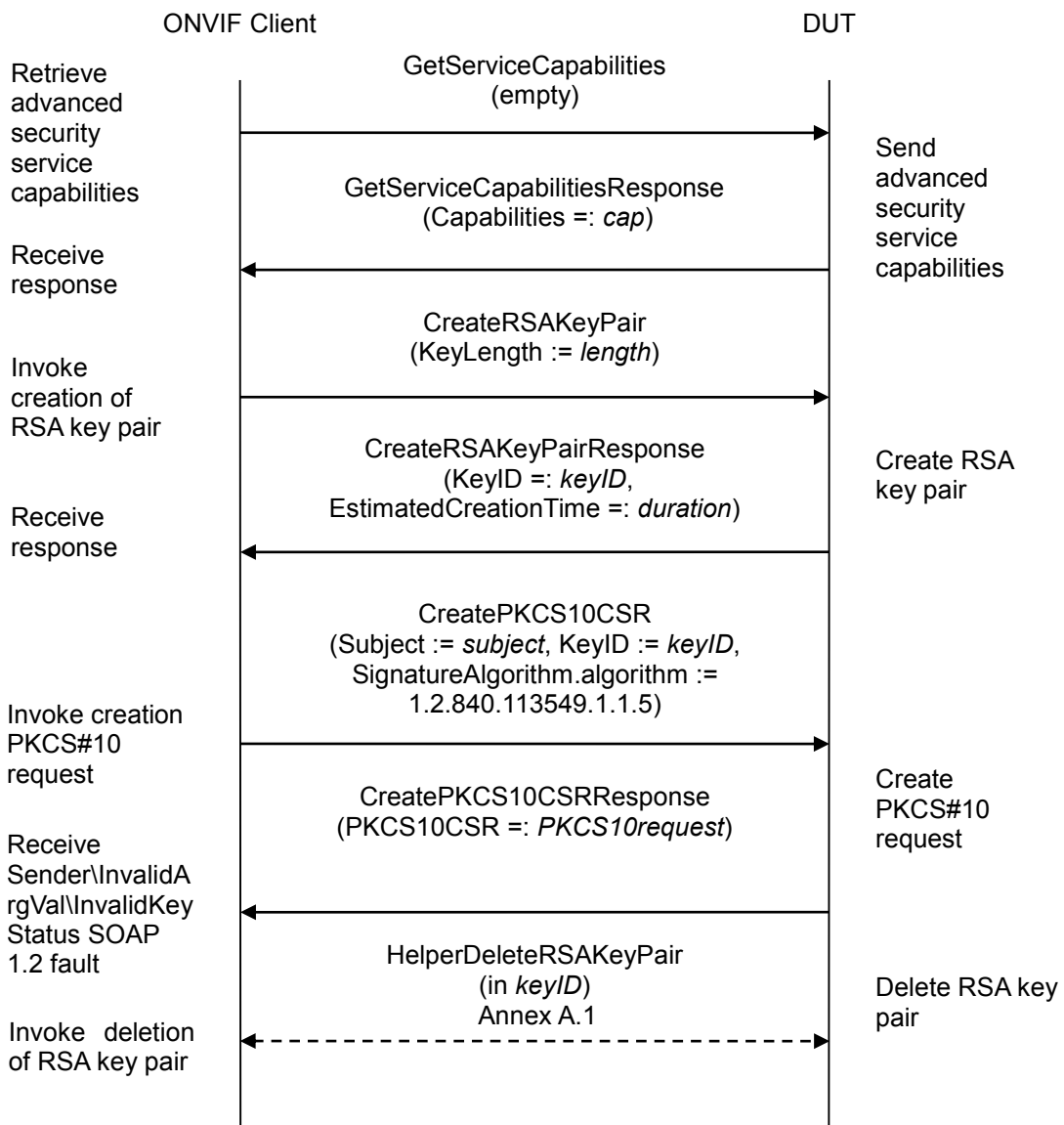
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To verify env:Sender\ter:InvalidArgVal\ter:InvalidKeyStatus is returned when key pair has status Generating.

**Pre-Requisite:** Advanced Security Service is received from the DUT. Create PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional RSA key pair.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServiceCapabilities**.
4. The DUT responds with a **GetServiceCapabilitiesResponse** message with parameters
  - Capabilities =: *cap*
5. Set *keyLength* := (the smallest key length in the list of supported RSA key lengths (*cap.RSAKeyLengths*) that is greater than or equal to 1024).
6. If there is no such key length, set *keyLength* := (the largest supported RSA key length).



7. ONVIF Client invokes **CreateRSAKeyPair** with parameter
  - $KeyLength := keyLength$
8. The DUT responds with **CreateRSAKeyPairResponse** message with parameters
  - $KeyID =: keyID$
  - $EstimatedCreationTime =: duration$
9. If  $duration$  is less than 2 sec:
  - 9.1. ONVIF Client deletes RSA key pair (in  $keyID$ ) by following the procedure mentioned in Annex A.1.
  - 9.2. Set  $keyLength :=$  (the smallest supported RSA key length that is larger than the current  $keyLength$ )
  - 9.3. If no such key length exists, log WARNING message, and PASS the test.
  - 9.4. Go to step 7.
10. ONVIF Client invokes **CreatePKCS10CSR** with parameter
  - $Subject := subject$  (see Annex A.2)
  - $KeyID := keyID$
  - CSRAtribute skipped
  - $SignatureAlgorithm.algorithm := 1.2.840.113549.1.1.5$  (OID of SHA-1 with RSA Encryption algorithm)
11. The DUT returns env:Sender\ter:InvalidArgVal\ter:InvalidKeyStatus SOAP 1.2 fault.
12. If the DUT returns normal **CreatePKCS10CSRResponse** message:
  - 12.1. ONVIF Client deletes the RSA key pair (in  $keyID$ ) by following the procedure mentioned in Annex A.1,
  - 12.2. Set  $keyLength :=$  (the smallest supported RSA key length that is larger than the current  $keyLength$ )
  - 12.3. If no such key length exists, log WARNING message, and PASS the test.
  - 12.4. Go to step 7.
13. ONVIF Client deletes the RSA key pair (in  $keyID$ ) by following the procedure mentioned in Annex A.1 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**



The DUT did not send env:Sender\ter:InvalidArgVal\ter:InvalidKeyStatus SOAP 1.2 fault.

The DUT did not send **CreateRSAKeyPair** messages.



#### **4.2.21 DeleteCertificate – CA – Preserve Public Key**

**Test Label:** Delete Certificate – CA – Preserve Public Key

**Test Case ID:** ADVANCED\_SECURITY-2-1-21

**ONVIF Core Specification Coverage:** Advanced Security, Keystore – Certificate Management

**Command Under Test:** DeleteCertificate

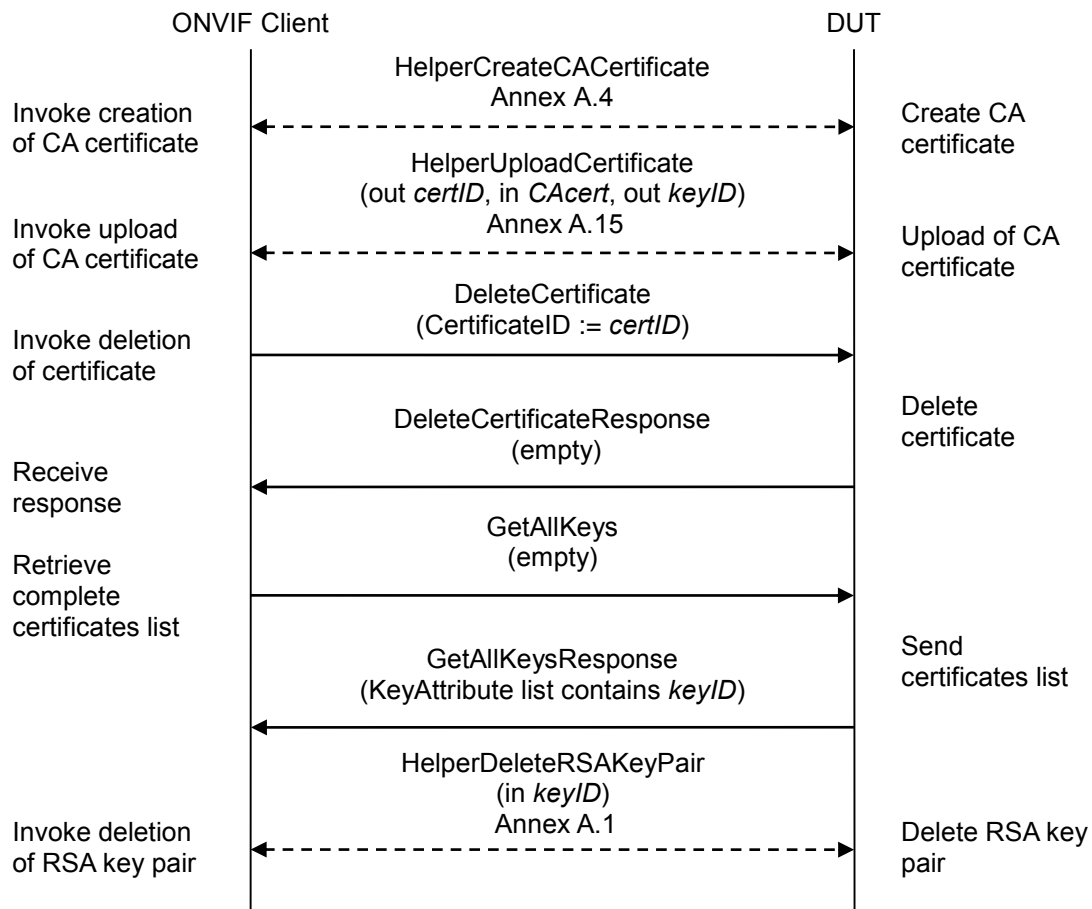
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test that the DUT does not delete the public key that is contained in the certificate from the keystore.

**Pre-Requisite:** Advanced Security Service is received from the DUT. Create PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. The DUT shall have enough free storage capacity for one additional key pair. The DUT shall have enough free storage capacity for one additional certificate. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



### Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a CA certificate (out *CACert*) and a corresponding private key (out *privateKey*) by following the procedure described in Annex A.4.
4. ONVIF Client uploads the CA certificate (out *certID*, in *CACert*) and new RSA key pair with the public key from the CA certificate (out *keyID*) by following the procedure described in Annex A.15.
5. ONVIF Client invokes **DeleteCertificate** with parameters
  - *CertificateID := certID*
6. The DUT responds with a **DeleteCertificateResponse** message.
7. ONVIF Client invokes **GetAllKeys**.
8. The DUT responds with a **GetAllKeysResponse** message where KeyAttribute list contains *keyID*.
9. ONVIF Client deletes the RSA key pair (in *keyID*) by following the procedure mentioned in



Annex A.1 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The **GetAllKeysResponse** message at Step 8 does not contain *keyID* in KeyAttribute list.

The DUT did not send **DeleteCertificateResponse** message.

The DUT did not send **GetAllKeysResponse** message.



**4.2.22 Upload certificate – delete linked key (negative test)**

**Test Label:** Upload certificate – delete linked key (negative test)

**Test Case ID:** ADVANCED\_SECURITY-2-1-22

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** UploadCertificate

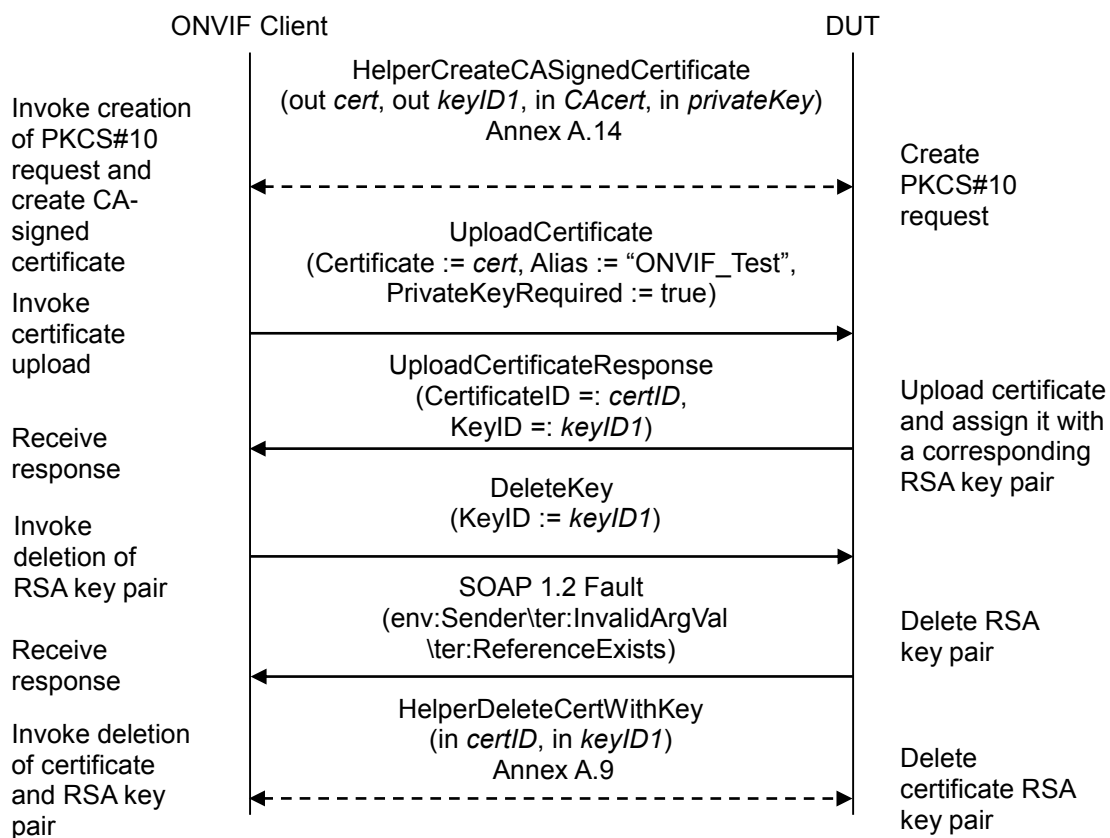
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the link of a certificate to RSA Key Pair by attempting to delete key of an uploaded certificate.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**





1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding private key (out *privateKey*) by following the procedure described in Annex A.4.
4. ONVIF Client creates a certificate (out *cert*) from the PKCS#10 request with RSA key pair (out *keyID1*) and associated CA certificate (in *CAcert*) and a corresponding private key (in *privateKey*) by following the procedure described in Annex A.14.
5. ONVIF Client invokes **UploadCertificate** with parameters
  - Certificate := *cert*
  - Alias := "ONVIF\_Test"
  - PrivateKeyRequired : = true
6. The DUT responds with a **UploadCertificateResponse** message with parameters
  - CertificateID =: *certID*
  - KeyID =: *keyID1*
7. ONVIF Client invokes **DeleteKey** with parameters
  - KeyID := *keyID1*
8. The DUT returns env:Sender\ter:InvalidArgVal\ter:ReferenceExists SOAP 1.2 fault.
9. ONVIF Client deletes the CA certificate (in *certID*) and related RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.9 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send env:Sender\ter:InvalidArgVal\ter:ReferenceExists SOAP 1.2 fault.

The DUT did not send **UploadCertificateResponse** message.

The DUT did not send **DeleteKeyResponse** message.



#### 4.2.23 Upload certificate – Upload malformed certificate (negative test)

**Test Label:** Upload certificate – Upload malformed certificate (negative test)

**Test Case ID:** ADVANCED\_SECURITY-2-1-23

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** UploadCertificate

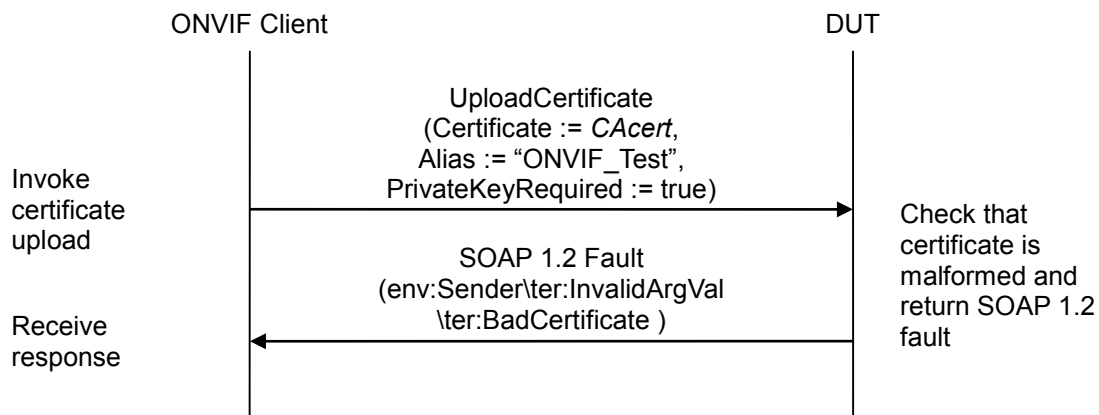
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To verify that DUT produces InvalidArgVal\BadCertificate fault if UploadCertificate is invoked for a malformed X.509 certificate.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



#### Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding private key (out *privateKey*) by following the procedure described in Annex A.4.
4. ONVIF Client corrupts *CAcert*.
5. ONVIF Client invokes **UploadCertificate** with parameters
  - Certificate := *CAcert* (malformed)



- Alias := "ONVIF\_Test"
- PrivateKeyRequired : = false

6. The DUT returns env:Sender\ter:InvalidArgVal\ter:BadCertificate SOAP 1.2 fault.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send env:Sender\ter:InvalidArgVal\ter:BadCertificate SOAP 1.2 fault.

The DUT did not send **UploadCertificateResponse** message.



#### 4.2.24 Upload certificate – Upload expired certificate

**Test Label:** Upload certificate – Upload expired certificate

**Test Case ID:** ADVANCED\_SECURITY-2-1-24

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** UploadCertificate

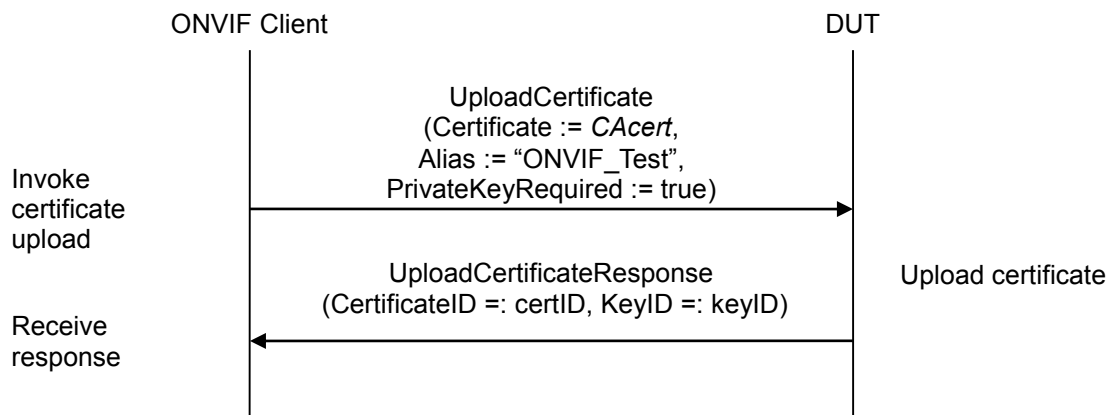
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To verify that DUT does not produce an InvalidArgVal\BadCertificate fault if UploadCertificate is invoked for an expired certificate.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



#### Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding private key (out *privateKey*) by following the procedure described in Annex A.22.
4. ONVIF Client invokes **UploadCertificate** with parameters
  - Certificate := *CAcert*
  - Alias := "ONVIF\_Test"
  - PrivateKeyRequired := false



5. The DUT responds with an **UploadCertificateResponse** message with parameters
  - CertificateID =: *certID*
  - KeyID =: *keyID*
6. Check that the DUT does not return env:Sender\ter:InvalidArgVal\ter:BadCertificate SOAP 1.2 fault in **UploadCertificateResponse**.
7. ONVIF Client deletes the CA certificate (in *certID*) and related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.9 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT send env:Sender\ter:InvalidArgVal\ter:BadCertificate SOAP 1.2 fault.

The DUT did not send **UploadCertificateResponse** message.

**Note:** If the DUT sends another SOAP 1.2 fault message, log WARNING message, and PASS the test.





**4.2.25 CreateSelfSignedCertificate with PKCS#12**

**Test Label:** CreateSelfSignedCertificate with PKCS#12

**Test Case ID:** ADVANCED\_SECURITY-2-1-27

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** CreateSelfSignedCertificate

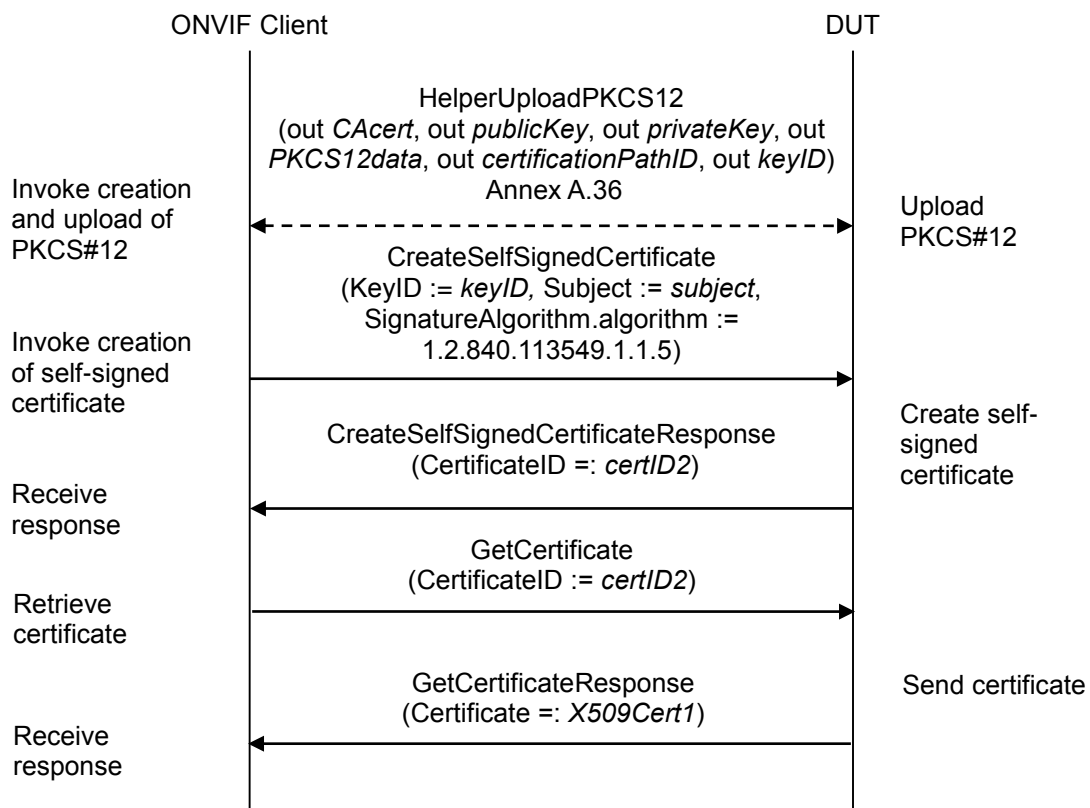
**WSDL Reference:** advancedsecurity.wsdl

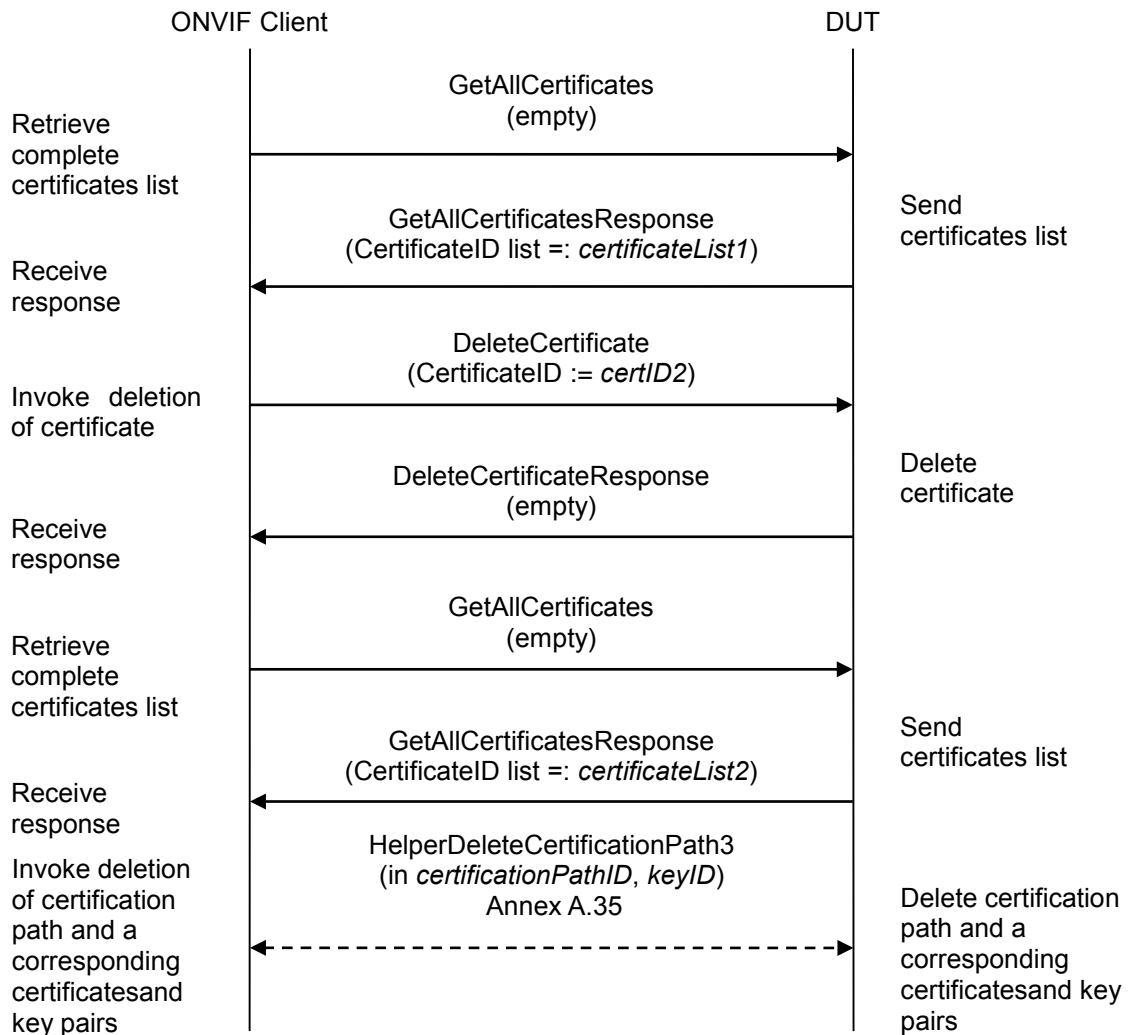
**Test Purpose:** Verify that DUT correctly handles self-signed certificates that are based on keys uploaded to the DUT with PKCS#12.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. Certificate along with an RSA private key in a PKCS#12 data structure upload is supported by the DUT as indicated by the PKCS12CertificateWithRSAPrivateKeyUpload capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for two additional certificates. The DUT shall have enough free storage capacity for one additional certification path.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding public key (out *publicKey*) in the certificate along with the corresponding private key (out *privateKey*) in the form of a PKCS#12 file (out *PKCS12data*) and uploads it with certification path ID (out *certificationPathID*) and key pair ID (out *keyID*) by following the procedure described in Annex A.36.
4. ONVIF Client invokes **CreateSelfSignedCertificate** with parameters
  - X509Version skipped
  - KeyID := *keyID*
  - Subject := *subject* (see Annex A.34)





- Alias skipped
  - notValidBefore skipped
  - notValidAfter skipped
  - SignatureAlgorithm.algorithm := 1.2.840.113549.1.1.5 (OID of SHA-1 with RSA Encryption algorithm)
  - SignatureAlgorithm.parameters skipped
  - SignatureAlgorithm.anyParameters skipped
  - Extension skipped
5. The DUT responds with a **CreateSelfSignedCertificateResponse** message with parameters
    - CertificateID =: *certID2*
  6. ONVIF Client invokes **GetCertificate** message with parameters
    - CertificateID := *certID2*
  7. The DUT responds with a **GetCertificateResponse** message with parameters
    - Certificate =: *X509Cert1*
  8. If *X509Cert1.CertificateID* is not equal to *certID2*, FAIL the test and go to the step 19.
  9. If *X509Cert1.KeyID* is not equal to *keyID*, FAIL the test and go to the step 19.
  10. ONVIF Client validates that *X509Cert1.CertificateContent* is correctly DER encoded (see Annex A.19).
  11. If *X509Cert1.CertificateContent* is incorrectly DER encoded, FAIL the test and go to the step 19.
  12. ONVIF Client verifies that the subject in *X509Cert1.CertificateContent* is equal to *subject*.
  13. If *X509Cert1.CertificateContent* contains a wrong subject, FAIL the test and go to the step 19.
  14. ONVIF Client invokes **GetAllCertificates**.
  15. The DUT responds with a **GetAllCertificatesResponse** message with parameters
    - CertificateID list =: *certificateList1*
  16. If *certificateList1* does not contain certificate with *Certificate.CertificateID* equal to *certID2*, FAIL the test and go to the step 19.
  17. Set:
    - *certificateList1.Certificate[CertificateID = certID2]* =: *X509Cert2*
  18. If *X509Cert1* is not equal to *X509Cert2*, FAIL the test and go to the step 19.
  19. ONVIF Client invokes **DeleteCertificate** with parameters
    - CertificateID := *certID2*



20. The DUT responds with a **DeleteCertificateResponse** message.

21. If test did not fail:

21.1. ONVIF Client invokes **GetAllCertificates**.

21.2. The DUT responds with a **GetAllCertificatesResponse** message with parameters

- CertificateID list =: *certificateList2*

21.3. If *certificateList2* contains certificate with Certificate.CertificateID equal to *certID2*, FAIL the test and go to the step 22.

22. ONVIF Client deletes the certification path (in *certificationPathID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.35 to restore DUT configuration and skip other steps.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **CreateSelfSignedCertificateResponse** message.

The DUT did not send **GetCertificateResponse** message.

The DUT did not send **GetAllCertificatesResponse** message.

The DUT did not send **DeleteCertificateResponse** message.

**Note:** The following fields are compared at step 18:

- CertificateID
- KeyID
- Alias
- CertificateContent



**4.2.26 Create PKCS#10 request with PKCS#12**

**Test Label:** Create PKCS#10 Request with PKCS#12

**Test Case ID:** ADVANCED\_SECURITY-2-1-28

**ONVIF Core Specification Coverage:** Advanced Security, Keystore - Certificate Management

**Command Under Test:** CreatePKCS10CSR

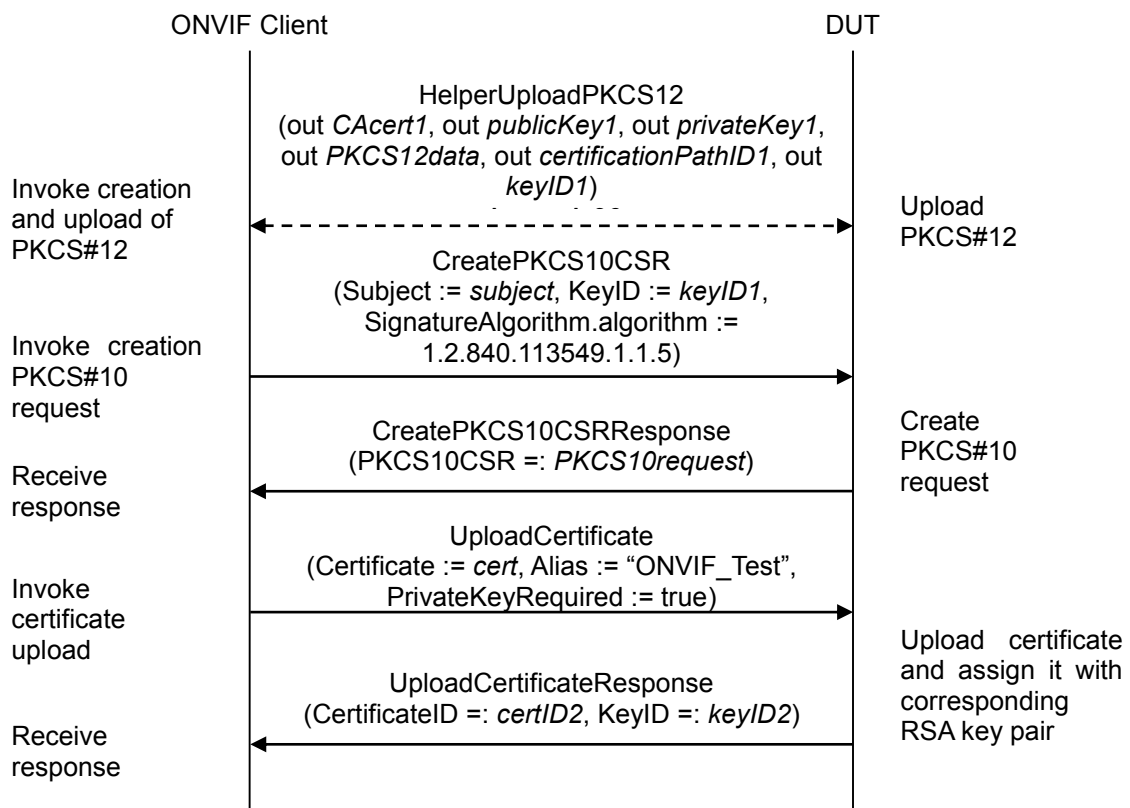
**WSDL Reference:** advancedsecurity.wsdl

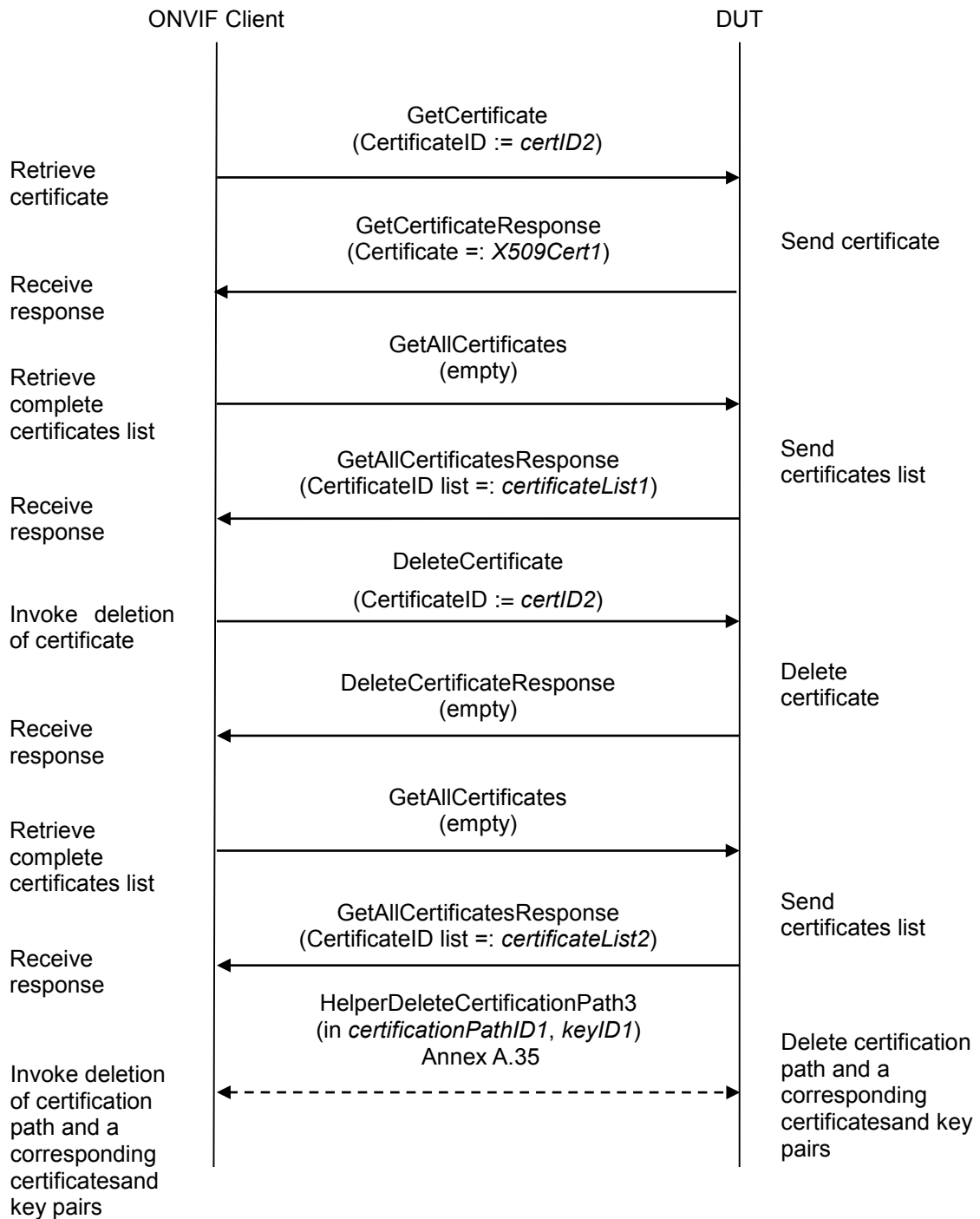
**Test Purpose:** Verify that DUT correctly generates a PKCS#10 request for a key pair that is uploaded with PKCS#12.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. Certificate along with an RSA private key in a PKCS#12 data structure upload is supported by the DUT as indicated by the PKCS12CertificateWithRSAPrivateKeyUpload capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certification path.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a CA certificate (out *CAcert1*) and a corresponding public key (out



*publicKey1*) in the certificate along with the corresponding private key (out *privateKey1*) in the form of a PKCS#12 file (out *PKCS12data*) and uploads it with certification path ID (out *certificationPathID1*) and key pair ID (out *keyID1*) by following the procedure described in Annex A.36.

4. ONVIF Client invokes **CreatePKCS10CSR** with parameter
  - Subject := *subject* (see Annex A.2)
  - KeyID := *keyID1*
  - CSRAttribute skipped
  - SignatureAlgorithm.algorithm := 1.2.840.113549.1.1.5 (OID of SHA-1 with RSA Encryption algorithm)
5. The DUT responds with **CreatePKCS10CSRResponse** message with parameters
  - PKCS10CSR =: *PKCS10request*
6. ONVIF Client creates a CA certificate (out *CAcert2*) with subject (in “CN=ONVIF TT2,C=US”) and a corresponding public key (out *publicKey2*) in the certificate along with the corresponding private key (out *privateKey2*) by following the procedure described in Annex A.4.
7. ONVIF Client creates a certificate (out *cert*) from the PKCS#10 request (in *PKCS10request*) and an associated CA certificate (in *CAcert2*) with related private key (in *privateKey2*) by following the procedure described in Annex A.3.
8. ONVIF Client invokes **UploadCertificate** with parameters
  - Certificate := *cert*
  - Alias := “ONVIF\_Test”
  - PrivateKeyRequired := false
9. The DUT responds with a **UploadCertificateResponse** message with parameters
  - CertificateID =: *certID2*
  - KeyID =: *keyID2*
10. If *keyID2* is not equal to *keyID1*, FAIL the test and go to step 24.
11. ONVIF Client invokes **GetCertificate** message with parameters
  - CertificateID := *certID2*
12. The DUT responds with a **GetCertificateResponse** message with parameters
  - Certificate =: *X509Cert1*
13. If *X509Cert1.CertificateID* is not equal to *certID2*, FAIL the test and go to the step 24.
14. If *X509Cert1.KeyID* is not equal to *keyID1*, FAIL the test and go to the step 24.
15. ONVIF Client validates that *X509Cert1.CertificateContent* is correctly DER encoded (see Annex A.19).



16. If *X509Cert1.CertificateContent* is incorrectly DER encoded, FAIL the test and go to the step 24.
17. ONVIF Client verifies that the subject in *X509Cert1.CertificateContent* is equal to "CN=ONVIF TT2,C=US".
18. If *X509Cert1.CertificateContent* contains a wrong subject, FAIL the test and go to the step 24.
19. ONVIF Client invokes **GetAllCertificates**.
20. The DUT responds with a **GetAllCertificatesResponse** message with parameters
  - CertificateID list =: *certificateList1*
21. If *certificateList1* does not contain certificate with Certificate.CertificateID equal to *certID2*, FAIL the test and go to the step 24.
22. Set:
  - *certificateList1.Certificate*[CertificateID = *certID2*] =: *X509Cert2*
23. If *X509Cert1* is not equal to *X509Cert2*, FAIL the test and go to the step 24.
24. ONVIF Client invokes **DeleteCertificate** with parameters
  - CertificateID := *certID2*
25. The DUT responds with a **DeleteCertificateResponse** message.
26. If test did not fail:
  - 26.1. ONVIF Client invokes **GetAllCertificates**.
  - 26.2. The DUT responds with a **GetAllCertificatesResponse** message with parameters
    - CertificateID list =: *certificateList2*
  - 26.3. If *certificateList2* contains certificate with Certificate.CertificateID equal to *certID2*, FAIL the test and go to the step 28.
27. If step 10 failed:
  - 27.1. ONVIF Client invokes **DeleteKey** with parameters
    - KeyID =: *keyID2*
  - 27.2. The DUT responds with a **DeleteKeyResponse** message.
28. ONVIF Client deletes the certification path (in *certificationPathID1*) and RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.35 to restore DUT configuration and skip other steps.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**



The DUT did not send **CreatePKCS10CSRResponse** message.

The DUT did not send **UploadCertificateResponse** message.

The DUT did not send **GetCertificateResponse** message.

The DUT did not send **GetAllCertificatesResponse** message.

The DUT did not send **DeleteKeyResponse** message.

The DUT did not send **DeleteCertificateResponse** message.

**Note:** The following fields are compared at step 23:

- CertificateID
- KeyID
- Alias
- CertificateContent



### 4.3 TLS Server

#### 4.3.1 Certificate Management

##### 4.3.1.1 Add Server Certificate Assignment – self-signed

**Test Label:** Add Server Certificate Assignment - Self-Signed

**Test Case ID:** ADVANCED\_SECURITY-3-1-1

**ONVIF Core Specification Coverage:** Advanced Security, TLS Server

**Command Under Test:** AddServerCertificateAssignment

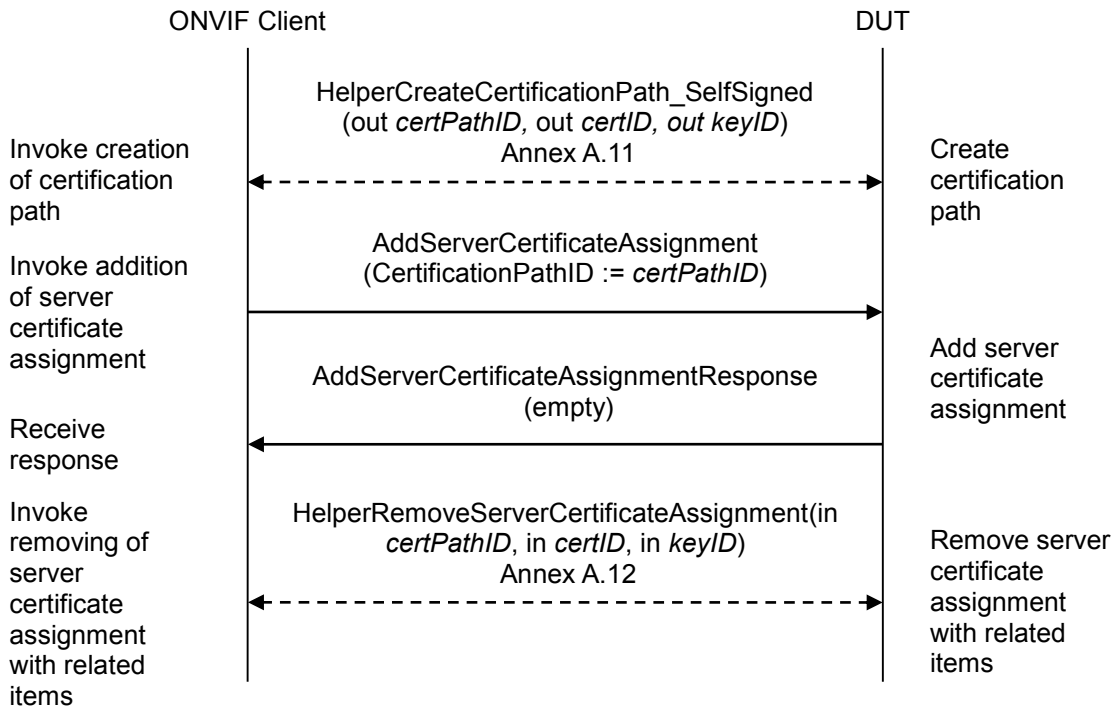
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the assignment of a self-signed certificate to a TLS server.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional server certificate assignment.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**







**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a certification path (out *certPathID*) based on self-signed certificate (out *certID1*) and related RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.11.
4. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
  - CertificationPathID := *certPathID*
5. The DUT responds with an **AddServerCertificateAssignmentResponse** message.
6. ONVIF Client removes server certification assignment and deletes related certification path (in *certPathID*), self-signed certificate (in *certID*) and the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.12 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **AddServerCertificateAssignmentResponse** message.



**4.3.1.2 Add Server Certificate Assignment – CA**

**Test Label:** Add Server Certificate Assignment – CA

**Test Case ID:** ADVANCED\_SECURITY-3-1-2

**ONVIF Core Specification Coverage:** Advanced Security, TLS Server

**Command Under Test:** AddServerCertificateAssignment

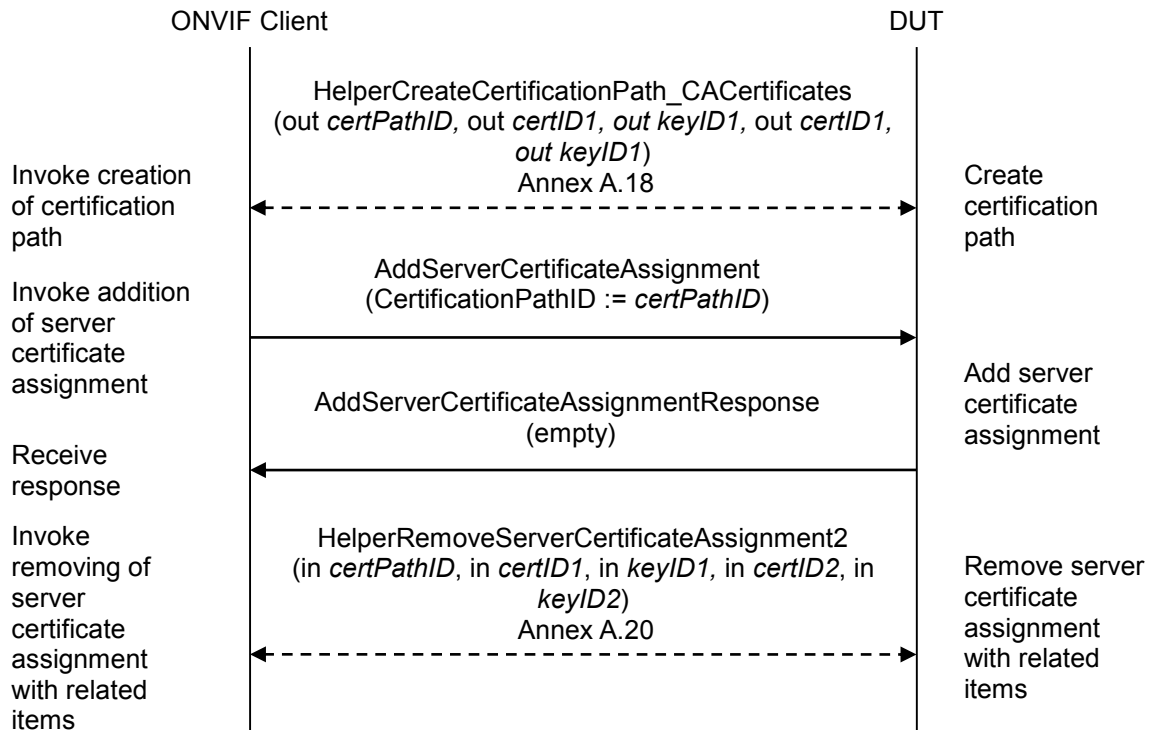
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the assignment of a certificate (signed + CA) to a TLS server.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for two additional RSA key pairs. The DUT shall have enough free storage capacity for two additional certificates. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional server certificate assignment. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**



1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a certification path (out *certPathID*) based on CA-signed certificate (out *certID1*) and related RSA key pair (out *keyID1*) and a corresponding CA certificate (out *certID2*) and related RSA key pair (out *keyID2*) by following the procedure mentioned in Annex A.18.
4. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
  - CertificationPathID := *certPathID*
5. The DUT responds with an **AddServerCertificateAssignmentResponse** message.
6. ONVIF Client removes server certification assignment and deletes related certification path (in *certPathID*), related CA certificate (in *certID2*) and RSA key pair (in *keyID2*) and related CA-signed certificate (in *certID1*) and RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.20 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **AddServerCertificateAssignmentResponse** message.



#### **4.3.1.3 Replace Server Certificate Assignment – self-signed**

**Test Label:** Replace Server Certificate Assignment - Self-Signed

**Test Case ID:** ADVANCED\_SECURITY-3-1-3

**ONVIF Core Specification Coverage:** Advanced Security, TLS Server

**Command Under Test:** ReplaceServerCertificateAssignment

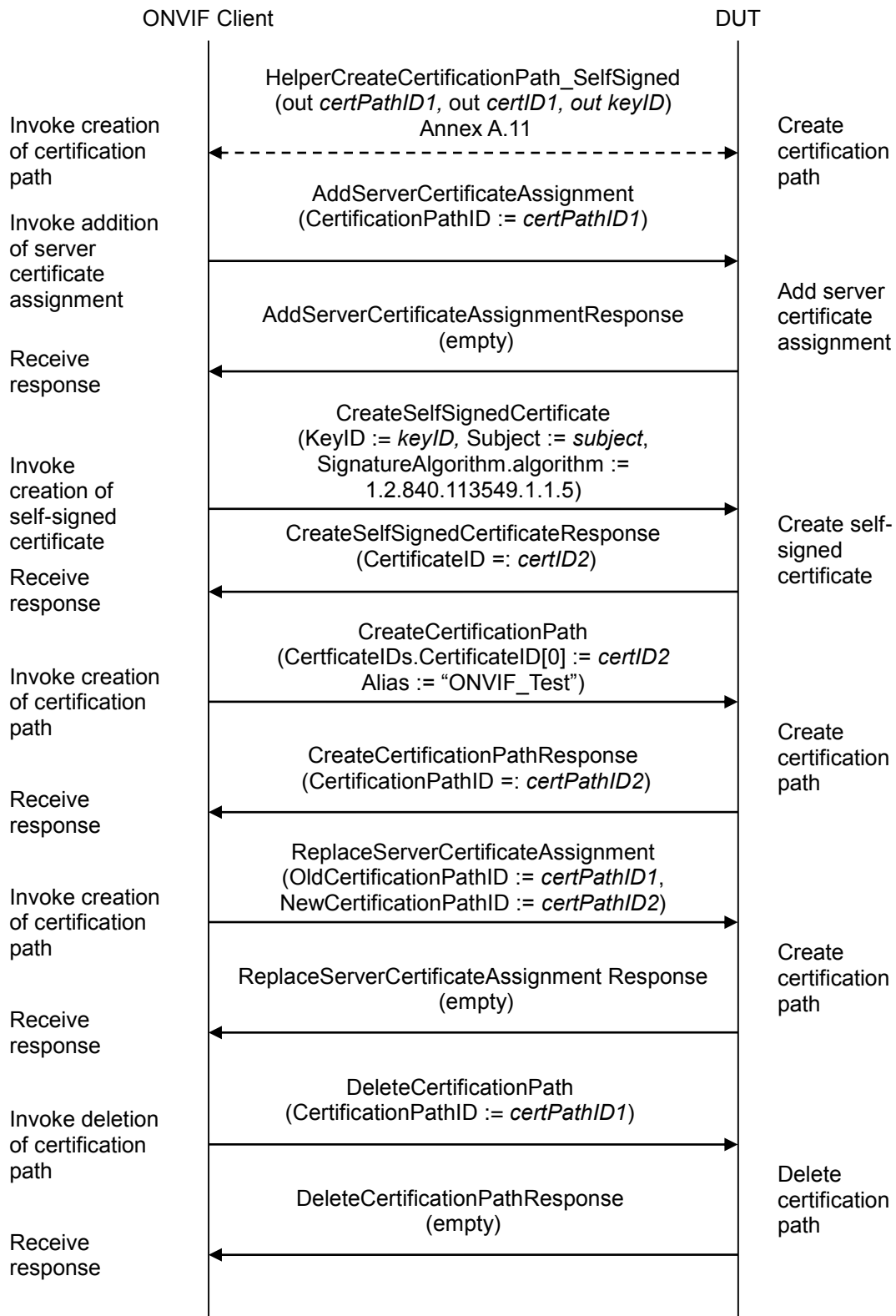
**WSDL Reference:** advancedsecurity.wsdl

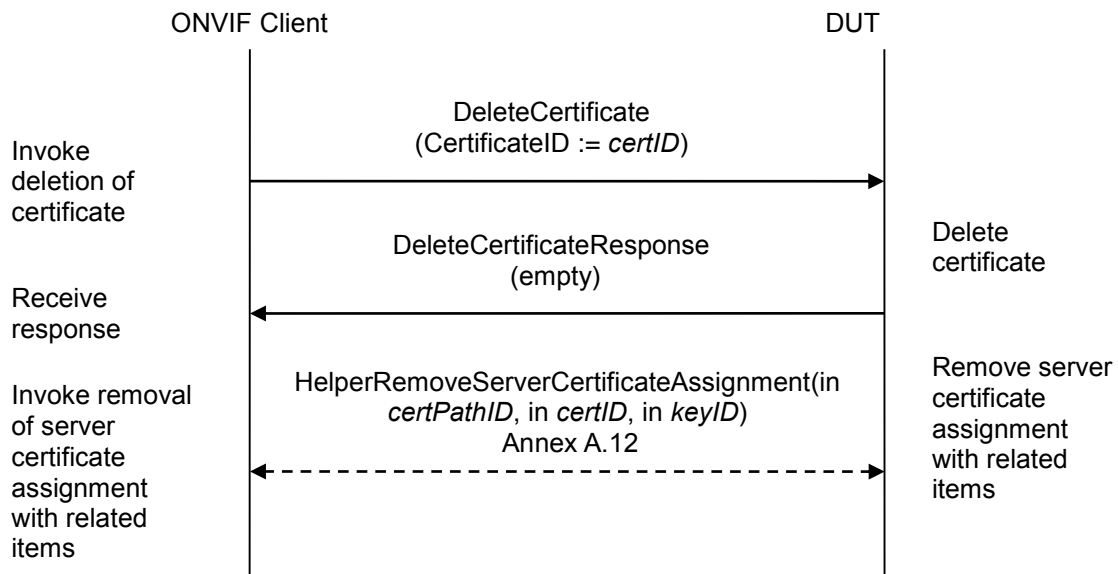
**Test Purpose:** To test the replacement of a self-signed certificate to a TLS server.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for two additional certificates. The DUT shall have enough free storage capacity for two additional certification paths. The DUT shall have enough free storage capacity for one additional server certificate assignment.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





#### Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a certification path (out *certPathID1*) based on self-signed certificate (out *certID1*) and related RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.11.
4. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
  - CertificationPathID := *certPathID1*
5. The DUT responds with an **AddServerCertificateAssignmentResponse** message.
6. ONVIF Client invokes **CreateSelfSignedCertificate** with parameters
  - X509Version skipped
  - KeyID := *keyID*
  - Subject := *subject* (see Annex A.2)
  - Alias skipped
  - notValidBefore skipped
  - notValidAfter skipped
  - SignatureAlgorithm.algorithm := 1.2.840.113549.1.1.5 (OID of SHA-1 with RSA Encryption algorithm)
  - SignatureAlgorithm.parameters skipped



- SignatureAlgorithm.anyParameters skipped
  - Extension skipped
7. The DUT responds with a **CreateSelfSignedCertificateResponse** message with parameters
    - CertificateID := *certID2*
  8. ONVIF Client invokes **CreateCertificationPath** with parameters
    - CertificateIDs.CertificateID[0] := *certID2*
    - Alias := "ONVIF\_Test"
  9. The DUT responds with a **CreateCertificationPathResponse** message with parameters
    - CertificationPathID := *certPathID2*
  10. ONVIF Client invokes **ReplaceServerCertificateAssignment** with parameters
    - OldCertificationPathID := *certPathID1*
    - NewCertificationPathID := *certPathID2*
  11. The DUT responds with a **ReplaceServerCertificateAssignmentResponse** message.
  12. ONVIF Client invokes **DeleteCertificationPath** with parameters
    - CertificationPathID := *certPathID1*
  13. The DUT responds with a **DeleteCertificationPathResponse** message.
  14. ONVIF Client invokes **DeleteCertificate** with parameters
    - CertificateID := *certID1*
  15. The DUT responds with a **DeleteCertificateResponse** message.
  16. ONVIF Client removes server certification assignment and deletes related certification path (*certPathID2*), the self-signed certificate (*certID2*) and the RSA key pair (*keyID*) by following the procedure mentioned in Annex A.12 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **AddServerCertificateAssignmentResponse** message.

The DUT did not send **CreateSelfSignedCertificateResponse** message.

The DUT did not send **CreateCertificationPathResponse** message.

The DUT did not send **ReplaceServerCertificateAssignmentResponse** message.



The DUT did not send **DeleteCertificationPathResponse** message.

The DUT did not send **DeleteCertificateResponse** message.





**4.3.1.4 Replace Server Certificate Assignment – CA**

**Test Label:** Replace Server Certificate Assignment – CA

**Test Case ID:** ADVANCED\_SECURITY-3-1-4

**ONVIF Core Specification Coverage:** Advanced Security, TLS Server

**Command Under Test:** ReplaceServerCertificateAssignment

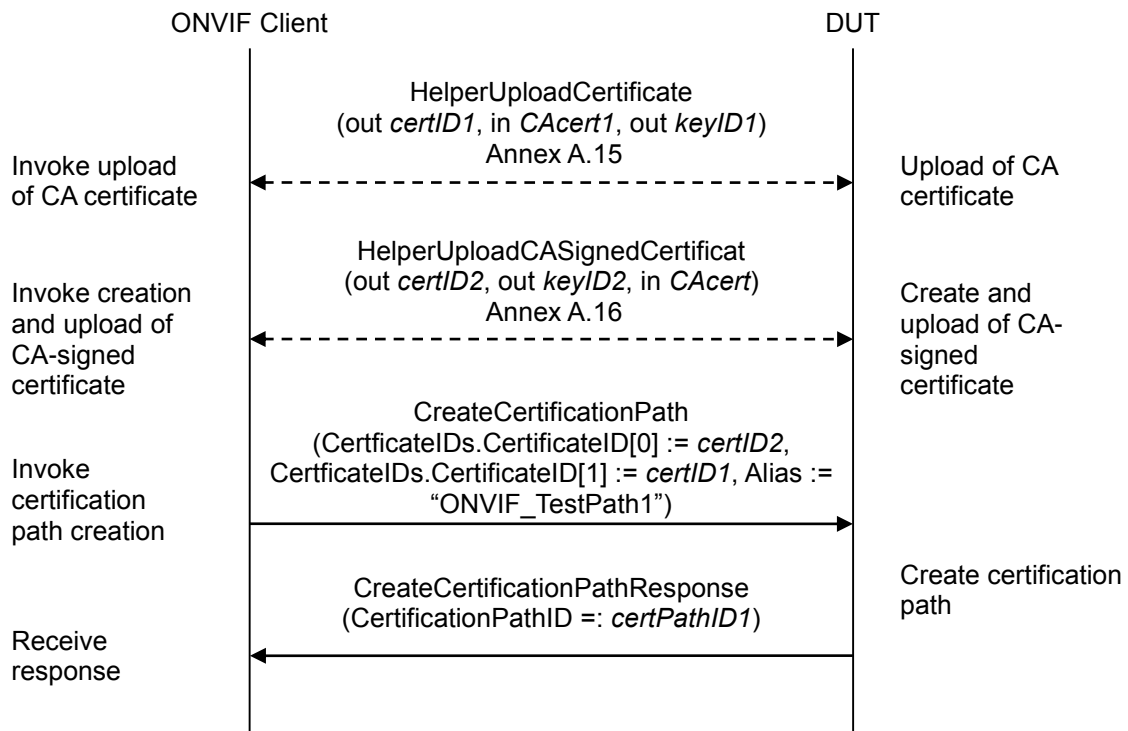
**WSDL Reference:** advancedsecurity.wsdl

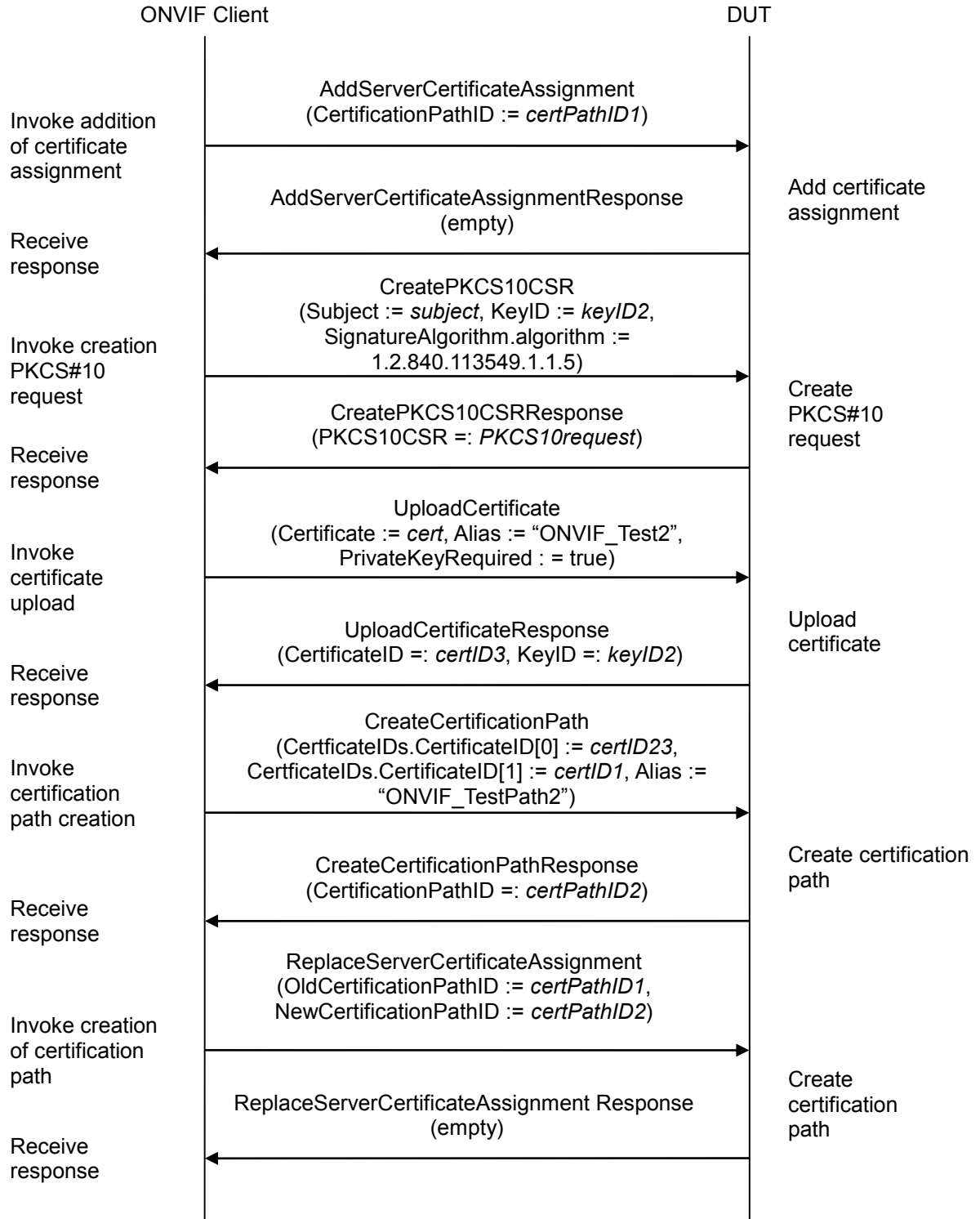
**Test Purpose:** To test the replacement of a signed and CA certificate to a TLS server.

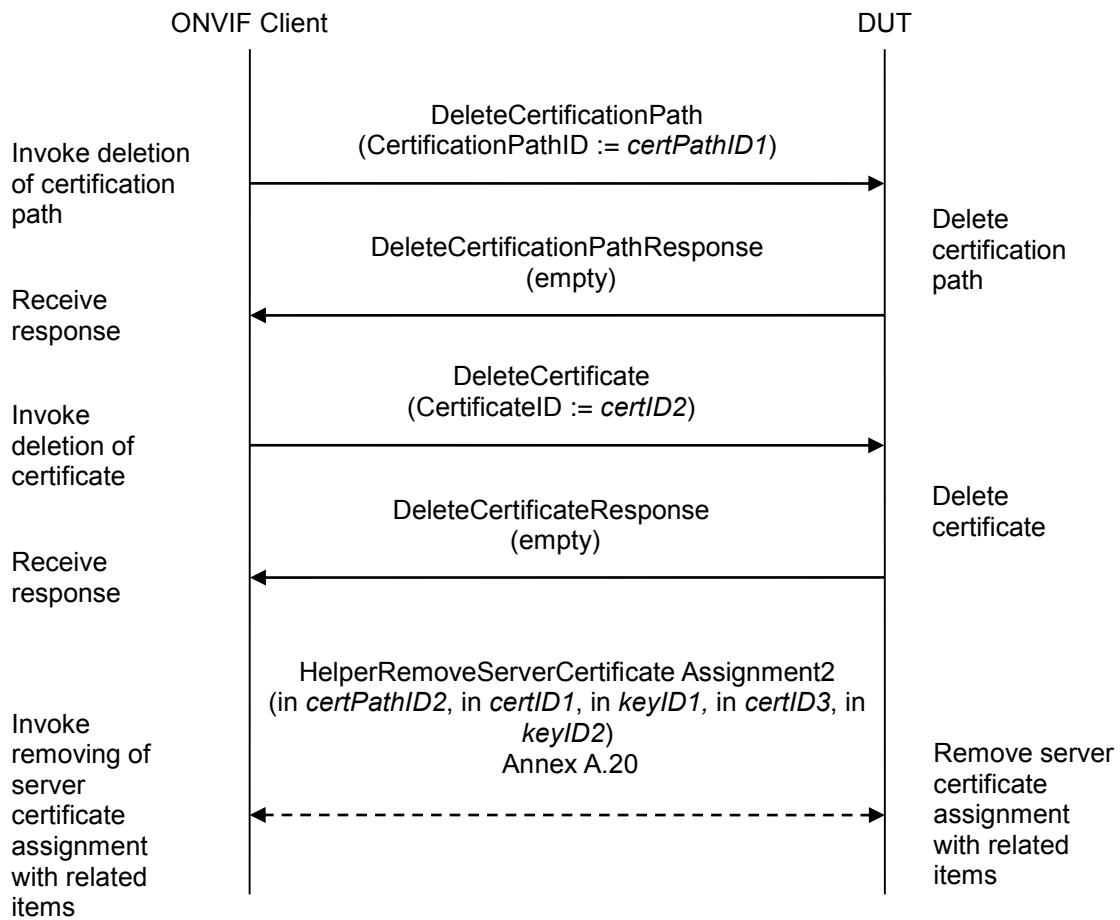
**Pre-requisite:** Advanced Security Service is received from the DUT. Create PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for two additional RSA key pairs. The DUT shall have enough free storage capacity for three additional certificates. The DUT shall have enough free storage capacity for two additional certification paths. The DUT shall have enough free storage capacity for one additional server certificate assignment. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**







**Test Procedure:**

1. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding private key (out *privateKey*) by following the procedure described in Annex A.4.
2. ONVIF Client uploads a CA certificate (out *certID1*, in *CAcert*) and new RSA key pair with the public key from the CA certificate (out *keyID1*) by following the procedure described in Annex A.15.
3. ONVIF Client creates and uploads a CA-signed certificate (out *certID2*) for RSA key pair (out *keyID2*) and associated CA certificate (in *CAcert*) and a corresponding private key (in *privateKey*) by following the procedure described in Annex A.16.
4. ONVIF Client invokes **CreateCertificationPath** with parameters
  - CertificateIDs.CertificateID[0] := *certID2*
  - CertificateIDs.CertificateID[1] := *certID1*
  - Alias := "ONVIF\_TestPath1"
5. The DUT responds with a **CreateCertificationPathResponse** message with parameter



- CertificationPathID =: *certPathID1*
6. ONVIF Client invokes **AddServerCertificateAssignment** with parameter
    - CertificationPathID := *certPathID1*
  7. The DUT responds with an **AddServerCertificateAssignmentResponse** message.
  8. ONVIF Client invokes **CreatePKCS10CSR** with parameters
    - Subject := *subject* (see Annex A.2)
    - KeyID := *keyID2*
    - CSRAttribute skipped
    - SignatureAlgorithm.algorithm := 1.2.840.113549.1.1.5 (OID of SHA-1 with RSA Encryption algorithm)
  9. The DUT responds with **CreatePKCS10CSRResponse** message with parameter
    - PKCS10CSR =: *PKCS10request*
  10. ONVIF Client creates a certificate (out *cert*) from the PKCS#10 request (in *PKCS10request*) and associated CA certificate (in *CACert*) with related private key (in *privateKey*) by following the procedure described in Annex A.3.
  11. ONVIF Client invokes **UploadCertificate** with parameters
    - Certificate := *cert*
    - Alias := "ONVIF\_Test2"
    - PrivateKeyRequired := true
  12. The DUT responds with a **UploadCertificateResponse** message with parameters
    - CertificateID =: *certID3*
    - KeyID =: *keyID2* (the same with step 8)
  13. ONVIF Client invokes **CreateCertificationPath** with parameters
    - CertificateIDs.CertificateID[0] := *certID3*
    - CertificateIDs.CertificateID[1] := *certID1*
    - Alias := "ONVIF\_TestPath2"
  14. The DUT responds with a **CreateCertificationPathResponse** message with parameter
    - CertificationPathID =: *certPathID2*
  15. ONVIF Client invokes **ReplaceServerCertificateAssignment** with parameters
    - OldCertificationPathID := *certPathID1*
    - NewCertificationPathID := *certPathID2*



16. The DUT responds with a **ReplaceServerCertificateAssignmentResponse** message.
17. ONVIF Client invokes **DeleteCertificationPath** with parameter
  - CertificationPathID := *certPathID1*
18. The DUT responds with a **DeleteCertificationPathResponse** message.
19. ONVIF Client invokes **DeleteCertificate** with parameters
  - CertificateID := *certID2*
20. The DUT responds with a **DeleteCertificateResponse** message.
21. ONVIF Client removes server certification assignment and deletes related certification path (in *certPathID2*), related CA certificate (in *certID1*) and RSA key pair (in *keyID1*) and related CA-signed certificate (in *certID3*) and RSA key pair (in *keyID2*) by following the procedure mentioned in Annex A.20 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

- The DUT did not send **AddServerCertificateAssignmentResponse** message.
- The DUT did not send **CreateCertificationPathResponse** message(s).
- The DUT did not send **ReplaceServerCertificateAssignmentResponse** message.
- The DUT did not send **CreatePKCS10CSRResponse** message.
- The DUT did not send **UploadCertificateResponse** message.
- The DUT did not send **DeleteCertificationPathResponse** message.
- The DUT did not send **DeleteCertificateResponse** message.



#### **4.3.1.5 Get Assigned Server Certificates – self-signed**

**Test Label:** Get Assigned Server Certificate - self-signed

**Test Case ID:** ADVANCED\_SECURITY-3-1-5

**ONVIF Core Specification Coverage:** Advanced Security, TLS Server

**Command Under Test:** GetAssignedServerCertificates

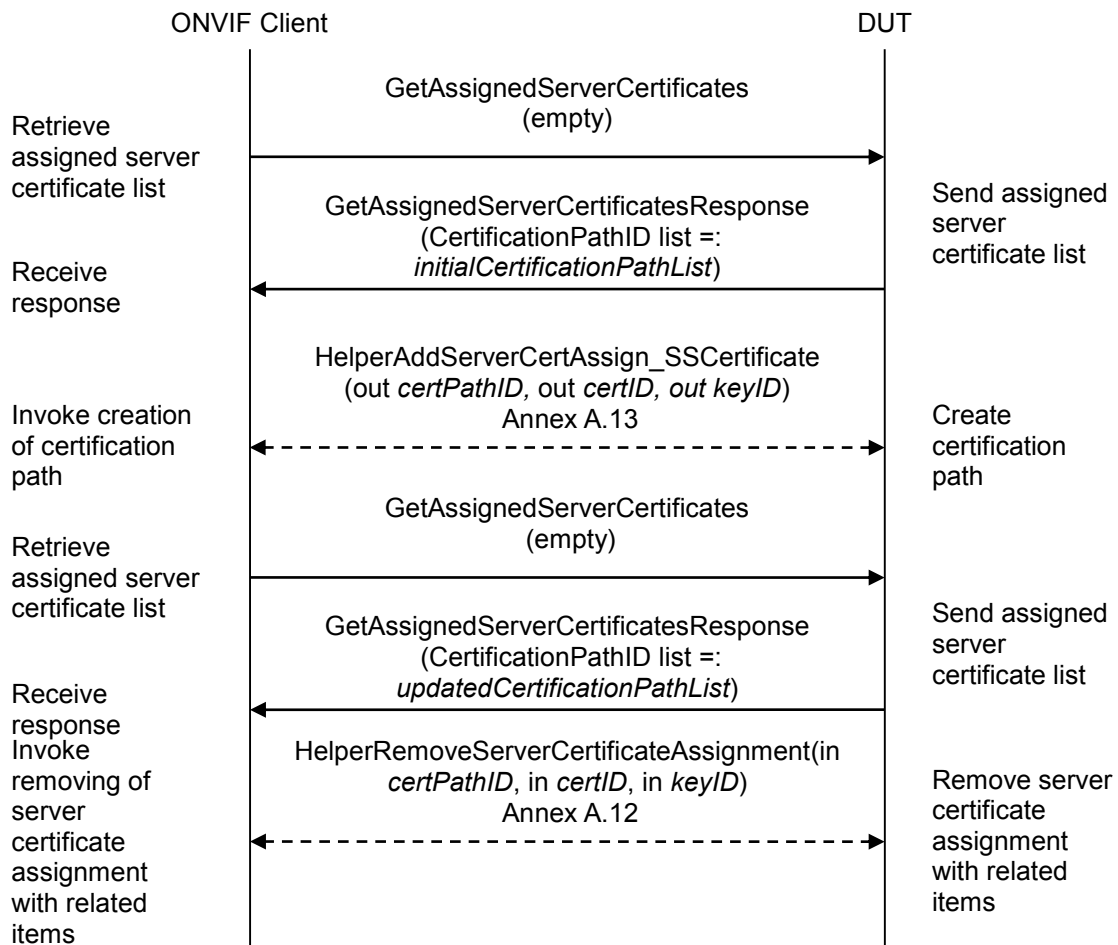
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the retrieval of a self-signed certificate assignment to a TLS server

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional server certificate assignment.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAssignedServerCertificates**.
4. The DUT responds with a **GetAssignedServerCertificatesResponse** message with parameters
  - CertificationPathID list =: *initialCertificationPathList*
5. ONVIF Client adds server certification assignment and creates related certification path (out *certPathID*), the self-signed certificate (out *certID*) and the RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.13.
6. ONVIF Client invokes **GetAssignedServerCertificates**.
7. The DUT responds with a **GetAssignedServerCertificatesResponse** message with parameters
  - CertificationPathID list =: *updatedCertificationPathList*



8. If *updatedCertificationPathList* does not contain *certPathID* and all certification paths from *initialCertificationPathList*, FAIL the test, and go to the step 10.
9. If *updatedCertificationPathList* contains certification paths other than *certPathID* or certification paths from *initialCertificationPathList*, FAIL the test, and go to the step 10.
10. ONVIF Client removes server certification assignment and deletes related certification path (in *certPathID*), self-signed certificate (in *certID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.12 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetAssignedServerCertificatesResponse** message(s).

**Note:** The DUT may return an empty list at step 4.





#### **4.3.1.6 Get Assigned Server Certificates – CA**

**Test Label:** Get Assigned Server Certificate - self-signed

**Test Case ID:** ADVANCED\_SECURITY-3-1-6

**ONVIF Core Specification Coverage:** Advanced Security, TLS Server

**Command Under Test:** GetAssignedServerCertificates

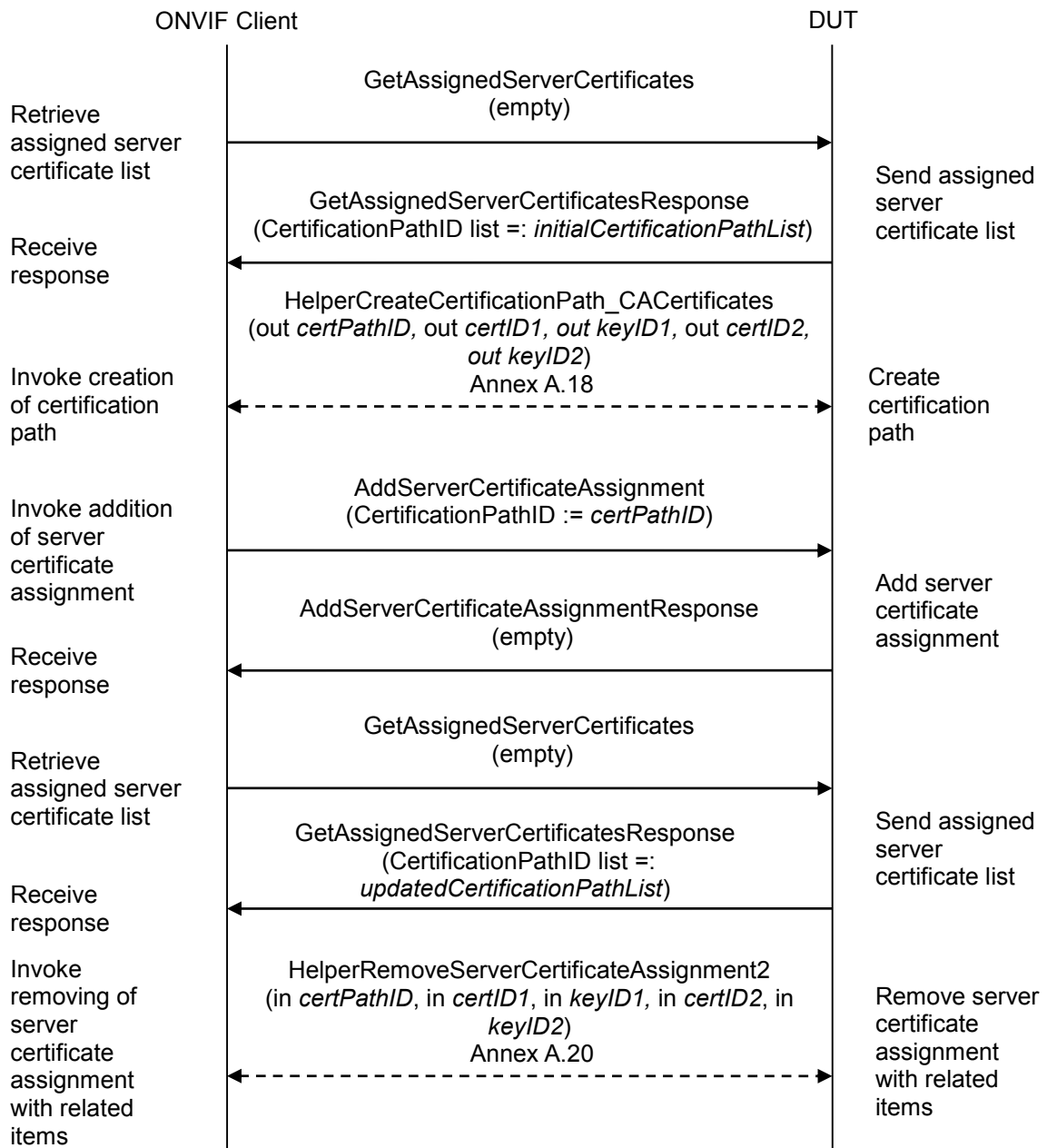
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the retrieval of certificate (signed + CA) assignment to a TLS server.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for two additional RSA key pairs. The DUT shall have enough free storage capacity for two additional certificates. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional server certificate assignment. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAssignedServerCertificates**.
4. The DUT responds with a **GetAssignedServerCertificatesResponse** message with parameters
  - CertificationPathID list =: *initialCertificationPathList*



5. ONVIF Client creates a certification path (out *certPathID*) based on CA-signed certificate (out *certID1*) and related RSA key pair (out *keyID1*) and a corresponding CA certificate (out *certID2*) and related RSA key pair (out *keyID2*) by following the procedure mentioned in Annex A.18.
6. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
  - CertificationPathID := *certPathID*
7. The DUT responds with an **AddServerCertificateAssignmentResponse** message.
8. ONVIF Client invokes **GetAssignedServerCertificates**.
9. The DUT responds with a **GetAssignedServerCertificatesResponse** message with parameters
  - CertificationPathID list =: *updatedCertificationPathList*
10. If *updatedCertificationPathList* does not contain *certPathID* and all certification paths from *initialCertificationPathList*, FAIL the test, and go to the step 12.
11. If *updatedCertificationPathList* contains certification paths other than *certPathID* or certification paths from *initialCertificationPathList*, FAIL the test, and go to the step 12.
12. ONVIF Client removes server certification assignment and deletes related certification path (in *certPathID*), related CA certificate (in *certID2*) and RSA key pair (in *keyID2*) and related CA-signed certificate (in *certID1*) and RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.20 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetAssignedServerCertificatesResponse** message(s).

The DUT did not send **AddServerCertificateAssignmentResponse** messages.

**Note:** The DUT may return an empty list at step 4.



**4.3.1.7 Remove Server Certificate Assignment – self-signed**

**Test Label:** Remove Server Certificate Assignment - Self-Signed

**Test Case ID:** ADVANCED\_SECURITY-3-1-7

**ONVIF Core Specification Coverage:** Advanced Security, TLS Server

**Command Under Test:** RemoveServerCertificateAssignment

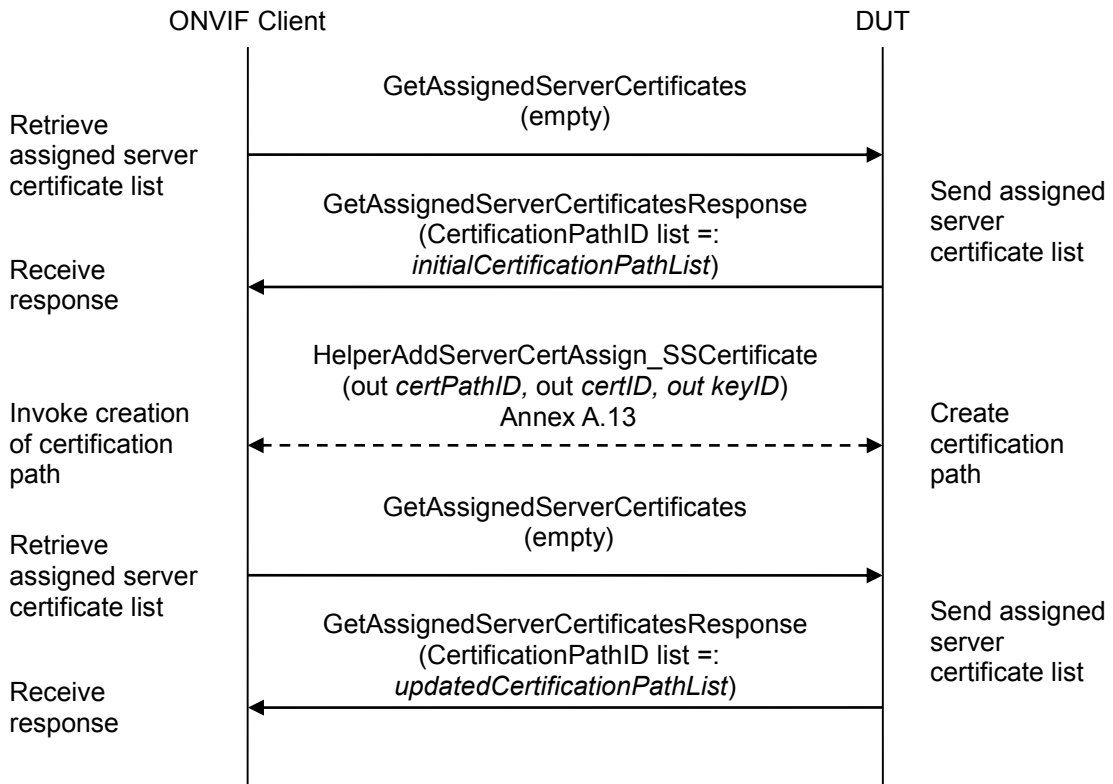
**WSDL Reference:** advancedsecurity.wsdl

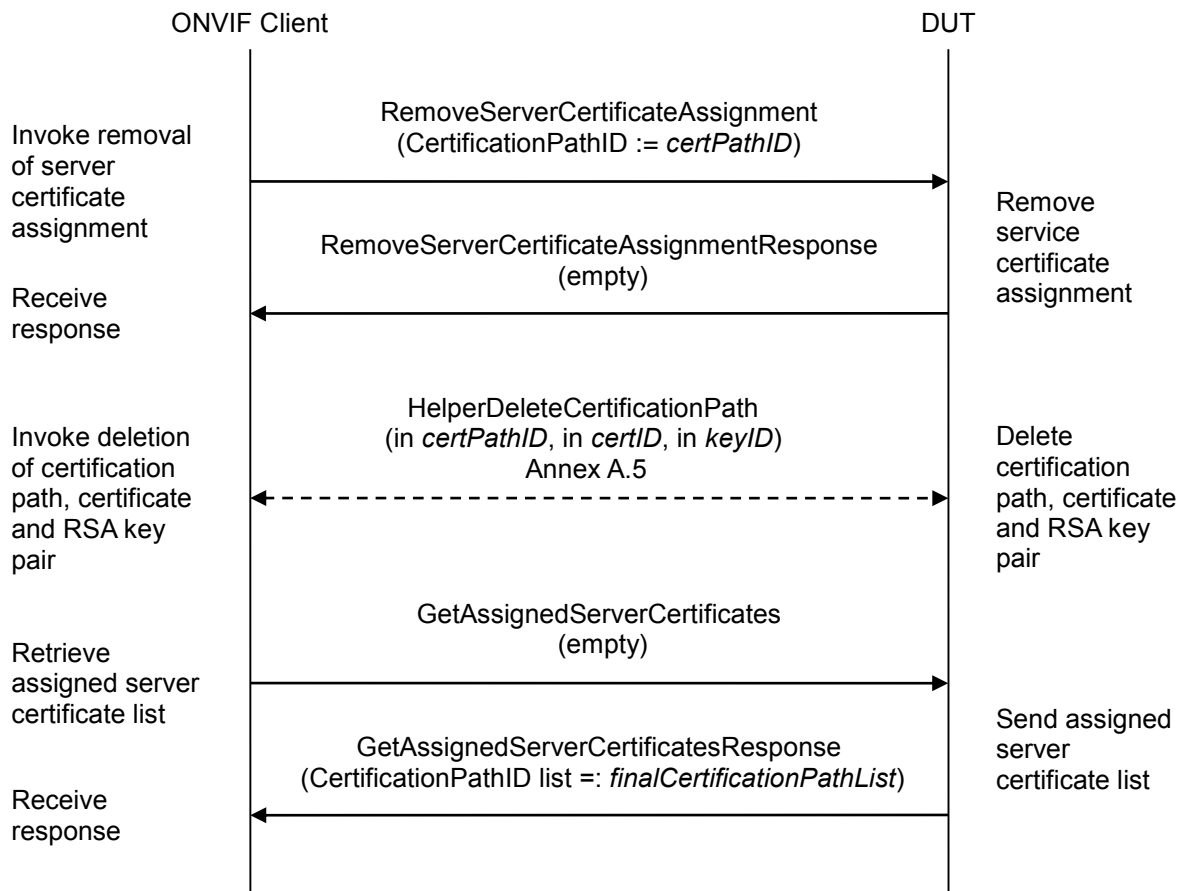
**Test Purpose:** To test the removal of a self-signed certificate assignment on a TLS server.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional server certificate assignment.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAssignedServerCertificates**.
4. The DUT responds with a **GetAssignedServerCertificatesResponse** message with parameters
  - CertificationPathID list =: *initialCertificationPathList*
5. ONVIF Client adds server certification assignment and creates related certification path (out *certPathID*), the self-signed certificate (out *certID*) and the RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.13.
6. ONVIF Client invokes **GetAssignedServerCertificates**.
7. The DUT responds with a **GetAssignedServerCertificatesResponse** message with parameters
  - CertificationPathID list =: *updatedCertificationPathList*
8. If *updatedCertificationPathList* does not contain *certPathID* and all certification paths from *initialCertificationPathList*, FAIL the test, remove server certification assignment and deletes



related certification path (in *certPathID*), self-signed certificate (in *certID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.12 to restore DUT configuration, and skip other steps.

9. If *updatedCertificationPathList* contains certification paths other than *certPathID* or certification paths from *initialCertificationPathList*, FAIL the test, remove server certification assignment and deletes related certification path (in *certPathID*), self-signed certificate (in *certID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.12 to restore DUT configuration, and skip other steps.
10. ONVIF Client invokes **RemoveServerCertificateAssignment** with parameters
  - CertificationPathID := *certPathID*
11. The DUT responds with a **RemoveServerCertificateAssignmentResponse** message.
12. ONVIF Client deletes the certification path (in *certPathID*) and related self-signed certificate (in *certID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.5 to restore DUT configuration.
13. ONVIF Client invokes **GetAssignedServerCertificates**.
14. The DUT responds with a **GetAssignedServerCertificatesResponse** message with parameters
  - CertificationPathID list =: *finalCertificationPathList*
15. If *finalCertificationPathList* is not equal *initialCertificationPathList*, FAIL the test.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **RemoveServerCertificateAssignmentResponse** message.

The DUT did not send **GetAssignedServerCertificatesResponse** messages(s).

**Note:** The DUT may return an empty list at step 4.



#### **4.3.1.8 Remove Server Certificate Assignment – CA**

**Test Label:** Remove Server Certificate Assignment – CA

**Test Case ID:** ADVANCED\_SECURITY-3-1-8

**ONVIF Core Specification Coverage:** Advanced Security, TLS Server

**Command Under Test:** RemoveServerCertificateAssignment

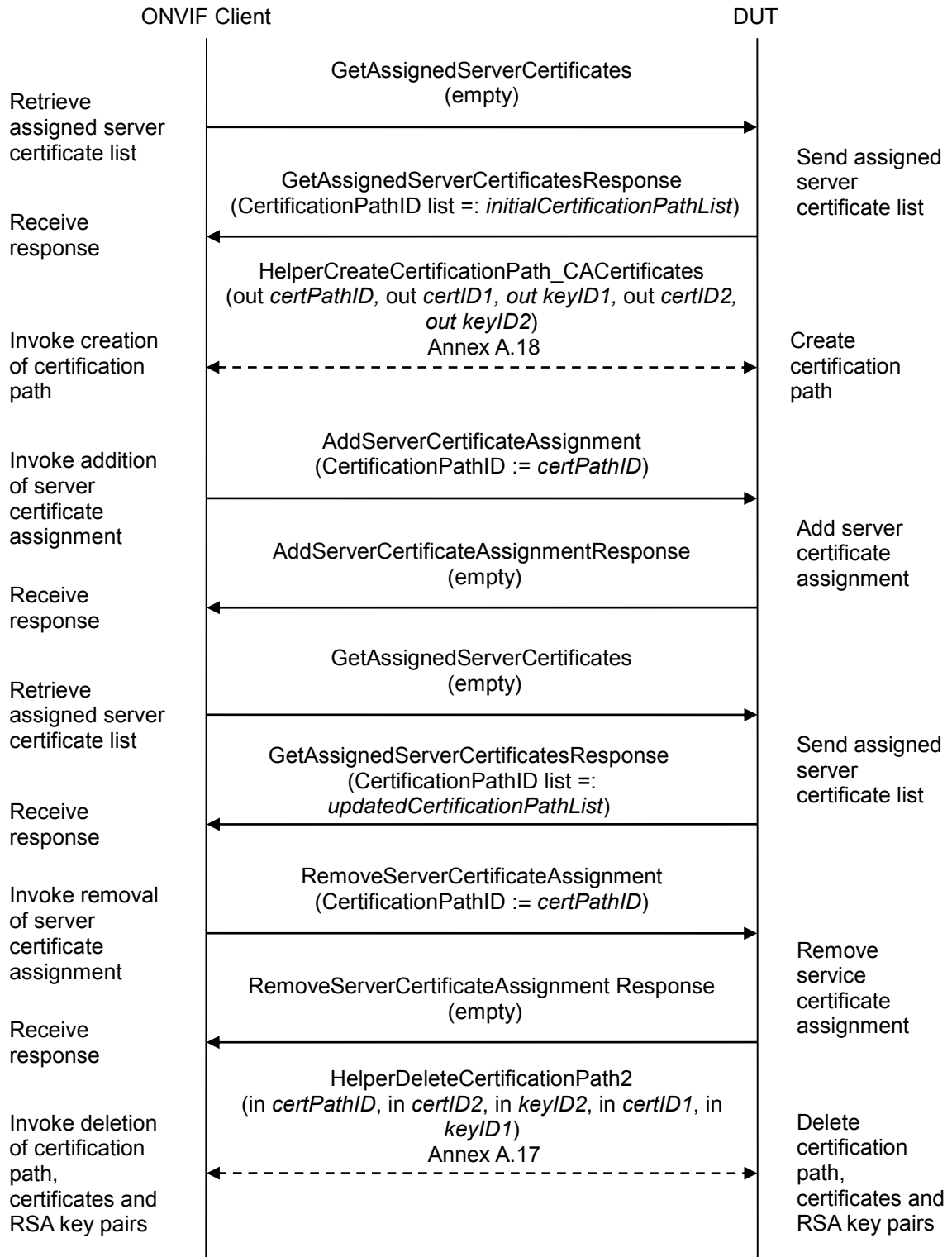
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To test the removal of certificate (signed + CA) assignment to a TLS server.

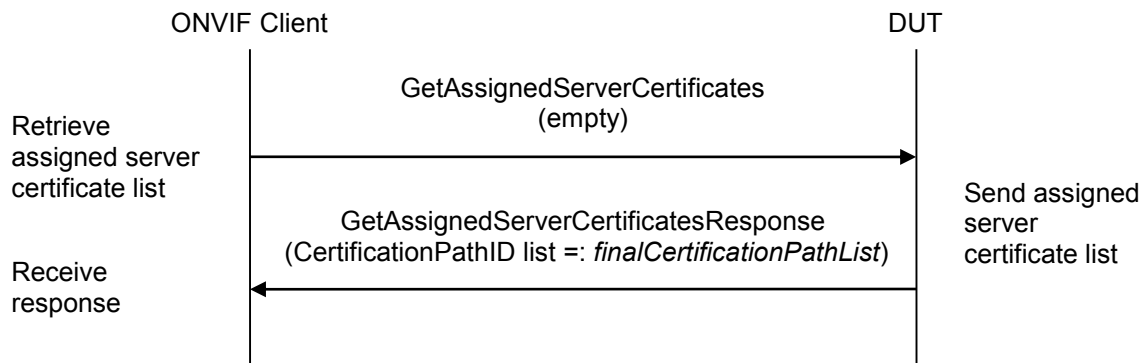
**Pre-requisite:** Advanced Security Service is received from the DUT. Create PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for two additional RSA key pairs. The DUT shall have enough free storage capacity for two additional certificates. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional server certificate assignment. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**







### Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAssignedServerCertificates**.
4. The DUT responds with a **GetAssignedServerCertificatesResponse** message with parameters
  - CertificationPathID list =: *initialCertificationPathList*
5. ONVIF Client creates a certification path (out *certPathID*) based on CA-signed certificate (out *certID1*) and related RSA key pair (out *keyID1*) and a corresponding CA certificate (out *certID2*) and related RSA key pair (out *keyID2*) by following the procedure mentioned in Annex A.18.
6. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
  - CertificationPathID := *certPathID*
7. The DUT responds with an **AddServerCertificateAssignmentResponse** message.
8. ONVIF Client invokes **GetAssignedServerCertificates**.
9. The DUT responds with a **GetAssignedServerCertificatesResponse** message with parameters
  - CertificationPathID list =: *updatedCertificationPathList*
10. If *updatedCertificationPathList* does not contain *certPathID* and all certification paths from *initialCertificationPathList*, FAIL the test, perform steps 12-14 to restore DUT settings, and skip other steps.
11. If *updatedCertificationPathList* contains certification paths other than *certPathID* or certification paths from *initialCertificationPathList*, FAIL the test, perform steps 12-14 to restore DUT settings, and skip other steps.
12. ONVIF Client invokes **RemoveServerCertificateAssignment** with parameters
  - CertificationPathID := *certPathID*
13. The DUT responds with a **RemoveServerCertificateAssignmentResponse** message.



14. ONVIF Client deletes the certification path (in *certPathID*), related CA certificate (in *certID2*) and RSA key pair (in *keyID2*) and related CA-signed certificate (in *certID1*) and RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.17 to restore DUT configuration.
15. ONVIF Client invokes **GetAssignedServerCertificates**.
16. The DUT responds with a **GetAssignedServerCertificatesResponse** message with parameters
  - CertificationPathID list =: *finalCertificationPathList*
17. If *finalCertificationPathList* is not equal to *initialCertificationPathList*, FAIL the test.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **RemoveServerCertificateAssignmentResponse** message.

The DUT did not send **GetAssignedServerCertificatesResponse** messages.

The DUT did not send **AddServerCertificateAssignmentResponse** message.

**Note:** The DUT may return an empty list at step 4.



## **4.3.2 TLS Handshaking**

### **4.3.2.1 Basic TLS Handshake**

**Test Label:** Basic TLS Handshake

**Test Case ID:** ADVANCED\_SECURITY-3-2-3

**ONVIF Core Specification Coverage:** Advanced Security, TLS Server

**Command Under Test:** execute Basic TLS Handshake

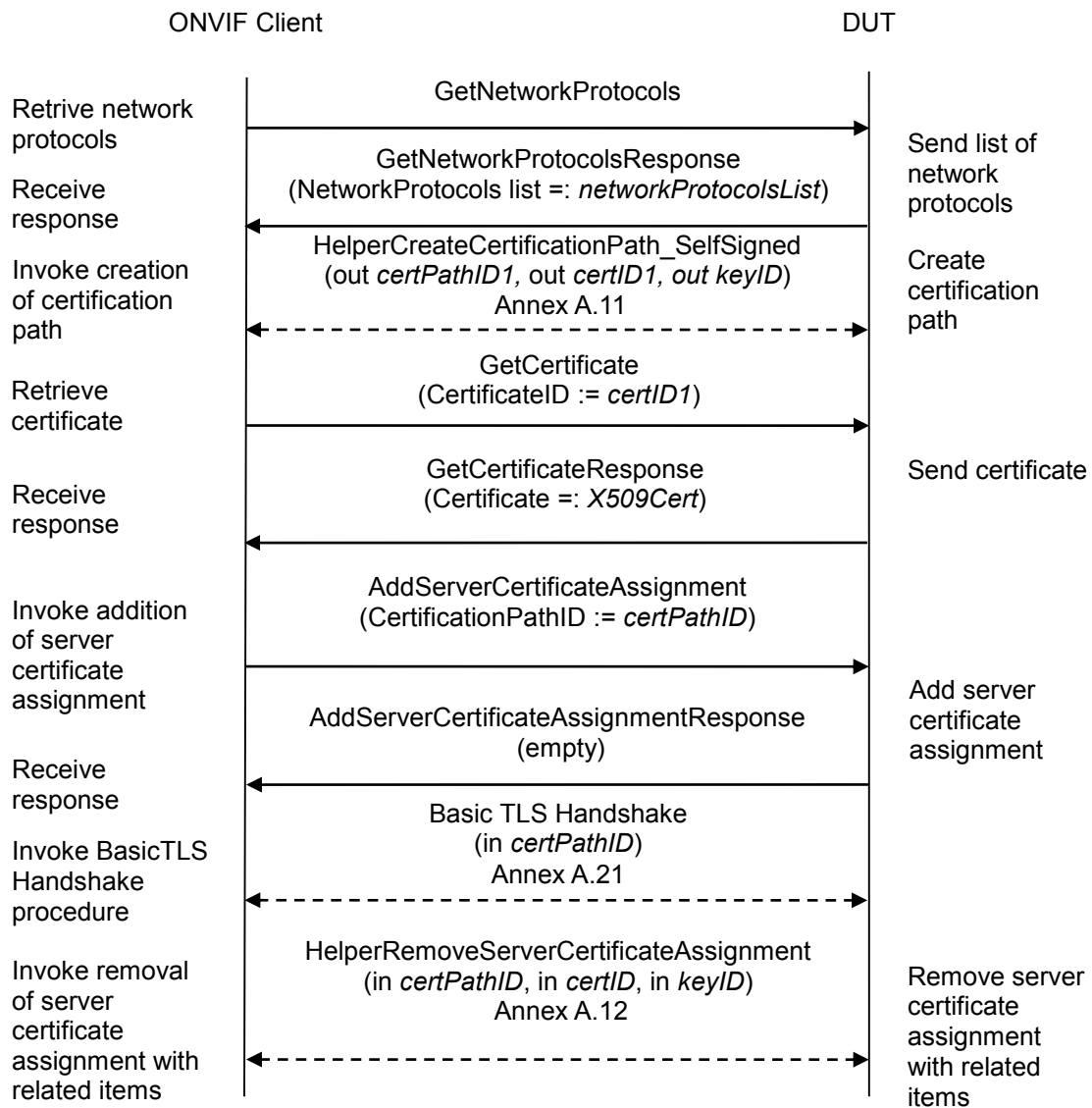
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** Check TLS handshake with certificate.

**Pre-Requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional server certificate assignment.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetNetworkProtocols** to retrieve configured network protocols of the DUT.
4. The DUT responds with a **GetNetworkProtocolsResponse** message with parameters
  - NetworkProtocols list =: *networkProtocolsList*
5. If *networkProtocolsList* does not contain network protocol with NetworkProtocols.Name is equal to "HTTPS", FAIL the test and skip other steps.



6. ONVIF Client creates a certification path (out *certPathID*) based on self-signed certificate (out *certID*) and related RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.11.
7. ONVIF Client invokes **GetCertificate** message with parameters
  - CertificateID := *certID*
8. The DUT responds with a **GetCertificateResponse** message with parameters
  - Certificate =: *X509Cert*
9. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
  - CertificationPathID := *certPathID*
10. The DUT responds with an **AddServerCertificateAssignmentResponse** message.
11. If HTTPS protocol with NetworkProtocols.Name is equal to "HTTPS" from *networkProtocolsList* has NetworkProtocols.Enabled equal to true, go to the step 15.
12. ONVIF Client invokes **SetNetworkProtocols** message with parameters
  - NetworkProtocols[0].Name := HTTPS
  - NetworkProtocols[0].Enabled := true
  - NetworkProtocols[0].Port := 443
  - NetworkProtocols[0].Extension skipped
13. The DUT responds with a **SetNetworkProtocolsResponse** message.
14. ONVIF Client waits for time *timeout1*.
15. ONVIF Client checks that HTTPS protocol Port is open. If HTTPS protocol *portHTTPS* is not open, FAIL the test and go to the step 17.
16. ONVIF Client verifies basic TLS handshake with expecting Server Certificate (in *certPathID*) using specified port (in *portHTTPS*) by following the procedure mentioned in Annex A.21.
17. If HTTPS protocol with NetworkProtocols.Name is equal to "HTTPS" from *networkProtocolsList* has NetworkProtocols.Enabled equal to true, go to the step 21.
18. ONVIF Client invokes **SetNetworkProtocols** message with parameters
  - NetworkProtocols list := *networkProtocolsList*
19. The DUT responds with a **SetNetworkProtocolsResponse** message.
20. ONVIF Client waits for time *timeout1*.
21. ONVIF Client removes server certification assignment and deletes related certification path (in *certPathID*), self-signed certificate (in *certID*) and the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.12 to restore DUT configuration.

**Test Result:**



**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetNetworkProtocolsResponse** message.

The DUT did not send **GetCertificateResponse** message.

The DUT did not send **AddServerCertificateAssignmentResponse** message.

The DUT did not provide Basic TLS handshake at Step 16.

**Note:** The corresponding to HTTPS port number (*portHTTPS*) from *networkProtocolsList* shall be used in case HTTPS protocol was enabled in *networkProtocolsList*. Otherwise, 443 port number shall be used.

**Note:** If the DUT presents Certificate which is not equal to *X509Cert* during the Annex A.21 execution, log WARNING message.

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.



#### **4.3.2.2 Basic TLS Handshake after Replace Server Certificate Assignment**

**Test Label:** Basic TLS Handshake after Replace Server Certificate Assignment

**Test Case ID:** ADVANCED\_SECURITY-3-2-4

**ONVIF Core Specification Coverage:** Advanced Security, TLS Server

**Command Under Test:** ReplaceServerCertificateAssignment

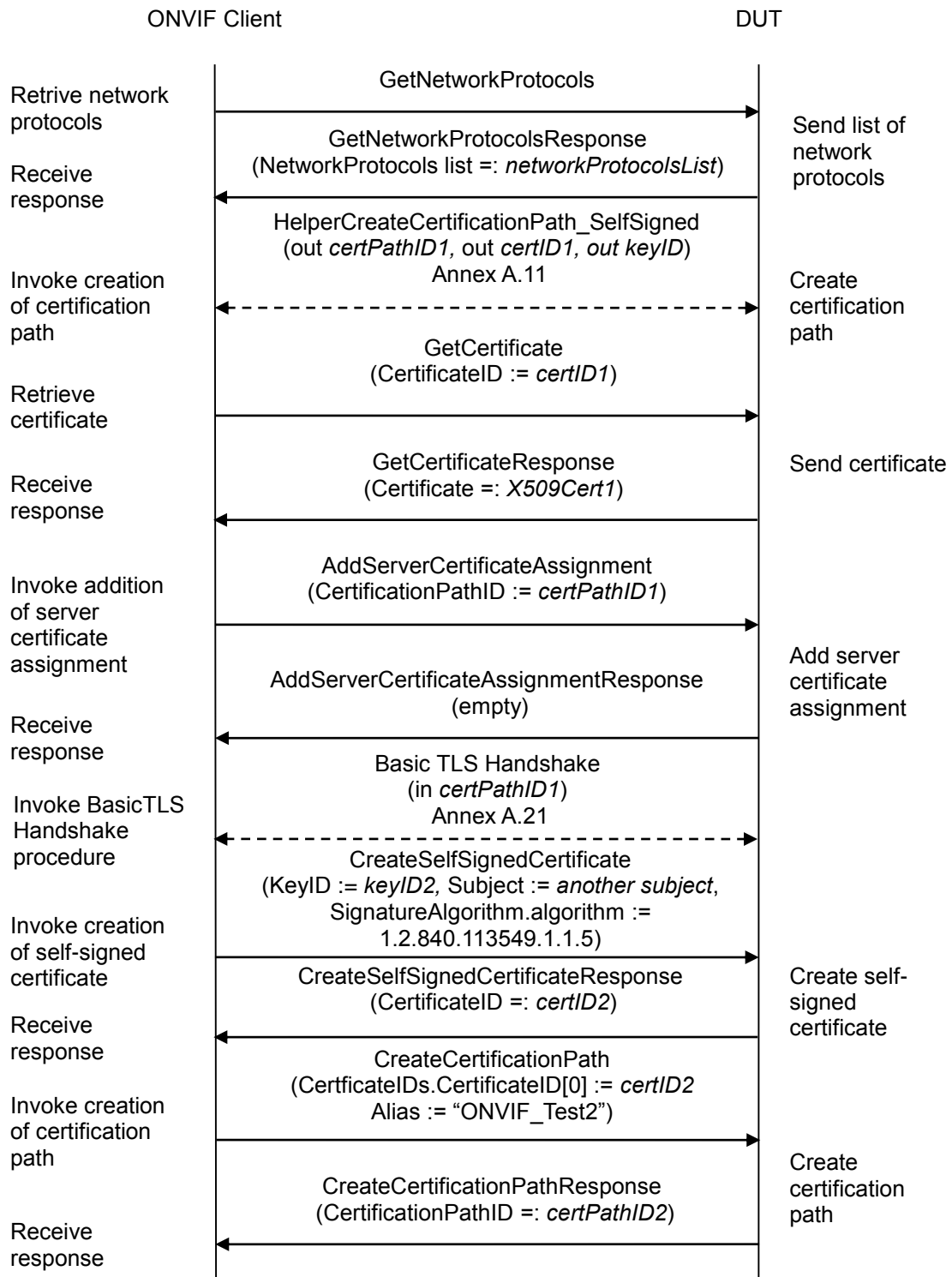
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** Check TLS handshake with replaced certificate.

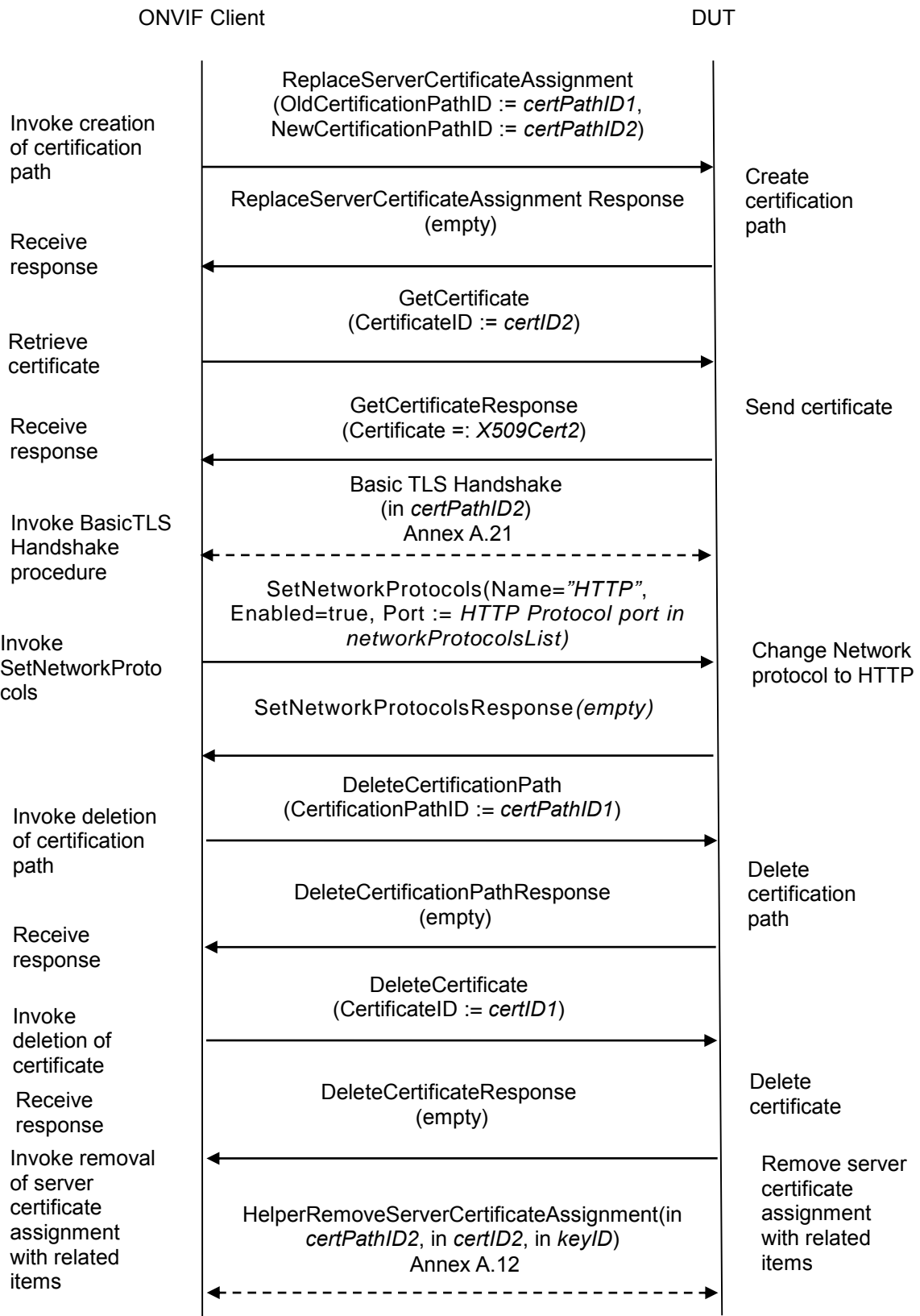
**Pre-Requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for two additional certificates. The DUT shall have enough free storage capacity for two additional certification paths. The DUT shall have enough free storage capacity for one additional server certificate assignment. There is no server certificate assignment at the device.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**









**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetNetworkProtocols** to retrieve configured network protocols of the DUT.
4. The DUT responds with a **GetNetworkProtocolsResponse** message with parameters
  - NetworkProtocols list =: *networkProtocolsList*
5. If *networkProtocolsList* does not contain network protocol with NetworkProtocols.Name is equal to "HTTPS", FAIL the test and skip other steps.
6. ONVIF Client creates a certification path (out *certPathID1*) based on self-signed certificate (out *certID1*) and related RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.11.
7. ONVIF Client invokes **GetCertificate** message with parameters
  - CertificateID := *certID1*
8. The DUT responds with a **GetCertificateResponse** message with parameters
  - Certificate =: *X509Cert1*
9. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
  - CertificationPathID := *certPathID1*
10. The DUT responds with an **AddServerCertificateAssignmentResponse** message.
11. If HTTPS protocol with NetworkProtocols.Name is equal to "HTTPS" from *networkProtocolsList* has NetworkProtocols.Enabled equal to true, go to the step 15.
12. ONVIF Client invokes **SetNetworkProtocols** message with parameters
  - NetworkProtocols[0].Name := HTTPS
  - NetworkProtocols[0].Enabled := true
  - NetworkProtocols[0].Port := 443
  - NetworkProtocols[0].Extension skipped
13. The DUT responds with a **SetNetworkProtocolsResponse** message.
14. ONVIF Client waits for time *timeout1*.
15. ONVIF Client checks that HTTPS protocol Port is open. If HTTPS protocol *portHTTPS* is not open, FAIL the test and go to the step 36.
16. ONVIF Client verifies basic TLS handshake with expecting Server Certificate (in *certPathID1*) using specified port (in *portHTTPS*) by following the procedure mentioned in Annex A.21.
17. ONVIF Client invokes CreateSelfSignedCertificate with parameters



- X509Version skipped
  - KeyID := *keyID*
  - Subject := *subject* (another than in *certID1*)
  - Alias skipped
  - notValidBefore skipped
  - notValidAfter skipped
  - SignatureAlgorithm.algorithm := 1.2.840.113549.1.1.5 (OID of SHA-1 with RSA Encryption algorithm)
  - SignatureAlgorithm.parameters skipped
  - SignatureAlgorithm.anyParameters skipped
  - Extension skipped
18. The DUT responds with a **CreateSelfSignedCertificateResponse** message with parameters
- CertificateID =: *certID2*
19. ONVIF Client invokes **CreateCertificationPath** with parameters
- CertificateIDs.CertificateID[0] := *certID2*
  - Alias := "ONVIF\_Test2"
20. The DUT responds with a **CreateCertificationPathResponse** message with parameters
- CertificationPathID =: *certPathID2*
21. ONVIF Client invokes **ReplaceServerCertificateAssignment** with parameters
- OldCertificationPathID := *certPathID1*
  - NewCertificationPathID := *certPathID2*
22. The DUT responds with a **ReplaceServerCertificateAssignmentResponse** message.
23. ONVIF Client waits for time *timeout1*.
24. ONVIF Client invokes **GetCertificate** message with parameters
- CertificateID := *certID2*
25. The DUT responds with a **GetCertificateResponse** message with parameters
- Certificate =: *X509Cert2*
26. ONVIF Client verifies basic TLS handshake with expecting Server Certificate (in *certPathID2*) using specified port (in *portHTTPS*) by following the procedure mentioned in Annex A.21.
27. ONVIF Client selects network protocol with Name ="HTTP" in *networkProtocolsList* (received in step 4) and saves this protocol in *HTTPProtocol* variable.



28. ONVIF Client invokes **SetNetworkProtocols** message with parameters
  - NetworkProtocols[0].Name := *HTTPProtocol.Name*
  - NetworkProtocols[0].Enabled := true
  - NetworkProtocols[0].Port := *HTTPProtocol.port*
  - NetworkProtocols[0].Extension := *HTTPProtocol.Extension*
29. The DUT responds with a **SetNetworkProtocolsResponse** message.
30. ONVIF Client invokes DeleteCertificationPath with parameters
  - CertificationPathID := *certPathID1*
31. The DUT responds with a **DeleteCertificationPathResponse** message.
32. ONVIF Client invokes DeleteCertificate with parameters
  - CertificateID := *certID1*
33. The DUT responds with a **DeleteCertificateResponse** message.
34. ONVIF Client invokes **SetNetworkProtocols** message with parameters
  - NetworkProtocols list := *networkProtocolsList*
35. The DUT responds with a **SetNetworkProtocolsResponse** message.
36. ONVIF Client waits for time *timeout1*.
37. ONVIF Client removes server certification assignment and deletes related certification path (*certPathID2*), the self-signed certificate (*certID2*) and the RSA key pair (*keyID*) by following the procedure mentioned in Annex A.12 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetNetworkProtocolsResponse** message.

The DUT did not send **GetCertificateResponse** messages.

The DUT did not send **AddServerCertificateAssignmentResponse** messages.

The DUT did not send **ReplaceServerCertificateAssignmentResponse** message.

The DUT did not send **DeleteCertificationPathResponse** message.

The DUT did not send **DeleteCertificateResponse** message.

The DUT did not provide Basic TLS handshake at Step 16.



The DUT did not provide Basic TLS handshake at Step 25.

**Note:** The corresponding to HTTPS port number (*portHTTPS*) from *networkProtocolsList* shall be used in case HTTPS protocol was enabled in *networkProtocolsList*. Otherwise, 443 port number shall be used.

**Note:** If the DUT presents Certificate which is not equal to *X509Cert* during the Annex A.21 execution, log WARNING message.

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.



**4.3.2.3 Basic TLS Handshake with Replace Server Certification Path and PKCS#12**

**Test Label:** Basic TLS Handshake with Replace Server Certification Path and PKCS#12

**Test Case ID:** ADVANCED\_SECURITY-3-2-5

**ONVIF Core Specification Coverage:** Advanced Security, TLS Server

**Command Under Test:** UploadCertificateWithPrivateKeyInPKCS12, AddServerCertificateAssignment, GetAssignedServerCertificates, ReplaceServerCertificateAssignment

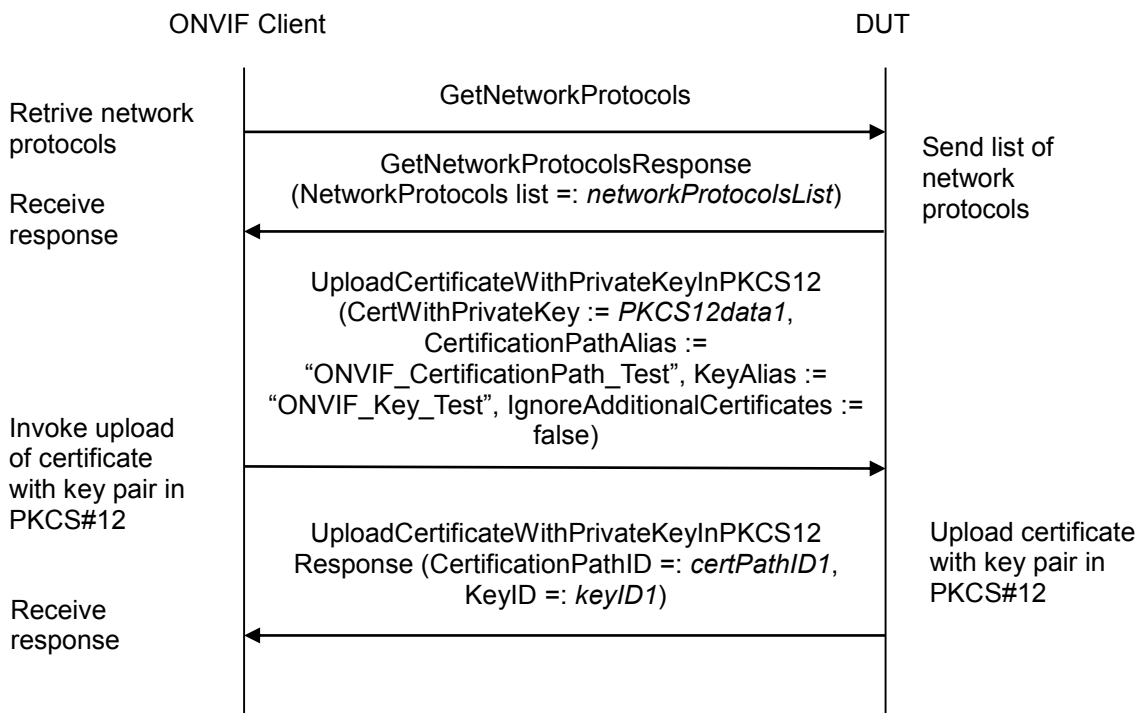
**WSDL Reference:** advancedsecurity.wsdl

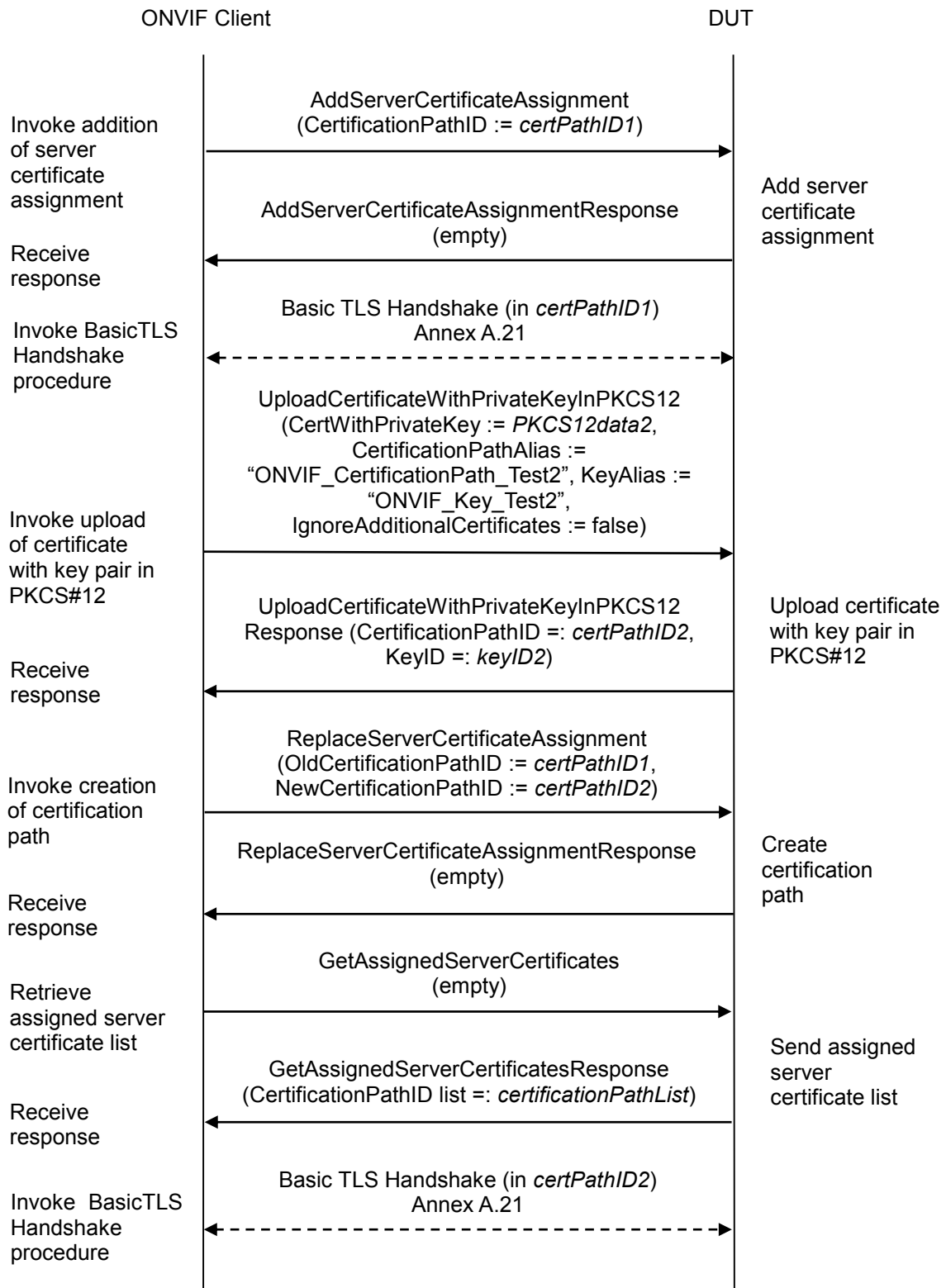
**Test Purpose:** Verify that DUT correctly performs TLS handshake after replace of server certification path with PKCS#12.

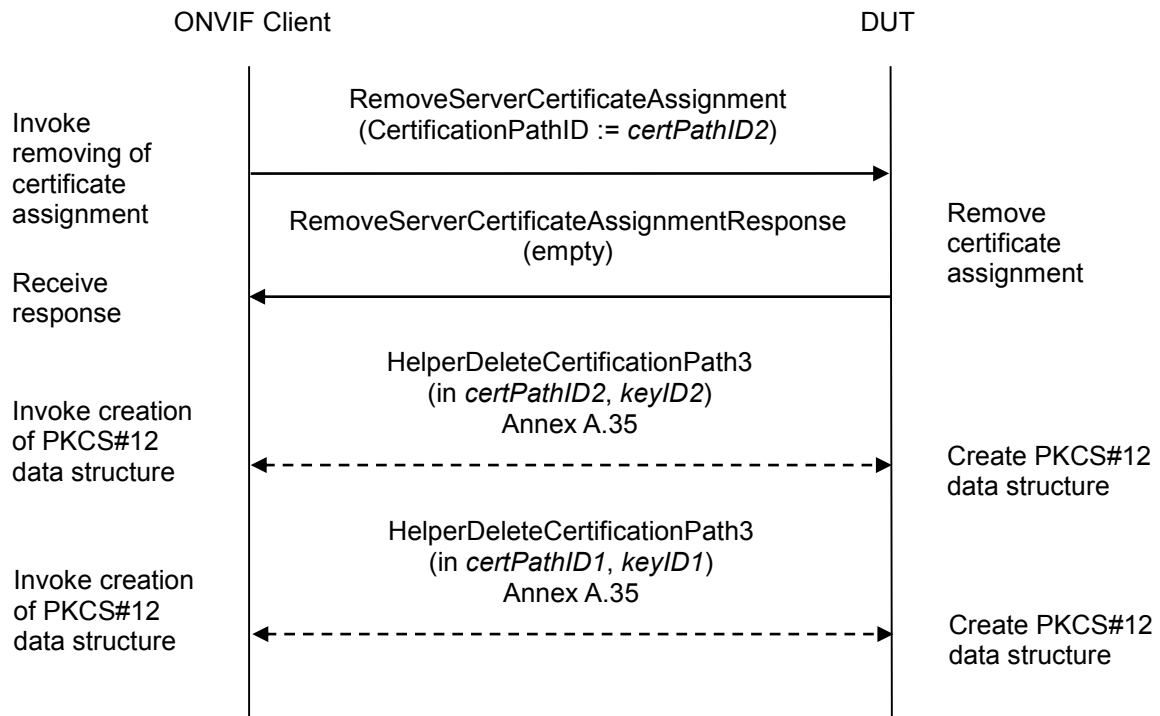
**Pre-Requirement:** Advanced Security Service is received from the DUT. Certificate along with an RSA private key in a PKCS#12 data structure upload is supported by the DUT as indicated by the PKCS12CertificateWithRSAPrivateKeyUpload capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for two additional certificates. The DUT shall have enough free storage capacity for two additional certification paths. The DUT shall have enough free storage capacity for one additional server certificate assignment. There is no server certificate assignment at the device.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**







**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetNetworkProtocols** to retrieve configured network protocols of the DUT.
4. The DUT responds with a **GetNetworkProtocolsResponse** message with parameters
  - NetworkProtocols list := *networkProtocolsList*
5. If *networkProtocolsList* does not contain network protocol with NetworkProtocols.Name is equal to "HTTPS", FAIL the test and skip other steps.
6. Set
  - *portHTTPS* := *networkProtocolsList*.NetworkProtocols[Name = "HTTPS"].Port
7. ONVIF Client creates a CA certificate (out *CAcert1*) and a corresponding public key (out *publicKey1*) in the certificate along with the corresponding private key (out *privateKey1*) in the form of a PKCS#12 file (out *PKCS12data1*) by following the procedure described in Annex A.30.
8. ONVIF Client invokes **UploadCertificateWithPrivateKeyInPKCS12** with parameters
  - CertWithPrivateKey := *PKCS12data1*
  - CertificationPathAlias := "ONVIF\_CertificationPath\_Test"





- KeyAlias := "ONVIF\_Key\_Test"
  - IgnoreAdditionalCertificates := false
  - IntegrityPassphraseID skipped
  - EncryptionPassphraseID skipped
9. The DUT responds with an **UploadCertificateWithPrivateKeyInPKCS12Response** message with parameters
- CertificationPathID =: *certPathID1*
  - KeyID =: *keyID1*
10. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
- CertificationPathID := *certPathID1*
11. The DUT responds with an **AddServerCertificateAssignmentResponse** message.
12. If HTTPS protocol with NetworkProtocols.Name is equal to "HTTPS" from *networkProtocolsList* has NetworkProtocols.Enabled equal to true, go to the step 17.
13. ONVIF Client invokes **SetNetworkProtocols** message with parameters
- NetworkProtocols[0].Name := HTTPS
  - NetworkProtocols[0].Enabled := true
  - NetworkProtocols[0].Port := 443
  - NetworkProtocols[0].Extension skipped
14. The DUT responds with a **SetNetworkProtocolsResponse** message.
15. ONVIF Client waits for time *timeout1*.
16. Set
- *portHTTPS* := 443
17. ONVIF Client checks that HTTPS protocol Port is open. If HTTPS protocol *portHTTPS* is not open, FAIL the test and go to the step 35.
18. ONVIF Client verifies basic TLS handshake with expecting Server Certificate (in *certPathID1*) using specified port (in *portHTTPS*) by following the procedure mentioned in Annex A.21.
19. If the DUT presents Certificate which is not equal to *PKCS12data1* during the Annex A.21 execution, FAIL the test and go to the step 35.
20. ONVIF Client creates a CA certificate (out *CAcert2*) with subject (in "CN=ONVIF TT2,C=US") and a corresponding public key (out *publicKey2*) in the certificate along with the corresponding private key (out *privateKey2*) in the form of a PKCS#12 file (out *PKCS12data2*) by following the procedure described in Annex A.30.
21. ONVIF Client invokes **UploadCertificateWithPrivateKeyInPKCS12** with parameters



- CertWithPrivateKey := *PKCS12data2*
  - CertificationPathAlias := "ONVIF\_CertificationPath\_Test2"
  - KeyAlias := "ONVIF\_Key\_Test2"
  - IgnoreAdditionalCertificates := false
  - IntegrityPassphraseID skipped
  - EncryptionPassphraseID skipped
22. The DUT responds with an **UploadCertificateWithPrivateKeyInPKCS12Response** message with parameters
- CertificationPathID =: *certPathID2*
  - KeyID =: *keyID2*
23. ONVIF Client invokes **ReplaceServerCertificateAssignment** with parameters
- OldCertificationPathID := *certPathID1*
  - NewCertificationPathID := *certPathID2*
24. The DUT responds with a **ReplaceServerCertificateAssignmentResponse** message.
25. ONVIF Client invokes **GetAssignedServerCertificates**.
26. The DUT responds with a **GetAssignedServerCertificatesResponse** message with parameters
- CertificationPathID list =: *certificationPathIDList*
27. If *certificationPathIDList* contains certification path with CertificationPathID equal to *certPathID1*, FAIL the test and go to the step 34.
28. If *certificationPathIDList* does not contain certification path with CertificationPathID equal to *certPathID2*, FAIL the test and go to the step 34.
29. ONVIF Client checks that HTTPS protocol Port is open. If HTTPS protocol *portHTTPS* is not open, FAIL the test and go to the step 32.
30. ONVIF Client verifies basic TLS handshake with expecting Server Certificate (in *certPathID2*) using specified port (in *portHTTPS*) by following the procedure mentioned in Annex A.21.
31. If the DUT presents Certificate which is not equal to *X509Cert* during the Annex A.21 execution, FAIL the test and go to the step 32.
32. ONVIF Client invokes **RemoveServerCertificateAssignment** with parameters
- CertificationPathID := *certPathID2*
33. The DUT responds with a **RemoveServerCertificateAssignmentResponse** message.
34. ONVIF Client deletes the certification path (in *certPathID2*) and RSA key pair (in *keyID2*) by following the procedure mentioned in Annex A.35 to restore DUT configuration.
35. ONVIF Client deletes the certification path (in *certPathID1*) and RSA key pair (in *keyID1*) by



following the procedure mentioned in Annex A.35 to restore DUT configuration.

36. ONVIF Client invokes **SetNetworkProtocols** message with parameters

- NetworkProtocols := *networkProtocolsList*

37. The DUT responds with a **SetNetworkProtocolsResponse** message.

38. ONVIF Client waits for time *timeout1*.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetNetworkProtocolsResponse** message.

The DUT did not send **SetNetworkProtocolsResponse** message.

The DUT did not send **UploadCertificateWithPrivateKeyInPKCS12Response** messages.

The DUT did not send **AddServerCertificateAssignmentResponse** messages.

The DUT did not send **ReplaceServerCertificateAssignmentResponse** message.

The DUT did not send **GetAssignedServerCertificatesResponse** message.

The DUT did not provide Basic TLS handshake at Step 18.

The DUT did not provide Basic TLS handshake at Step 30.

**Note:** The corresponding to HTTPS port number (*portHTTPS*) from *networkProtocolsList* shall be used in case HTTPS protocol was enabled in *networkProtocolsList*. Otherwise, 443 port number shall be used.

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.



**4.3.3 TLS client authentication**

**4.3.3.1 TLS client authentication – self-signed TLS server certificate with on-device RSA key pair**

**Test Label:** Verify TLS Client Authentication – Self-Signed TLS Server Certificate with On-Device RSA Key Pair

**Test Case ID:** ADVANCED\_SECURITY-3-3-1

**ONVIF Core Specification Coverage:** Advanced Security, TLS Server

**Command Under Test:** GetClientAuthenticationRequired, SetClientAuthenticationRequired, AddCertPathValidationPolicyAssignment

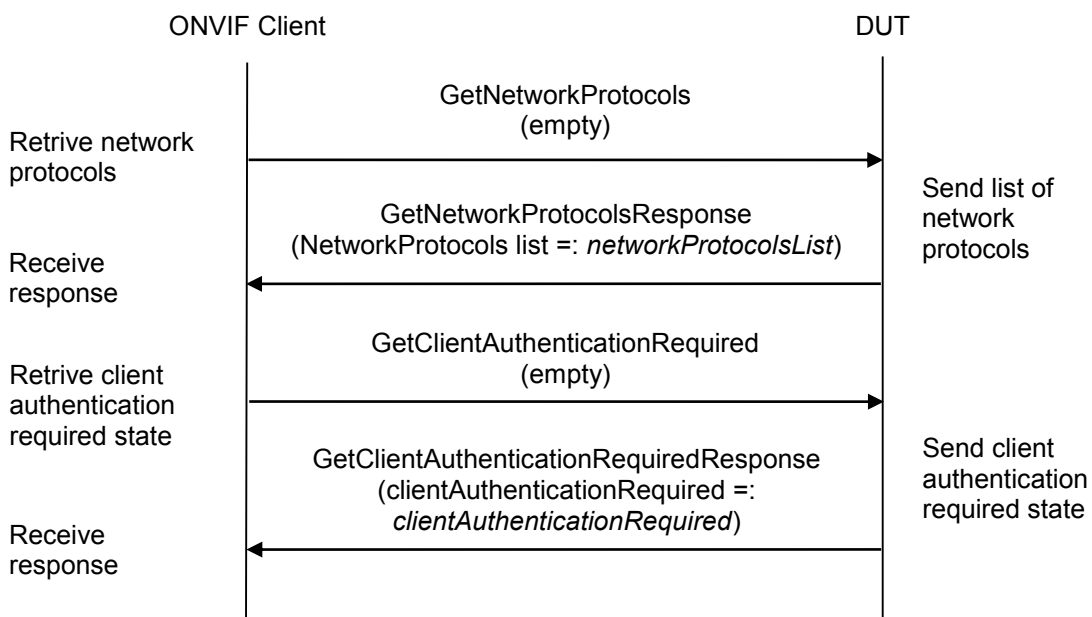
**WSDL Reference:** advancedsecurity.wsdl

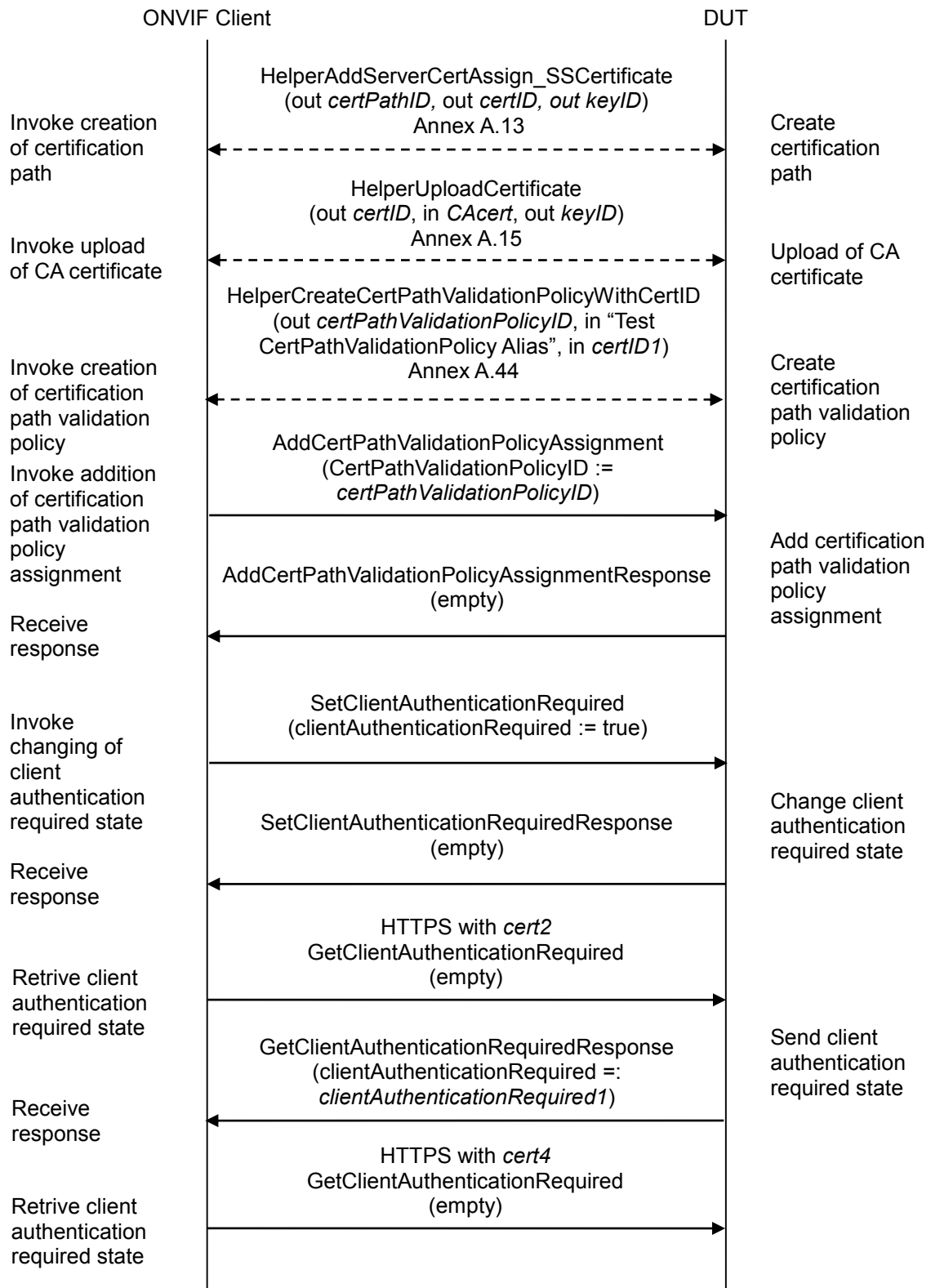
**Test Purpose:** To test the assignment of a self-signed certificate to a TLS server.

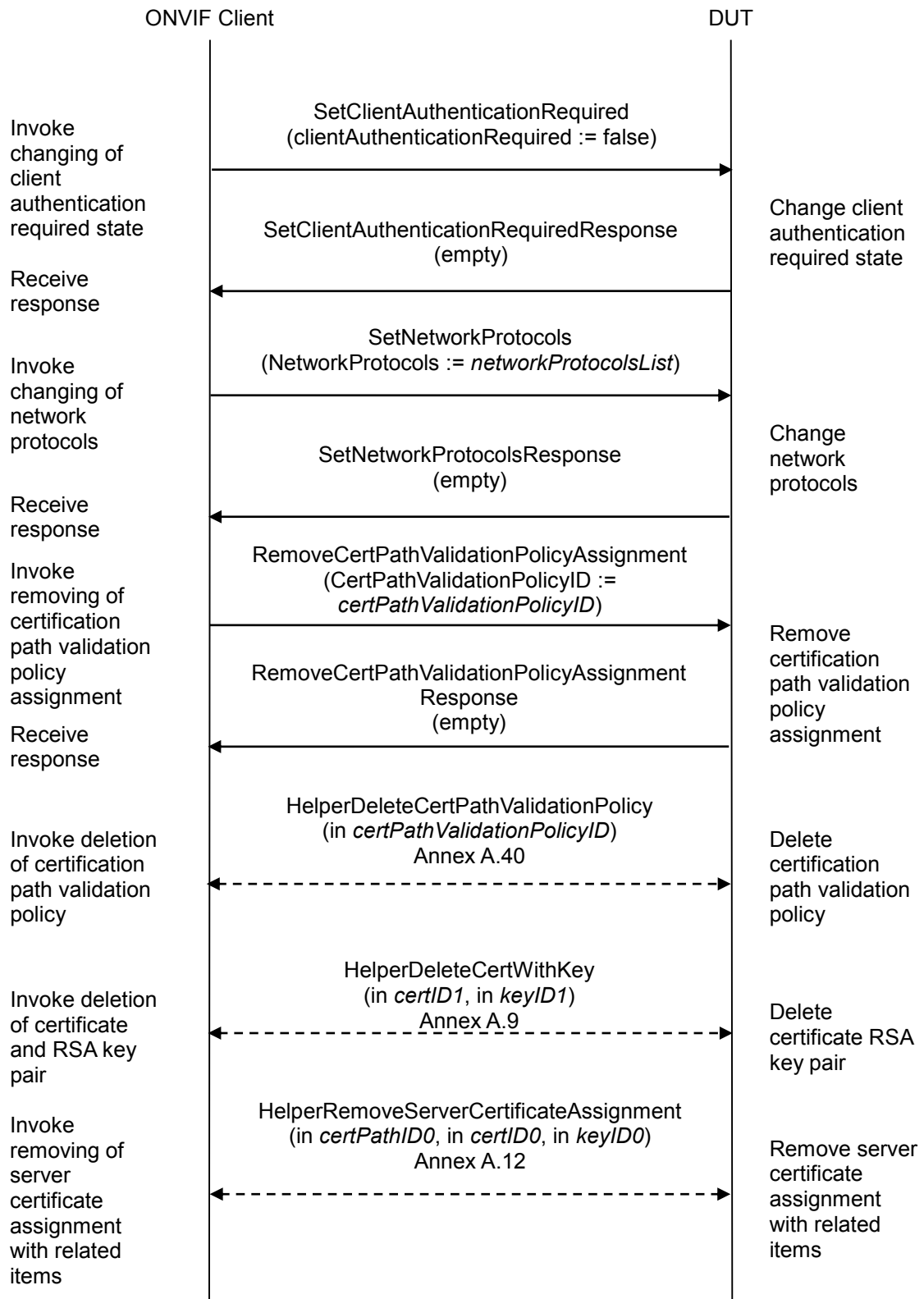
**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. Create PCKS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. TLS client authentication is supported by the DUT as indicated by the TLSClientAuthSupported capability. The DUT shall have enough free storage capacity for two additional RSA key pairs. The DUT shall have enough free storage capacity for two additional certificates. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional server certificate assignment. The DUT shall have enough free storage capacity for one additional certification path validation policy. The DUT shall have enough free storage capacity for one additional certification path validation policy assignment.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**









**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetNetworkProtocols** to retrieve configured network protocols of the DUT.
4. The DUT responds with a **GetNetworkProtocolsResponse** message with parameters
  - NetworkProtocols list =: *networkProtocolsList*
5. If *networkProtocolsList* does not contain network protocol with NetworkProtocols.Name is equal to "HTTPS", FAIL the test and skip other steps.
6. Set
  - *portHTTPS* := *networkProtocolsList*.NetworkProtocols[Name = "HTTPS"].Port
7. ONVIF Client invokes **GetClientAuthenticationRequired**.
8. The DUT responds with a **GetClientAuthenticationRequiredResponse** message with parameters
  - *clientAuthenticationRequired* =: *clientAuthenticationRequired*
9. ONVIF Client adds server certification assignment and creates related certification path (out *certPathID0*), the self-signed certificate (out *certID0*) and the RSA key pair (out *keyID0*) by following the procedure mentioned in Annex A.13.
10. ONVIF Client creates a CA certificate (out *CAcert1*) with subject (in "CN=ONVIF1 TT,C=US") and a corresponding public key (out *publicKey1*) in the certificate along with the corresponding private key (out *privateKey1*) by following the procedure described in Annex A.4.
11. ONVIF Client uploads a CA certificate (out *certID1*, in *CAcert1*) and new RSA key pair with the public key from the CA certificate (out *keyID1*) by following the procedure described in Annex A.15.
12. ONVIF Client creates a certificate (out *cert2*) signed by private key (in *privateKey1*) of the CA-certificate (in *CAcert1*) with subject (in "CN=ONVIF2 TT,C=US") and a corresponding public key (out *publicKey2*) in the certificate along with the corresponding private key (out *privateKey2*) by following the procedure described in Annex A.43.
13. ONVIF Client creates a CA certificate (out *CAcert3*) with subject (in "CN=ONVIF3 TT,C=US") and a corresponding public key (out *publicKey3*) in the certificate along with the corresponding private key (out *privateKey3*) by following the procedure described in Annex A.4.
14. ONVIF Client creates a certificate (out *cert4*) signed by private key (in *privateKey3*) of the CA-certificate (in *CAcert3*) with subject (in "CN=ONVIF4 TT,C=US") and a corresponding public key (out *publicKey4*) in the certificate along with the corresponding private key (out *privateKey4*) by following the procedure described in Annex A.43.
15. ONVIF Client creates certification path validation policy (out *certPathValidationPolicyID*) with specified alias (in "Test CertPathValidationPolicy Alias") and the certificate identifier (in *certID1*) for trust anchor by following the procedure mentioned in Annex A.44.



16. ONVIF Client invokes **AddCertPathValidationPolicyAssignment** with parameters
  - CertPathValidationPolicyID := *certPathValidationPolicyID*
17. The DUT responds with an **AddCertPathValidationPolicyAssignmentResponse** message.
18. If HTTPS protocol with NetworkProtocols.Name is equal to "HTTPS" from *networkProtocolsList* has NetworkProtocols.Enabled equal to true, go to the step 23.
19. ONVIF Client invokes **SetNetworkProtocols** message with parameters
  - NetworkProtocols[0].Name := HTTPS
  - NetworkProtocols[0].Enabled := true
  - NetworkProtocols[0].Port := 443
  - NetworkProtocols[0].Extension skipped
20. The DUT responds with a **SetNetworkProtocolsResponse** message.
21. ONVIF Client waits for time *timeout1*.
22. Set
  - *portHTTPS* := 443
23. If *clientAuthenticationRequired* is equal to false:
  - 23.1. ONVIF Client invokes **SetClientAuthenticationRequired** with parameters
    - clientAuthenticationRequired := true
  - 23.2. The DUT responds with a **SetClientAuthenticationRequiredResponse** message.
24. ONVIF Client invokes **GetClientAuthenticationRequired** through HTTPS using the client certificate *cert2* and port *portHTTPS*.
25. The DUT responds with a **GetClientAuthenticationRequiredResponse** message with parameters
  - clientAuthenticationRequired =: *clientAuthenticationRequired1*
26. ONVIF Client invokes **GetClientAuthenticationRequired** through HTTPS using the client certificate *cert4* and port *portHTTPS*.
27. The DUT does not establish a TLS connection.
28. If *clientAuthenticationRequired* is equal to false:
  - 28.1. ONVIF Client invokes **SetClientAuthenticationRequired** with parameters
    - clientAuthenticationRequired := false
  - 28.2. The DUT responds with a **SetClientAuthenticationRequiredResponse** message.
29. ONVIF Client invokes **SetNetworkProtocols** message with parameters





- NetworkProtocols := *networkProtocolsList*
30. The DUT responds with a **SetNetworkProtocolsResponse** message.
  31. ONVIF Client waits for time *timeout1*.
  32. ONVIF Client invokes **RemoveCertPathValidationPolicyAssignment** with parameters
    - CertPathValidationPolicyID := *certPathValidationPolicyID*
  33. The DUT responds with a **RemoveCertPathValidationPolicyAssignmentResponse** message.
  34. ONVIF Client deletes the certification path validation policy (in *certPathValidationPolicyID*) by following the procedure mentioned in Annex A.40 to restore DUT configuration.
  35. ONVIF Client deletes the CA certificate (in *certID1*) and related RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.9 to restore DUT configuration.
  36. ONVIF Client removes server certification assignment and deletes related certification path (in *certPathID0*), self-signed certificate (in *certID0*) and RSA key pair (in *keyID0*) by following the procedure mentioned in Annex A.12 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **RemoveCertPathValidationPolicyAssignmentResponse** message.

The DUT did not send **SetNetworkProtocolsResponse** message.

The DUT did not send **SetClientAuthenticationRequiredResponse** message.

The DUT establishes a TLS connection for step 27.

The DUT did not send **GetClientAuthenticationRequiredResponse** message.

The DUT did not send **AddCertPathValidationPolicyAssignmentResponse** message.

The DUT did not send **GetNetworkProtocolsResponse** message.

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.



**4.3.3.2 CRL processing with on-device RSA key pair**

**Test Label:** Verify CRL Processing with On-Device RSA Key Pair

**Test Case ID:** ADVANCED\_SECURITY-3-3-2

**ONVIF Core Specification Coverage:** Advanced Security, TLS Server

**Command Under Test:** GetClientAuthentication, SetClientAuthenticationRequired

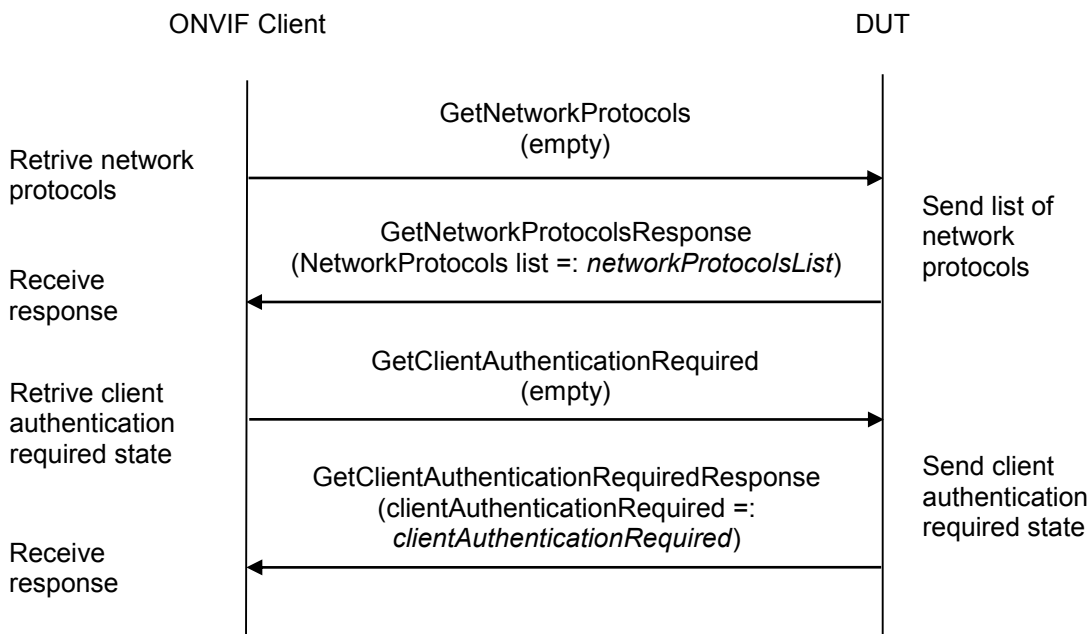
**WSDL Reference:** advancedsecurity.wsdl

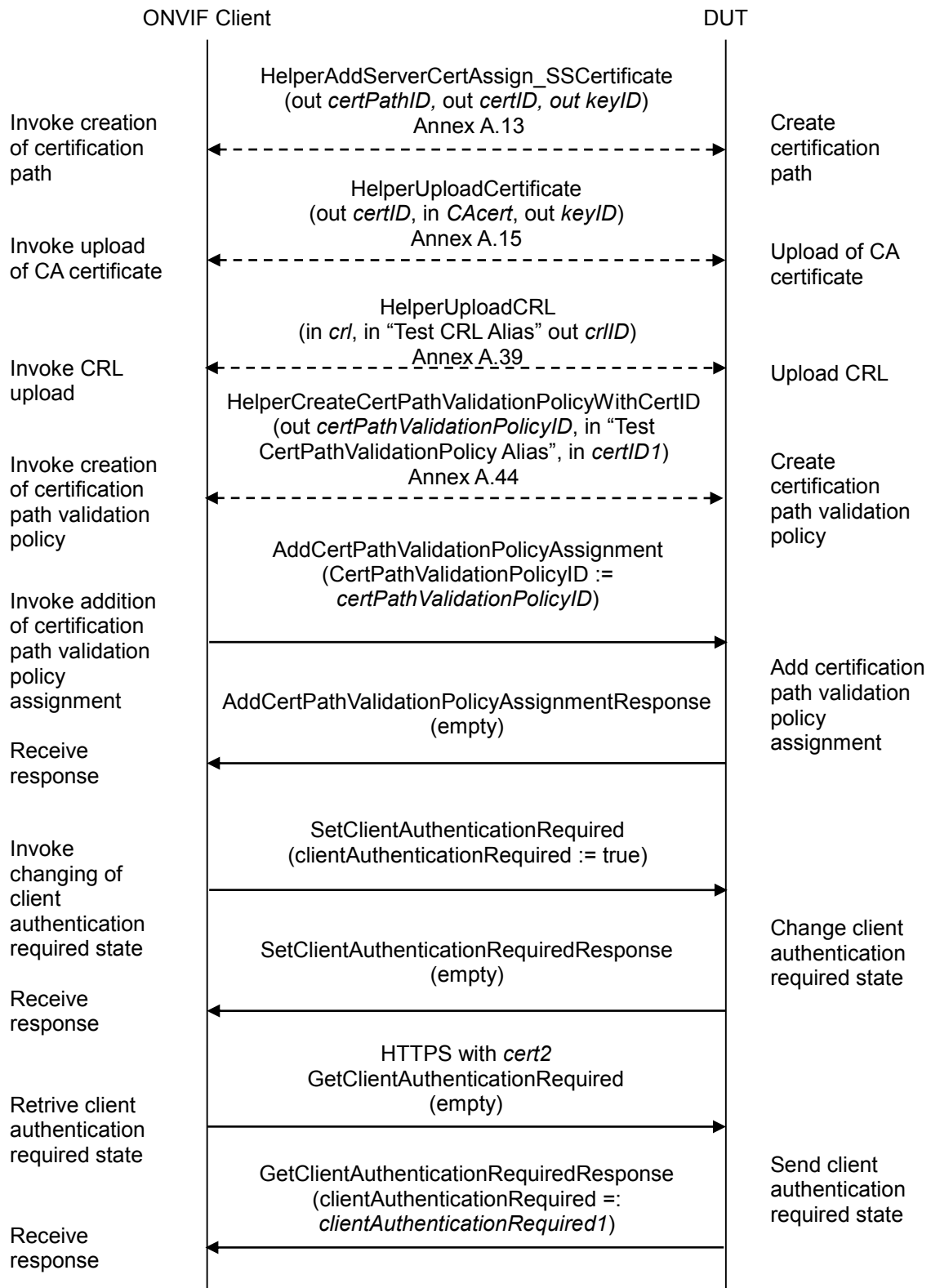
**Test Purpose:** To test the assignment of a self-signed certificate to a TLS server.

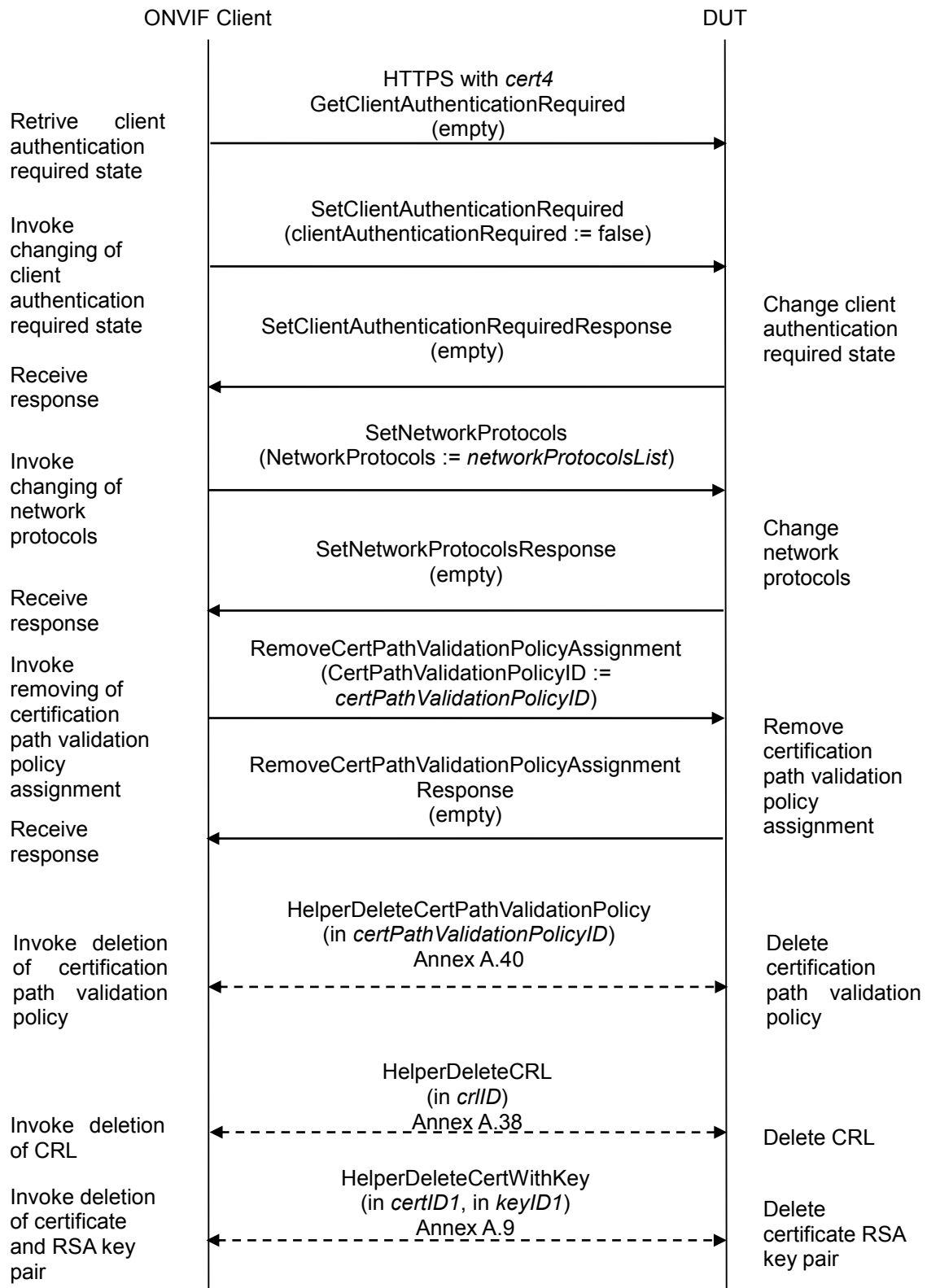
**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. Create PCKS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. CRLs supported by the DUT as indicated by the MaximumNumberOfCRLs capability. TLS client authentication is supported by the DUT as indicated by the TLSClientAuthSupported capability. The DUT shall have enough free storage capacity for two additional RSA key pairs. The DUT shall have enough free storage capacity for two additional certificates. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional server certificate assignment. The DUT shall have enough free storage capacity for one additional certification path validation policy. The DUT shall have enough free storage capacity for one additional CRL. The DUT shall have enough free storage capacity for one additional certification path validation policy assignment.

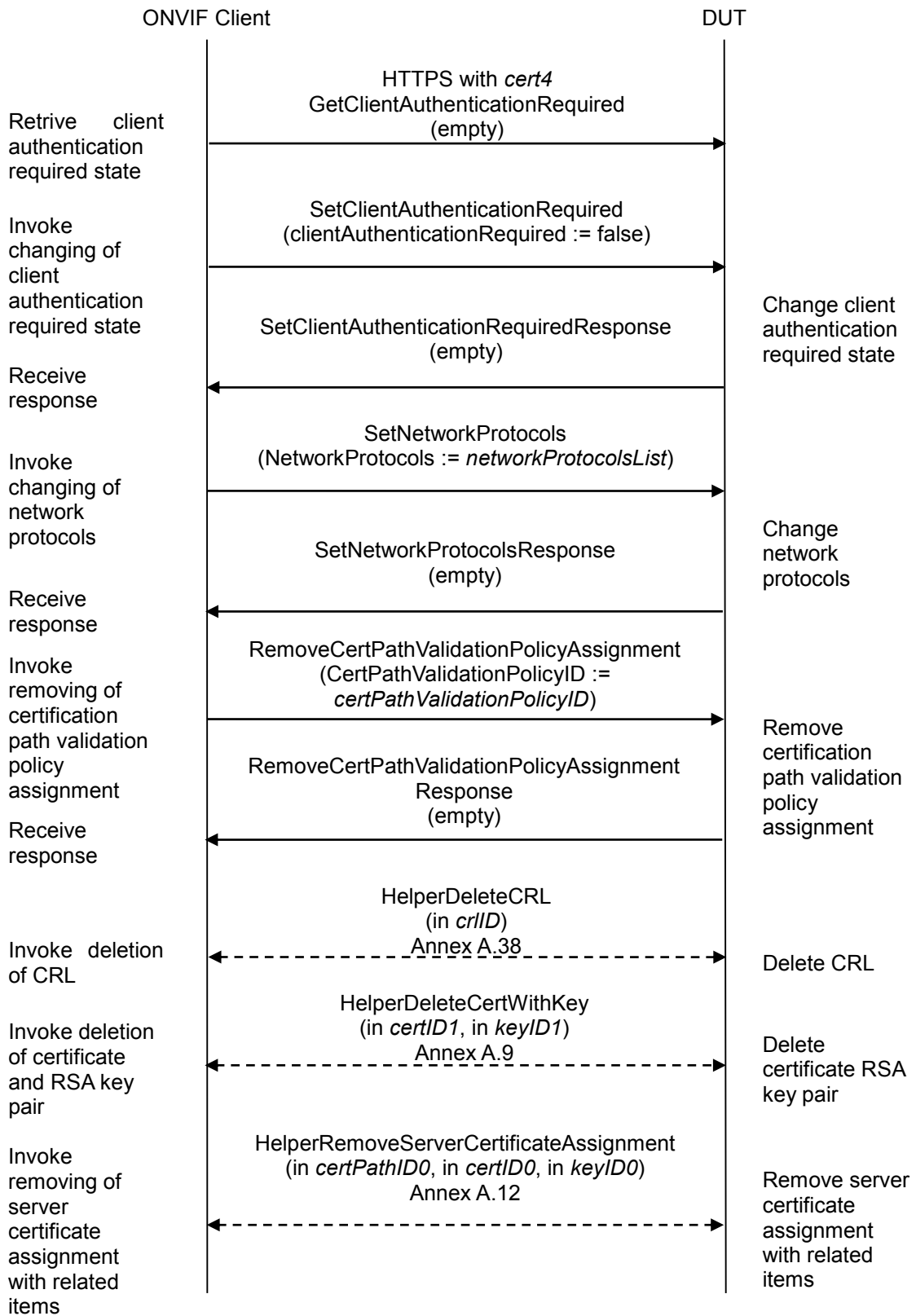
**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**











### Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetNetworkProtocols** to retrieve configured network protocols of the DUT.
4. The DUT responds with a **GetNetworkProtocolsResponse** message with parameters
  - NetworkProtocols list =: *networkProtocolsList*
5. If *networkProtocolsList* does not contain network protocol with NetworkProtocols.Name is equal to "HTTPS", FAIL the test and skip other steps.
6. Set
  - *portHTTPS* := *networkProtocolsList*.NetworkProtocols[Name = "HTTPS"].Port
7. ONVIF Client invokes **GetClientAuthenticationRequired**.
8. The DUT responds with a **GetClientAuthenticationRequiredResponse** message with parameters
  - *clientAuthenticationRequired* =: *clientAuthenticationRequired*
9. ONVIF Client adds server certification assignment and creates related certification path (out *certPathID0*), the self-signed certificate (out *certID0*) and the RSA key pair (out *keyID0*) by following the procedure mentioned in Annex A.13.
10. ONVIF Client creates a CA certificate (out *CAcert1*) with subject (in "CN=ONVIF1 TT,C=US") and a corresponding public key (out *publicKey1*) in the certificate along with the corresponding private key (out *privateKey1*) by following the procedure described in Annex A.4.
11. ONVIF Client uploads a CA certificate (out *certID1*, in *CAcert1*) and new RSA key pair with the public key from the CA certificate (out *keyID1*) by following the procedure described in Annex A.15.
12. ONVIF Client creates a certificate (out *cert2*) signed by private key (in *privateKey1*) of the CA-certificate (in *CAcert1*) with subject (in "CN=ONVIF2 TT,C=US") and a corresponding public key (out *publicKey2*) in the certificate along with the corresponding private key (out *privateKey2*) by following the procedure described in Annex A.43.
13. ONVIF Client creates a certificate (out *cert3*) signed by private key (in *privateKey1*) of the CA-certificate (in *CAcert1*) with subject (in "CN=ONVIF3 TT,C=US") and a corresponding public key (out *publicKey3*) in the certificate along with the corresponding private key (out *privateKey3*) by following the procedure described in Annex A.43.
14. ONVIF Client creates a CRL (out *crI*) for certificate revocation (in *cert3*) signed by private key (in *privateKey1*) by following the procedure mentioned in Annex A.45.
15. ONVIF Client uploads a CRL (in *crI*) with alias (in "ONVIF\_CRL\_Test") identifier (out *crIID*) by following the procedure described in Annex A.39.
16. ONVIF Client creates certification path validation policy (out *certPathValidationPolicyID*) with specified alias (in "Test CertPathValidationPolicy Alias") and the certificate identifier (in *certID1*)



for trust anchor by following the procedure mentioned in Annex A.44.

17. ONVIF Client invokes **AddCertPathValidationPolicyAssignment** with parameters
  - CertPathValidationPolicyID := *certPathValidationPolicyID*
18. The DUT responds with an **AddCertPathValidationPolicyAssignmentResponse** message.
19. If HTTPS protocol with NetworkProtocols.Name is equal to "HTTPS" from *networkProtocolsList* has NetworkProtocols.Enabled equal to true, go to the step 24.
20. ONVIF Client invokes **SetNetworkProtocols** message with parameters
  - NetworkProtocols[0].Name := HTTPS
  - NetworkProtocols[0].Enabled := true
  - NetworkProtocols[0].Port := 443
  - NetworkProtocols[0].Extension skipped
21. The DUT responds with a **SetNetworkProtocolsResponse** message.
22. ONVIF Client waits for time *timeout1*.
23. Set
  - *portHTTPS* := 443
24. If *clientAuthenticationRequired* is equal to false:
  - 24.1. ONVIF Client invokes **SetClientAuthenticationRequired** with parameters
    - clientAuthenticationRequired := true
  - 24.2. The DUT responds with a **SetClientAuthenticationRequiredResponse** message.
25. ONVIF Client invokes **GetClientAuthenticationRequired** through HTTPS using the client certificate *cert2* and port *portHTTPS*.
26. The DUT responds with a **GetClientAuthenticationRequiredResponse** message with parameters
  - clientAuthenticationRequired =: *clientAuthenticationRequired1*
27. ONVIF Client invokes **GetClientAuthenticationRequired** through HTTPS using the client certificate *cert3* and port *portHTTPS*.
28. The DUT does not establish a TLS connection.
29. If *clientAuthenticationRequired* is equal to false:
  - 29.1. ONVIF Client invokes **SetClientAuthenticationRequired** with parameters
    - clientAuthenticationRequired := false
  - 29.2. The DUT responds with a **SetClientAuthenticationRequiredResponse** message.



30. ONVIF Client invokes **SetNetworkProtocols** message with parameters
  - NetworkProtocols := *networkProtocolsList*
31. The DUT responds with a **SetNetworkProtocolsResponse** message.
32. ONVIF Client waits for time *timeout1*.
33. ONVIF Client invokes **RemoveCertPathValidationPolicyAssignment** with parameters
  - CertPathValidationPolicyID := *certPathValidationPolicyID*
34. The DUT responds with a **RemoveCertPathValidationPolicyAssignmentResponse** message.
35. ONVIF Client deletes the certification path validation policy (in *certPathValidationPolicyID*) by following the procedure mentioned in Annex A.40 to restore DUT configuration.
36. ONVIF Client deletes the CRL (in *crlID*) by following the procedure mentioned in Annex A.38 to restore DUT configuration.
37. ONVIF Client deletes the CA certificate (*certID1*) and related RSA key pair (*keyID1*) by following the procedure mentioned in Annex A.9 to restore DUT configuration.
38. ONVIF Client removes server certification assignment and deletes related certification path (in *certPathID0*), self-signed certificate (in *certID0*) and RSA key pair (in *keyID0*) by following the procedure mentioned in Annex A.12 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **RemoveCertPathValidationPolicyAssignmentResponse** message.

The DUT did not send **SetNetworkProtocolsResponse** message.

The DUT did not send **SetClientAuthenticationRequiredResponse** message.

The DUT establishes a TLS connection for step 28.

The DUT did not send **GetClientAuthenticationRequiredResponse** message.

The DUT did not send **AddCertPathValidationPolicyAssignmentResponse** message.

The DUT did not send **GetNetworkProtocolsResponse** message.

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.





### 4.3.3.3 Replace certification path validation policy assignment

**Test Label:** Replace Certification Path Validation Policy Assignment

**Test Case ID:** ADVANCED\_SECURITY-3-3-3

**ONVIF Core Specification Coverage:** Advanced Security, Replace Certification Path Validation Policy

**Command Under Test:** ReplaceCertPathValidationPolicyAssignment, GetAssignedCertPathValidationPolicies

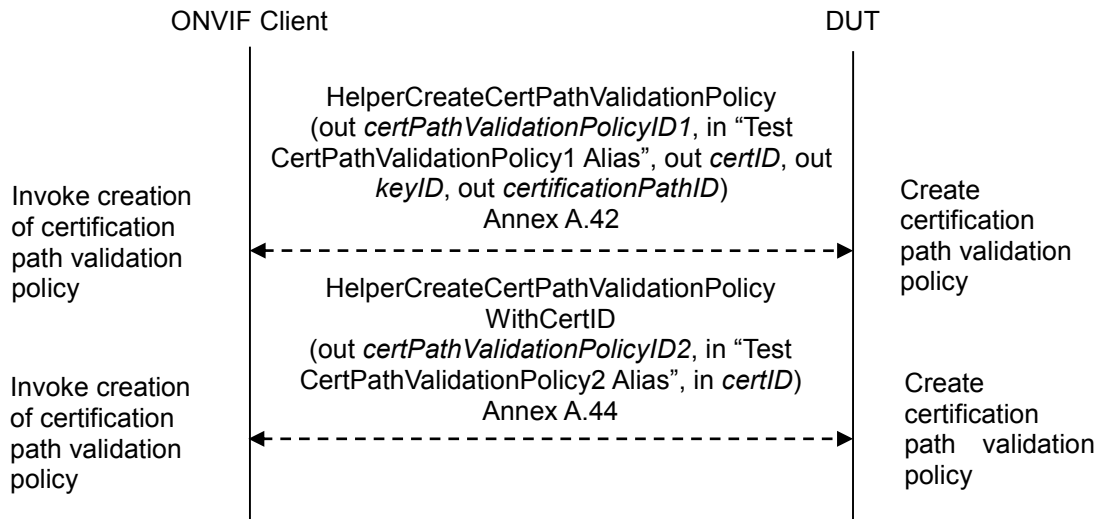
**WSDL Reference:** advancedsecurity.wsdl

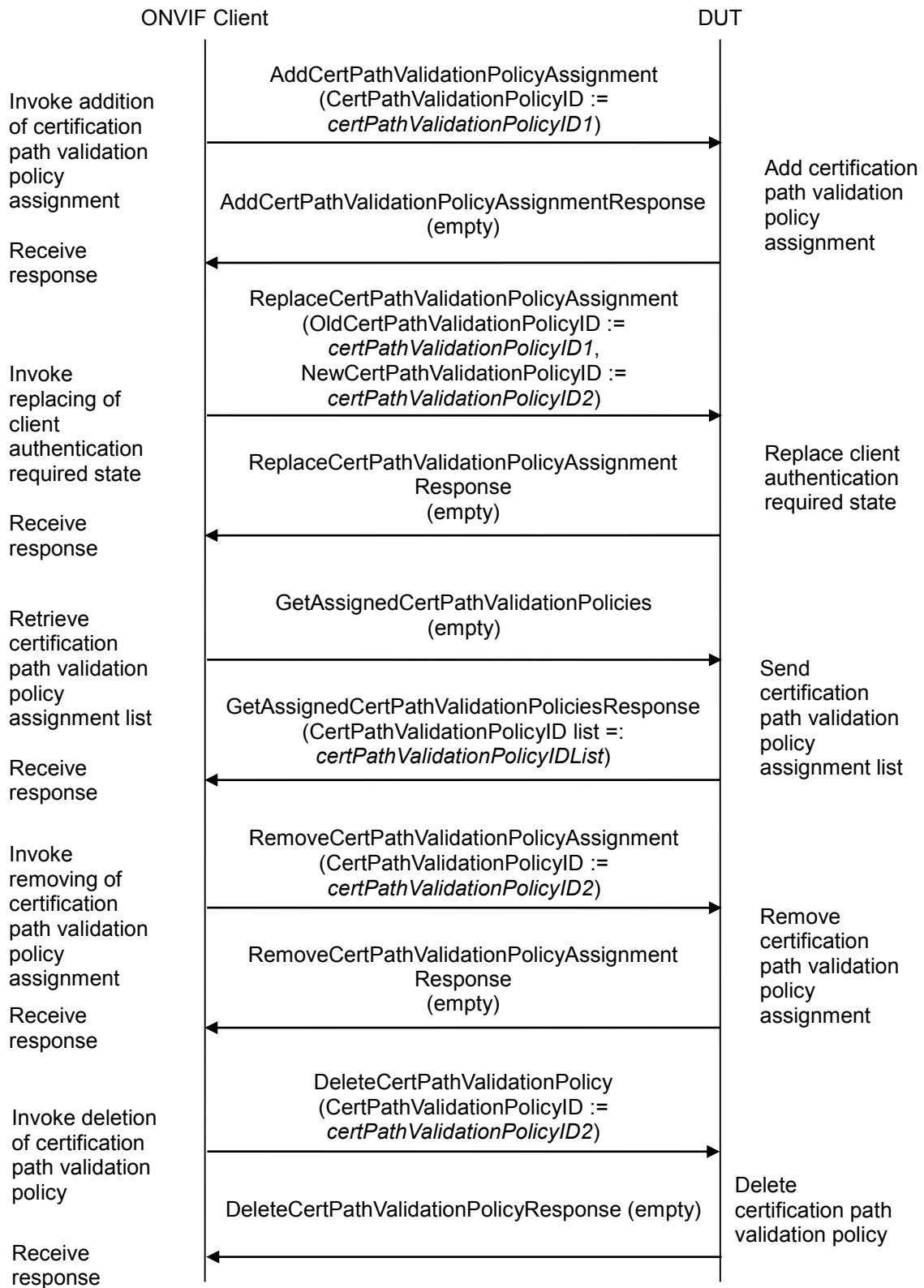
**Test Purpose:** Verify that DUT correctly supports replacing certification path validation policy assignments.

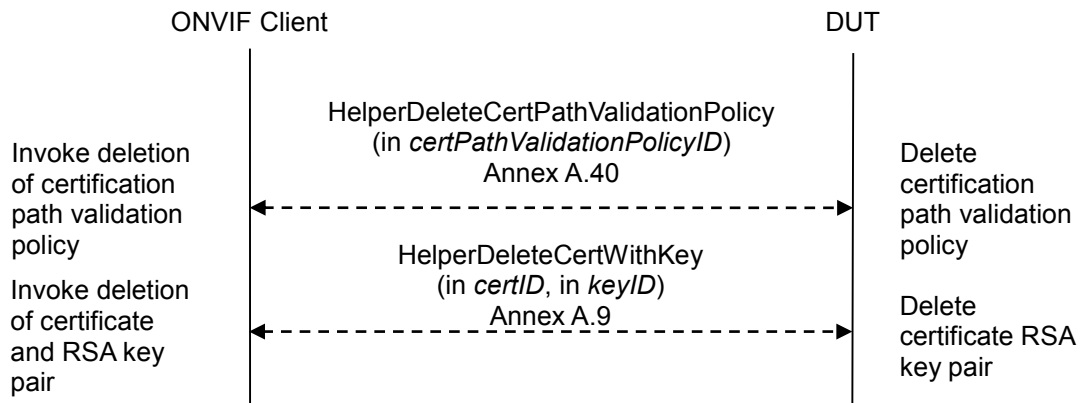
**Pre-requisite:** Advanced Security Service is received from the DUT. Certification path validation policy supported by the DUT as indicated by the MaximumNumberOfCertificationPathValidationPolicies capability. TLS client authentication is supported by the DUT as indicated by the TLSCientAuthSupported capability. The DUT shall have enough free storage capacity for two additional certification path validation policies. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. The DUT shall have enough free storage capacity for one additional certification path validation policy assignment.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**







**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates certification path validation policy identifier (out *certPathValidationPolicyID1*) with specified alias (in “Test CertPathValidationPolicy1 Alias”), related certificate (out *certID*), RSA key pair (out *keyID*) and certification path if any (out *certificationPathID*) by following the procedure mentioned in Annex A.42.
4. ONVIF Client creates certification path validation policy identifier (out *certPathValidationPolicyID2*) with specified alias (in “Test CertPathValidationPolicy2 Alias”) and the certificate identifier (in *certID*) for trust anchor by following the procedure mentioned in Annex A.44.
5. ONVIF Client invokes **AddCertPathValidationPolicyAssignment** with parameters
  - *CertPathValidationPolicyID* := *certPathValidationPolicyID1*
6. The DUT responds with an **AddCertPathValidationPolicyAssignmentResponse** message.
7. ONVIF Client invokes **ReplaceCertPathValidationPolicyAssignment** with parameters
  - *OldCertPathValidationPolicyID* := *certPathValidationPolicyID1*
  - *NewCertPathValidationPolicyID* := *certPathValidationPolicyID2*
8. The DUT responds with a **ReplaceCertPathValidationPolicyAssignmentResponse** message.
9. ONVIF Client invokes **GetAssignedCertPathValidationPolicies**.
10. The DUT responds with a **GetAssignedCertPathValidationPoliciesResponse** message with parameters
  - *CertPathValidationPolicyID list* =: *certPathValidationPolicyIDList*
11. If *certPathValidationPolicyIDList* contains *CertPathValidationPolicyID* equal to *certPathValidationPolicyID1*, FAIL the test and go to the step 14.



12. If *certPathValidationPolicyIDList* does not contain *CertPathValidationPolicyID* equal to *certPathValidationPolicyID2*, FAIL the test and go to the step 19.
13. Go to the step 17.
14. ONVIF Client invokes **RemoveCertPathValidationPolicyAssignment** with parameters
  - *CertPathValidationPolicyID* := *certPathValidationPolicyID1*
15. The DUT responds with a **RemoveCertPathValidationPolicyAssignmentResponse** message.
16. Go to the step 19.
17. ONVIF Client invokes **RemoveCertPathValidationPolicyAssignment** with parameters
  - *CertPathValidationPolicyID* := *certPathValidationPolicyID2*
18. The DUT responds with a **RemoveCertPathValidationPolicyAssignmentResponse** message.
19. ONVIF Client invokes **DeleteCertPathValidationPolicy** with parameters
  - *CertPathValidationPolicyID* := *certPathValidationPolicyID2*
20. DUT responds with a **DeleteCertPathValidationPolicyResponse** message.
21. ONVIF Client deletes the certification path validation policy (in *certPathValidationPolicyID1*) by following the procedure mentioned in Annex A.40 to restore DUT configuration.
22. If *certificationPathID* is null:
  - 22.1. ONVIF Client deletes the self-signed certificate (in *certID*) and related the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.9.
  - 22.2. Skip other steps.
23. If *certificationPathID* is not null:
  - 23.1. ONVIF Client deletes the certification path (in *certificationPathID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.35 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **AddCertPathValidationPolicyAssignmentResponse** message.

The DUT did not send **ReplaceCertPathValidationPolicyAssignmentResponse** message.

The DUT did not send **GetAssignedCertPathValidationPoliciesResponse** message.

The DUT did not send **RemoveCertPathValidationPolicyAssignmentResponse** message.

The DUT did not send **DeleteCertPathValidationPolicyResponse** message.



## **4.4 Referential integrity**

### **4.4.1 TLS Server Certificate - self-signed**

**Test Label:** Referential Integrity - TLS Server Certificate - Self-Signed

**Test Case ID:** ADVANCED\_SECURITY-4-1-1

**ONVIF Core Specification Coverage:** Advanced Security, Keystore – Key Management, Keystore - Certificate Management, TLS Server

**Command Under Test:** DeleteKey, DeleteCertificate. DeleteCertificationPath

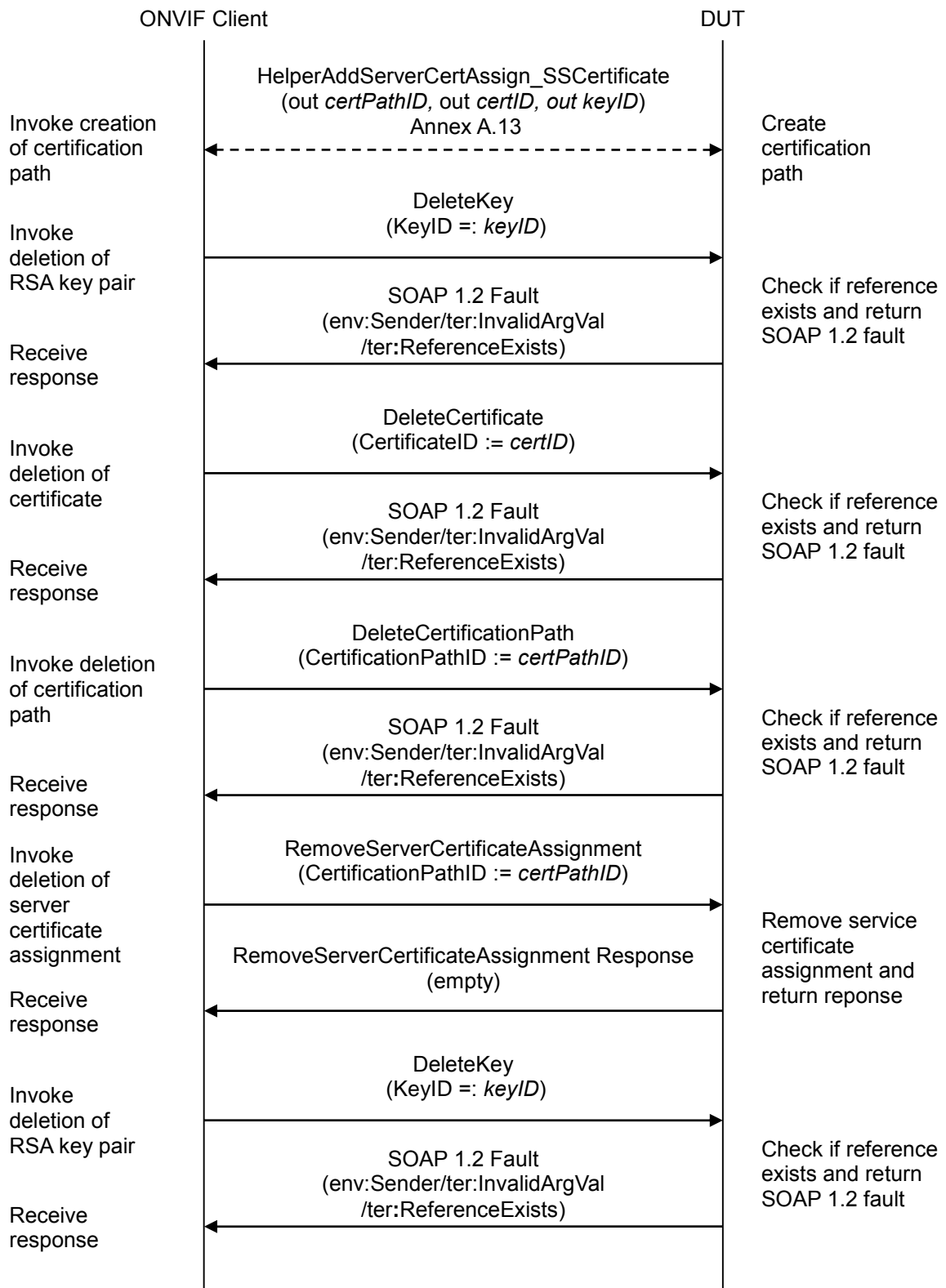
**WSDL Reference:** advancedsecurity.wsdl

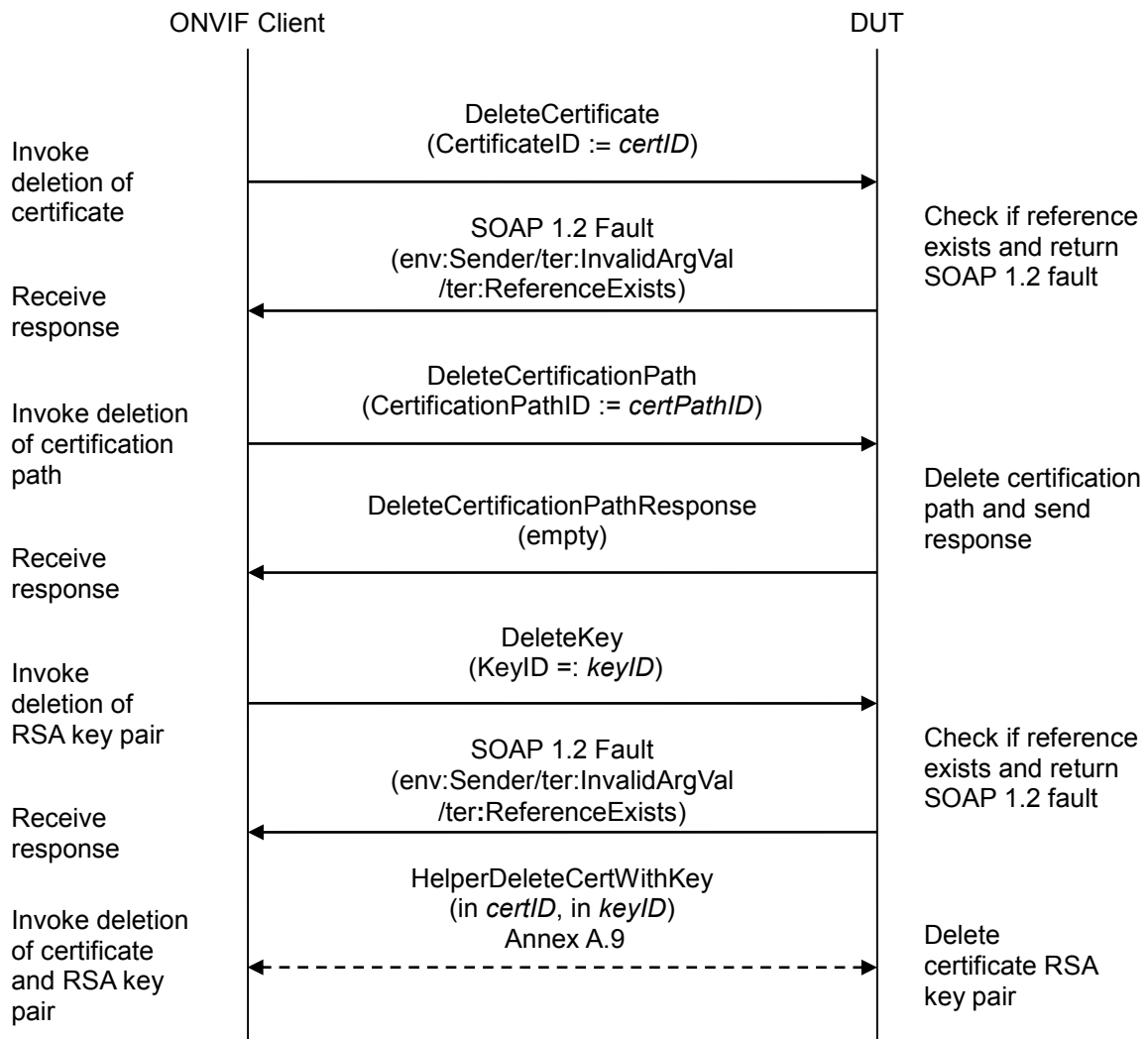
**Test Purpose:** To test the referential integrity of certificate assigned to a TLS server.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional server certificate assignment.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client adds server certification assignment and creates related certification path (out *certPathID*), the self-signed certificate (out *certID*) and the RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.13.
4. ONVIF Client invokes **DeleteKey** with parameters
  - KeyID =: *keyID*
5. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
6. ONVIF Client invokes **DeleteCertificate** with parameters



- CertificateID := certID
7. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
  8. ONVIF Client invokes **DeleteCertificationPath** with parameters
    - CertificationPathID := *certPathID*
  9. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
  10. ONVIF Client invokes **RemoveServerCertificateAssignment** with parameters
    - CertificationPathID := *certPathID*
  11. DUT responds with a **RemoveServerCertificateAssignmentResponse** message.
  12. ONVIF Client invokes **DeleteKey** with parameters
    - KeyID =: *keyID*
  13. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
  14. ONVIF Client invokes **DeleteCertificate** with parameters
    - CertificateID := *certID*
  15. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
  16. ONVIF Client invokes **DeleteCertificationPath** with parameters
    - CertificationPathID := *certPathID*
  17. The DUT responds with a **DeleteCertificationPathResponse** message.
  18. ONVIF Client invokes **DeleteKey** with parameters
    - KeyID =: *keyID*
  19. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
  20. ONVIF Client deletes the self-signed certificate (in *certID*) and related RSA key pair (in *keyID*) by following procedure mentioned in Annex A.9 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **DeleteCertificationPathResponse** message.

The DUT did not send **RemoveServerCertificateAssignmentResponse** message.

The DUT did not send the env:Receiver/ter:Action/ter:NoMatchingPrivateKey SOAP 1.2 fault message(s).





#### **4.4.2 TLS Server Certificate – CA**

**Test Label:** Referential Integrity - TLS Server Certificate – CA

**Test Case ID:** ADVANCED\_SECURITY-4-1-2

**ONVIF Core Specification Coverage:** Advanced Security, Keystore – Key Management, Keystore - Certificate Management, TLS Server

**Command Under Test:** DeleteKey, DeleteCertificate. DeleteCertificationPath

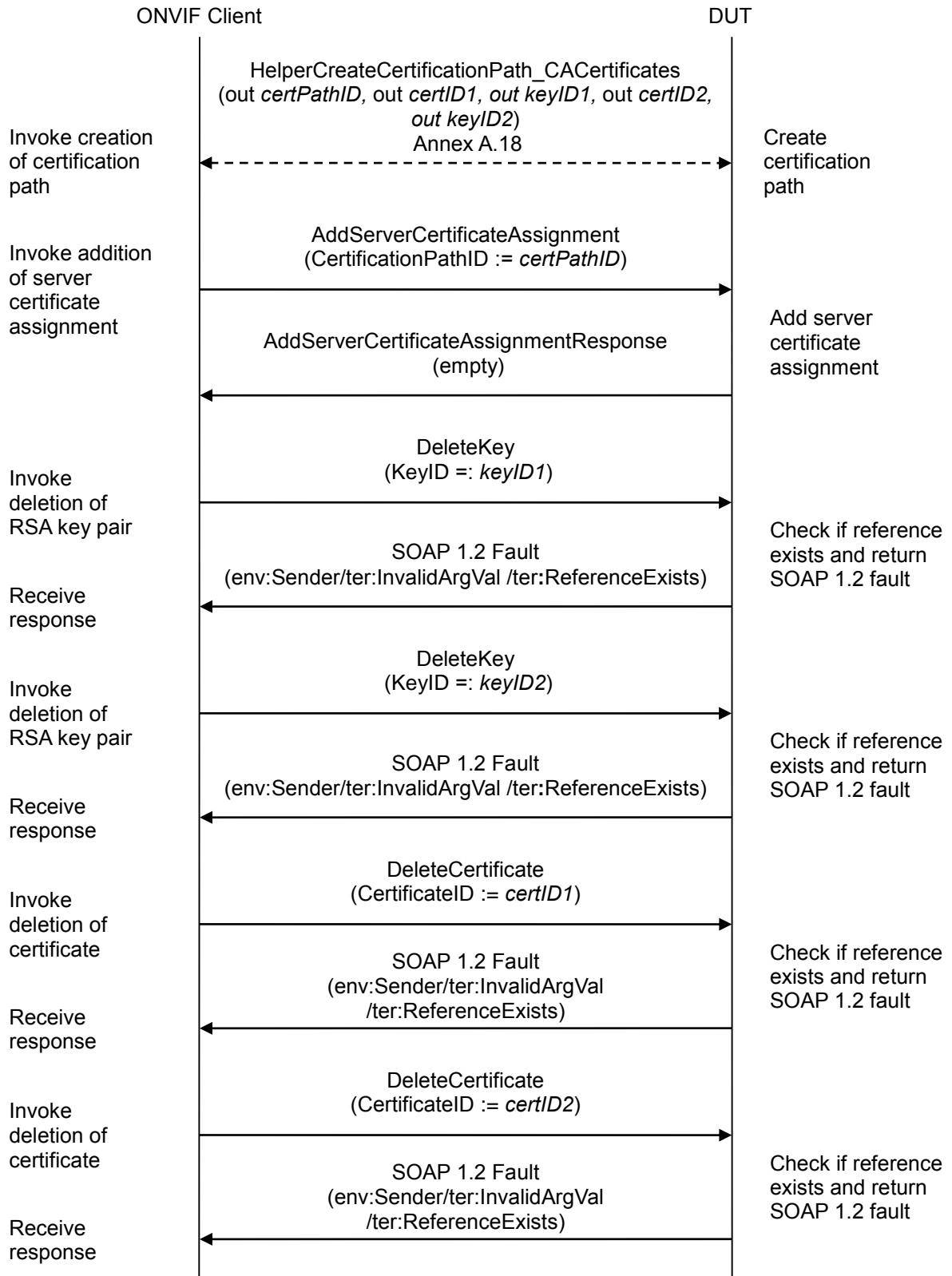
**WSDL Reference:** advancedsecurity.wsdl

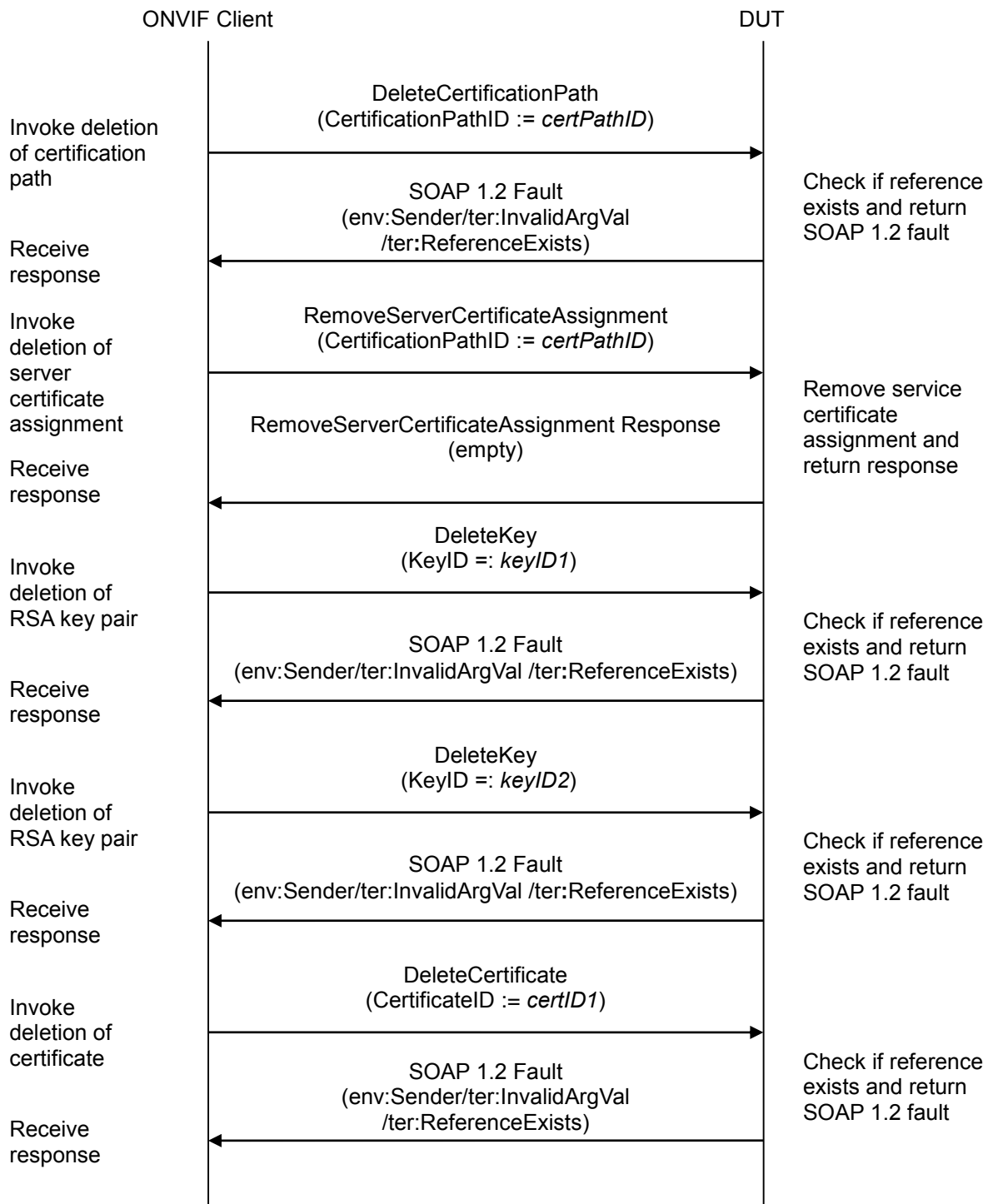
**Test Purpose:** To test the referential integrity of certificate assigned to a TLS server.

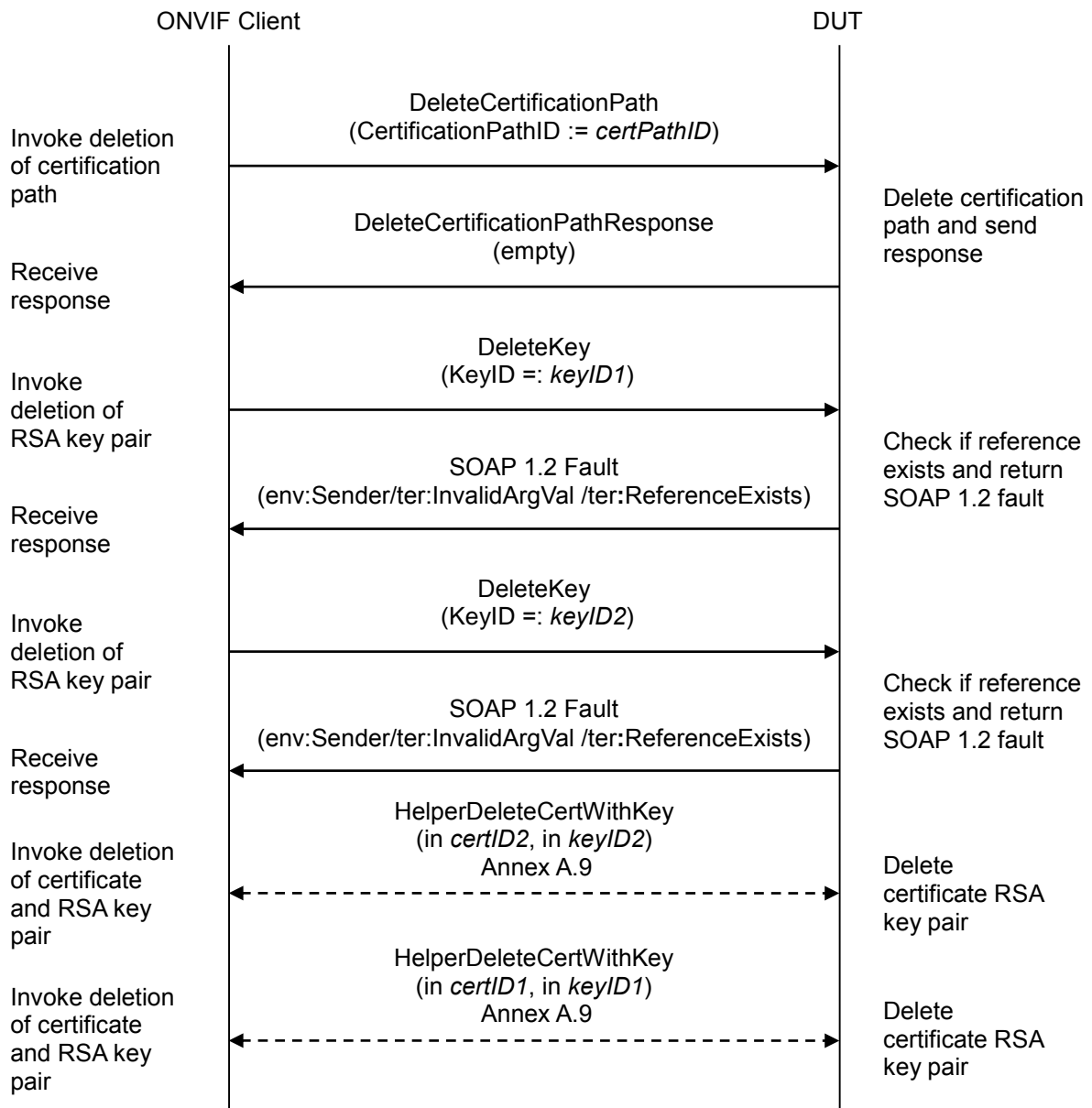
**Pre-requisite:** Advanced Security Service is received from the DUT. Create PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for two additional RSA key pairs. The DUT shall have enough free storage capacity for two additional certificates. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional server certificate assignment. Current time of the DUT shall be at least Jan 01, 1970.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**







**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a certification path (out *certPathID*) based on CA-signed certificate (out *certID1*) and related RSA key pair (out *keyID1*) and a corresponding CA certificate (out *certID2*) and related RSA key pair (out *keyID2*) by following the procedure mentioned in Annex A.18.
4. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
  - CertificationPathID := *certPathID*



5. The DUT responds with an **AddServerCertificateAssignmentResponse** message.
6. ONVIF Client invokes **DeleteKey** with parameters
  - KeyID =: *keyID1*
7. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
8. ONVIF Client invokes **DeleteKey** with parameters
  - KeyID =: *keyID2*
9. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
10. ONVIF Client invokes **DeleteCertificate** with parameters
  - CertificateID := *certID1*
11. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
12. ONVIF Client invokes **DeleteCertificate** with parameters
  - CertificateID := *certID2*
13. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
14. ONVIF Client invokes **DeleteCertificationPath** with parameters
  - CertificationPathID := *certPathID*
15. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
16. ONVIF Client invokes **RemoveServerCertificateAssignment** with parameters
  - CertificationPathID := *certPathID*
17. The DUT responds with a **RemoveServerCertificateAssignmentResponse** message.
18. ONVIF Client invokes **DeleteKey** with parameters
  - KeyID =: *keyID1*
19. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
20. ONVIF Client invokes **DeleteKey** with parameters
  - KeyID =: *keyID2*
21. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
22. ONVIF Client invokes **DeleteCertificate** with parameters
  - CertificateID := *certID1*
23. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
24. ONVIF Client invokes **DeleteCertificate** with parameters



- CertificateID := *certID2*
25. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
26. ONVIF Client invokes **DeleteCertificationPath** with parameters
- CertificationPathID := *certPathID*
27. The DUT responds with a **DeleteCertificationPathResponse** message.
28. ONVIF Client invokes **DeleteKey** with parameters
- KeyID =: *keyID1*
29. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
30. ONVIF Client invokes **DeleteKey** with parameters
- KeyID =: *keyID2*
31. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
32. ONVIF Client deletes the self-signed certificate (in *certID1*) and related RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.9 to restore DUT configuration.
33. ONVIF Client deletes the self-signed certificate (in *certID2*) and related RSA key pair (in *keyID2*) by following the procedure mentioned in Annex A.9 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **DeleteCertificationPathResponse** message.

The DUT did not send **RemoveServerCertificateAssignmentResponse** message.

The DUT did not send the env:Receiver/ter:Action/ter:NoMatchingPrivateKey SOAP 1.2 fault message(s).



## 4.5 Capabilities

### 4.5.1 Advanced Security Service Capabilities

**Test Label:** Advanced Security Capabilities Verification

**Test Case ID:** ADVANCED\_SECURITY-5-1-1

**ONVIF Core Specification Coverage:** Advanced Security, Capabilities

**Command Under Test:** GetServiceCapabilities (for Advanced Security Service)

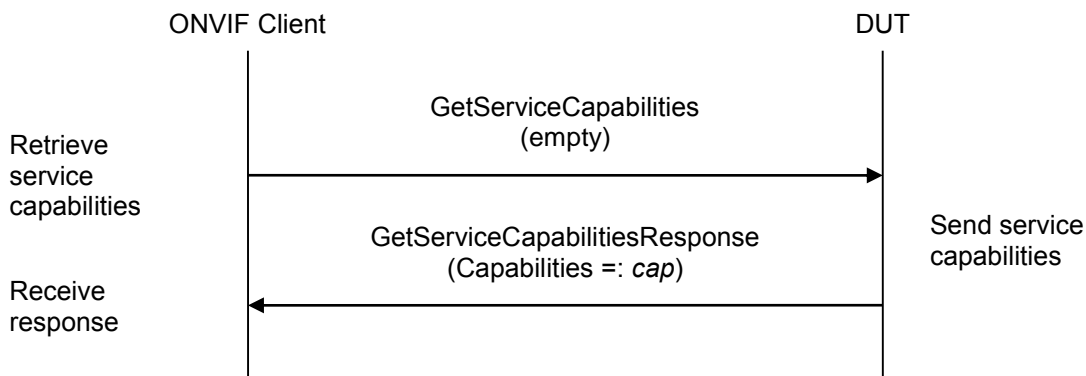
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To verify DUT Advanced Security Service Capabilities.

**Pre-requisite:** Advanced Security Service is received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServiceCapabilities**.
4. The DUT responds with a **GetServiceCapabilitiesResponse** message with parameters
  - Capabilities =: *cap*
5. If *cap.KeystoreCapabilities.MaximumNumberOfCertificates* > 0:
  - 5.1. If *cap.KeystoreCapabilities.MaximumNumberOfKeys* <= 0 or skipped, FAIL the test.
6. If *cap.KeystoreCapabilities.MaximumNumberOfCertificationPaths* > 0:
  - 6.1. If *cap.KeystoreCapabilities.MaximumNumberOfCertificates* < 2 or skipped, FAIL the test.



7. If `cap.KeystoreCapabilities.RSAKeyPairGeneration = true`:
  - 7.1. If `cap.KeystoreCapabilities.RSAKeyLenghts` is empty or skipped, FAIL the test.
  - 7.2. If `cap.KeystoreCapabilities.MaximumNumberOfKeys <= 0` or skipped, FAIL the test.
8. If `cap.KeystoreCapabilities.PKCS8RSAKeyPairUpload = true`:
  - 8.1. If `cap.KeystoreCapabilities.MaximumNumberOfPassphrases < 1` or skipped, FAIL the test.
  - 8.2. If `cap.KeystoreCapabilities.MaximumNumberOfKeys < 1` or skipped, FAIL the test.
  - 8.3. If `cap.KeystoreCapabilities.RSAKeyLenghts` is empty or skipped, FAIL the test.
  - 8.4. If `cap.KeystoreCapabilities.PasswordBasedEncryptionAlgorithms` is empty or skipped, FAIL the test.
  - 8.5. If `cap.KeystoreCapabilities.PasswordBasedEncryptionAlgorithms` does not contain "pbeWithSHAAnd3-KeyTripleDES-CBC" item, FAIL the test.
9. If `cap.KeystoreCapabilities.PKCS12CertificateWithRSAPrivateKeyUpload = true`:
  - 9.1. If `cap.KeystoreCapabilities.MaximumNumberOfPassphrases < 1` or skipped, FAIL the test.
  - 9.2. If `cap.KeystoreCapabilities.MaximumNumberOfKeys < 2` or skipped, FAIL the test.
  - 9.3. If `cap.KeystoreCapabilities.MaximumNumberOfCertificates < 2` or skipped, FAIL the test.
  - 9.4. If `cap.KeystoreCapabilities.MaximumNumberOfCertificationPaths <= 0` or skipped, FAIL the test.
  - 9.5. If `cap.KeystoreCapabilities.SignatureAlgorithms` list is empty, FAIL the test.
  - 9.6. If `cap.KeystoreCapabilities.RSAKeyLenghts` is empty or skipped, FAIL the test.
  - 9.7. If `cap.KeystoreCapabilities.PasswordBasedEncryptionAlgorithms` is empty or skipped, FAIL the test.
  - 9.8. If `cap.KeystoreCapabilities.PasswordBasedEncryptionAlgorithms` does not contain "pbeWithSHAAnd3-KeyTripleDES-CBC" item, FAIL the test.
  - 9.9. If `cap.KeystoreCapabilities.PasswordBasedMACAlgorithms` is empty or skipped, FAIL the test.
  - 9.10. If `cap.KeystoreCapabilities.PasswordBasedMACAlgorithms` does not contain "hmacWithSHA256" item, FAIL the test.
  - 9.11. If `cap.KeystoreCapabilities.X509Versions` is empty or skipped, FAIL the test.
  - 9.12. If `cap.KeystoreCapabilities.X509Versions` does not contain "3" item, FAIL the test.
  - 9.13. If `cap.KeystoreCapabilities.SignatureAlgorithms` list does not contain item with algorithm = "1.2.840.113549.1.1.5" (OID of SHA-1 with RSA Encryption algorithm), FAIL the test.
  - 9.14. If `cap.KeystoreCapabilities.SignatureAlgorithms` list does not contain item with algorithm = "1.2.840.113549.1.1.11" (OID of SHA-256 with RSA Encryption algorithm), FAIL the test.
10. If `cap.KeystoreCapabilities.PKCS10ExternalCertificationWithRSA = true`:





- 10.1. If (*cap.KeystoreCapabilities.RSAKeyPairGeneration* = false or skipped) and (*cap.KeystoreCapabilities.PKCS8RSAKeyPairUpload* = false or skipped) and (*cap.KeystoreCapabilities.PKCS12CertificateWithRSAPrivateKeyUpload* = false or skipped), FAIL the test.
- 10.2. If *cap.KeystoreCapabilities.SignatureAlgorithms* list is empty, FAIL the test.
- 10.3. If *cap.KeystoreCapabilities.MaximumNumberOfKeys* < 2 or skipped, FAIL the test.
- 10.4. If *cap.KeystoreCapabilities.MaximumNumberOfCertificates* < 2 or skipped, FAIL the test.
- 10.5. If *cap.KeystoreCapabilities.MaximumNumberOfCertificationPaths* <= 0 or skipped, FAIL the test.
- 10.6. If *cap.KeystoreCapabilities.SignatureAlgorithms* list does not contain item with algorithm = "1.2.840.113549.1.1.5" (OID of SHA-1 with RSA Encryption algorithm), FAIL the test.
- 10.7. If *cap.KeystoreCapabilities.SignatureAlgorithms* list does not contain item with algorithm = "1.2.840.113549.1.1.11" (OID of SHA-256 with RSA Encryption algorithm), FAIL the test.
11. If *cap.KeystoreCapabilities.SelfSignedCertificateCreationWithRSA* = true:
  - 11.1. If (*cap.KeystoreCapabilities.RSAKeyPairGeneration* = false or skipped) and (*cap.KeystoreCapabilities.PKCS8RSAKeyPairUpload* = false or skipped) and (*cap.KeystoreCapabilities.PKCS12CertificateWithRSAPrivateKeyUpload* = false or skipped), FAIL the test.
  - 11.2. If *cap.KeystoreCapabilities.MaximumNumberOfCertificates* <= 0 or skipped, FAIL the test.
  - 11.3. If *cap.KeystoreCapabilities.SignatureAlgorithms* list is empty, FAIL the test.
  - 11.4. If *cap.KeystoreCapabilities.SignatureAlgorithms* list does not contain item with algorithm = "1.2.840.113549.1.1.5" (OID of SHA-1 with RSA Encryption algorithm), FAIL the test.
  - 11.5. If *cap.KeystoreCapabilities.SignatureAlgorithms* list does not contain item with algorithm = "1.2.840.113549.1.1.11" (OID of SHA-256 with RSA Encryption algorithm), FAIL the test.
12. If *cap.KeystoreCapabilities.MaximumNumberOfCertificationPathValidationPolicies* > 0:
  - 12.1. If (*cap.KeystoreCapabilities.SelfSignedCertificateCreationWithRSA* = false or skipped) and (*cap.KeystoreCapabilities.PKCS10ExternalCertificationWithRSA* = false or skipped) and (*cap.KeystoreCapabilities.PKCS12CertificateWithRSAPrivateKeyUpload* = false or skipped), FAIL the test.
13. If *cap.TLSServerCapabilities.TLSServerSupported* is not empty:
  - 13.1. If *cap.TLSServerCapabilities.TLSServerSupported* does not contain the value 1.0, FAIL the test.
  - 13.2. If *cap.KeystoreCapabilities.MaximumNumberOfCertificationPaths* < 2 or skipped, FAIL the test.
  - 13.3. If *cap.TLSServerCapabilities.MaximumNumberOfTLSCertificationPaths* <= 0 or skipped, FAIL the test.
  - 13.4. If (*cap.KeystoreCapabilities.PKCS10ExternalCertificationWithRSA* = false or skipped) and (*cap.KeystoreCapabilities.SelfSignedCertificateCreationWithRSA* = false or skipped), FAIL the test.



14. If `cap.TLSServerCapabilities.TLSClientAuthSupported = true`:
  - 14.1. If `cap.TLSServerCapabilities.TLSServerSupported` is empty, FAIL the test.
  - 14.2. If `cap.KeystoreCapabilities.MaximumNumberOfCertificationPathValidationPolicies < 2` or skipped, FAIL the test.
  - 14.3. If `cap.TLSServerCapabilities.MaximumNumberOfTLSCertificationPathValidationPolicies <= 0` or skipped, FAIL the test.
  - 14.4. If `cap.KeystoreCapabilities.MaximumNumberOfCRLs <= 0` or skipped, FAIL the test.
15. If `cap.TLSServerCapabilities.MaximumNumberOfTLSCertificationPathValidationPolicies > 0`:
  - 15.1. If `cap.KeystoreCapabilities.MaximumNumberOfCertificationPathValidationPolicies <= 0` or skipped, FAIL the test.
16. If `cap.TLSServerCapabilities.TLSServerSupported` is not empty and `cap.KeystoreCapabilities.PKCS10ExternalCertificationWithRSA = true`:
  - 16.1. If `cap.KeystoreCapabilities.MaximumNumberOfCertificates < 3` or skipped, FAIL the test.
17. If `cap.TLSServerCapabilities.MaximumNumberOfTLSCertificationPaths > 0`:
  - 17.1. If `cap.KeystoreCapabilities.MaximumNumberOfCertificationPaths <= 0` or skipped, FAIL the test.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetServiceCapabilitiesResponse** message.



#### 4.5.2 Get Services and Get Advanced Security Service Capabilities Consistency

**Test Label:** Get Services and Advanced Security Service Capabilities Consistency Verification

**Test Case ID:** ADVANCED\_SECURITY-5-1-2

**ONVIF Core Specification Coverage:** Capability exchange (ONVIF Core Specification), Advanced Security, Capabilities

**Command Under Test:** GetServices, GetServiceCapabilities (for Advanced Security Service)

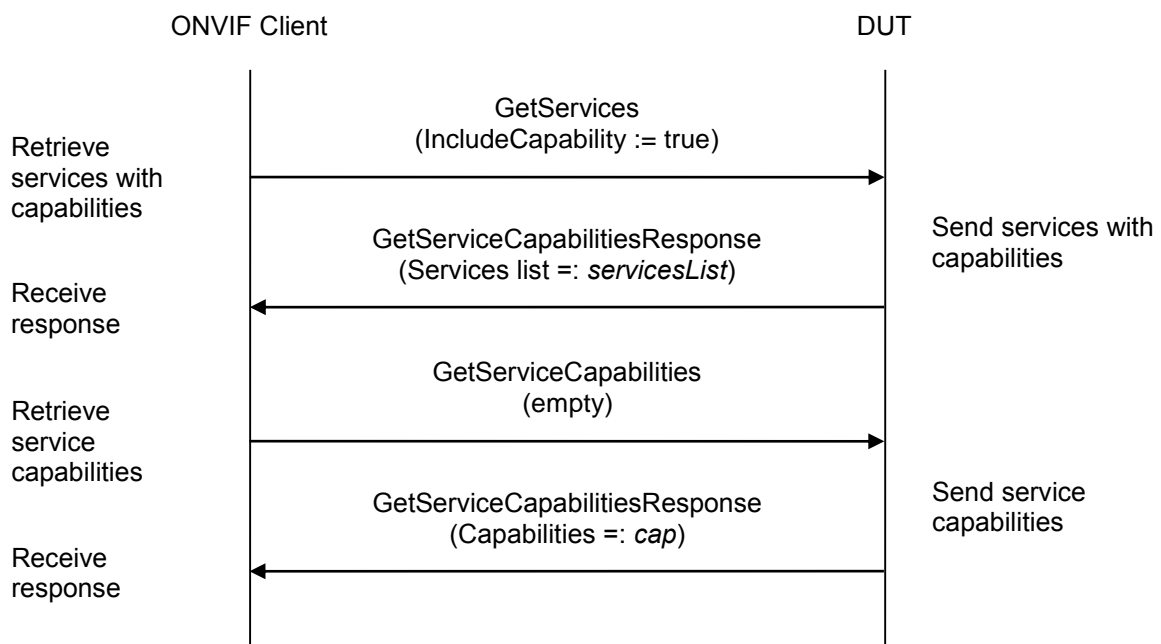
**WSDL Reference:** devicemgmt.wsdl, advancedsecurity.wsdl

**Test Purpose:** To verify Get Services and Advanced Security Service Capabilities consistency.

**Pre-requisite:** None.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



#### Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** with parameters
  - IncludeCapability := true
4. The DUT responds with a **GetServiceCapabilitiesResponse** message with parameters



- Services list =: *servicesList*
5. ONVIF Client selects Service with Service.Namespace = "http://www.onvif.org/ver10/advancedsecurity/wsd!":
- Services list [Namespace = "http://www.onvif.org/ver10/advancedsecurity/wsd!"] =: *advancedSecurityService*
6. ONVIF Client invokes **GetServiceCapabilities**.
7. The DUT responds with a **GetServiceCapabilitiesResponse** message with parameters
- Capabilities =: *cap*
8. If *cap* differs from *advancedSecurityService.Capabilities.Capabilities*, FAIL the test.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetServiceCapabilitiesResponse** message.

**Note:** The following fields are compared at step 8:

- KeystoreCapabilities:
  - SignatureAlgorithms
    - algorithm
    - parameters
  - MaximumNumberOfKeys
  - MaximumNumberOfCertificates
  - MaximumNumberOfCertificationPaths
  - RSAKeyPairGeneration
  - RSAKeyLengths
  - PKCS10ExternalCertificationWithRSA
  - SelfSignedCertificateCreationWithRSA
  - X509Versions
  - MaximumNumberOfPassphrases
  - PKCS8RSAKeyPairUpload
  - PKCS12CertificateWithRSAPrivateKeyUpload
  - PasswordBasedEncryptionAlgorithms



- PasswordBasedMACAlgorithms
- MaximumNumberOfCRLs
- MaximumNumberOfCertificationPathValidationPolicies
- EnforceTLSWebClientAuthExtKeyUsage
- TLSServerCapabilities
  - TLSServerSupported
  - MaximumNumberOfTLSCertificationPaths
  - TLSClientAuthSupported
  - MaximumNumberOfTLSCertificationPathValidationPolicies



## 4.6 Off-Device Key Generation Operations

### 4.6.1 Passphrase Management

#### 4.6.1.1 Upload Passphrase

**Test Label:** Upload Passphrase Verification

**Test Case ID:** ADVANCED\_SECURITY-6-1-1

**ONVIF Core Specification Coverage:** Advanced Security, Passphrase Management

**Command Under Test:** UploadPassphrase (for Advanced Security Service)

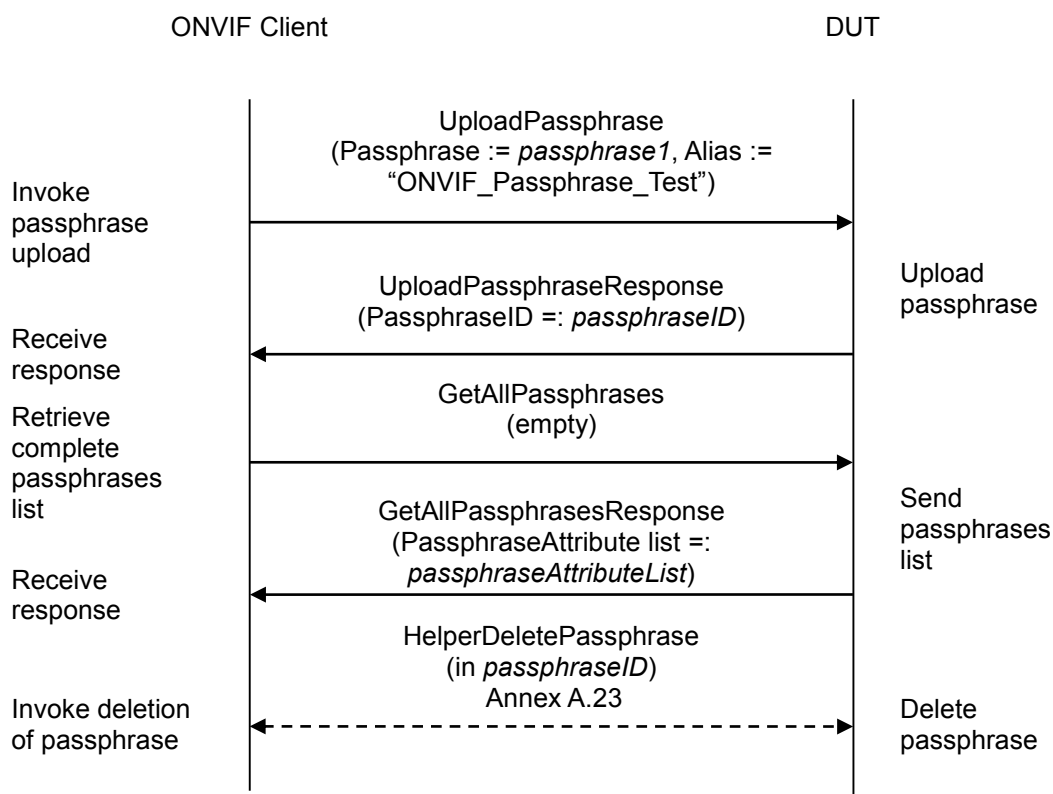
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To verify whether passphrases can be uploaded correctly.

**Pre-requisite:** Advanced Security Service is received from the DUT. Passphrase handling is supported by the DUT as indicated by the MaximumNumberOfPassphrases > 0 capability. The DUT shall have enough free storage capacity for one additional passphrase.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client generates an encryption passphrase *passphrase1* (see Annex A.24).
4. ONVIF Client invokes **UploadPassphrase** with parameters
  - Passphrase := *passphrase1*
  - KeyAlias := "ONVIF\_Passphrase\_Test"
5. The DUT responds with an **UploadPassphraseResponse** message with parameters
  - PassphraseID =: *passphraseID*
6. ONVIF Client invokes **GetAllPassphrases**.
7. The DUT responds with a **GetAllPassphrasesResponse** message with parameters
  - PassphraseAttribute list =: *passphraseAttributeList*
8. If *passphraseAttributeList* does not contain passphrase with PassphraseID equal to *passphraseID*, FAIL the test, and go to the step 10.
9. If passphrase with PassphraseID equal to *passphraseID* from *passphraseAttributeList* has Alias skipped or other than "ONVIF\_Passphrase\_Test", FAIL the test, and go to the step 10.
10. ONVIF Client deletes the passphrase (in *passphraseID*) by following the procedure mentioned in Annex A.23 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **UploadPassphraseResponse** message.

The DUT did not send **GetAllPassphrasesResponse** message.



**4.6.1.2 Delete Passphrase**

**Test Label:** Delete Passphrase Verification

**Test Case ID:** ADVANCED\_SECURITY-6-1-2

**ONVIF Core Specification Coverage:** Advanced Security, Passphrase Management

**Command Under Test:** DeletePassphrase (for Advanced Security Service)

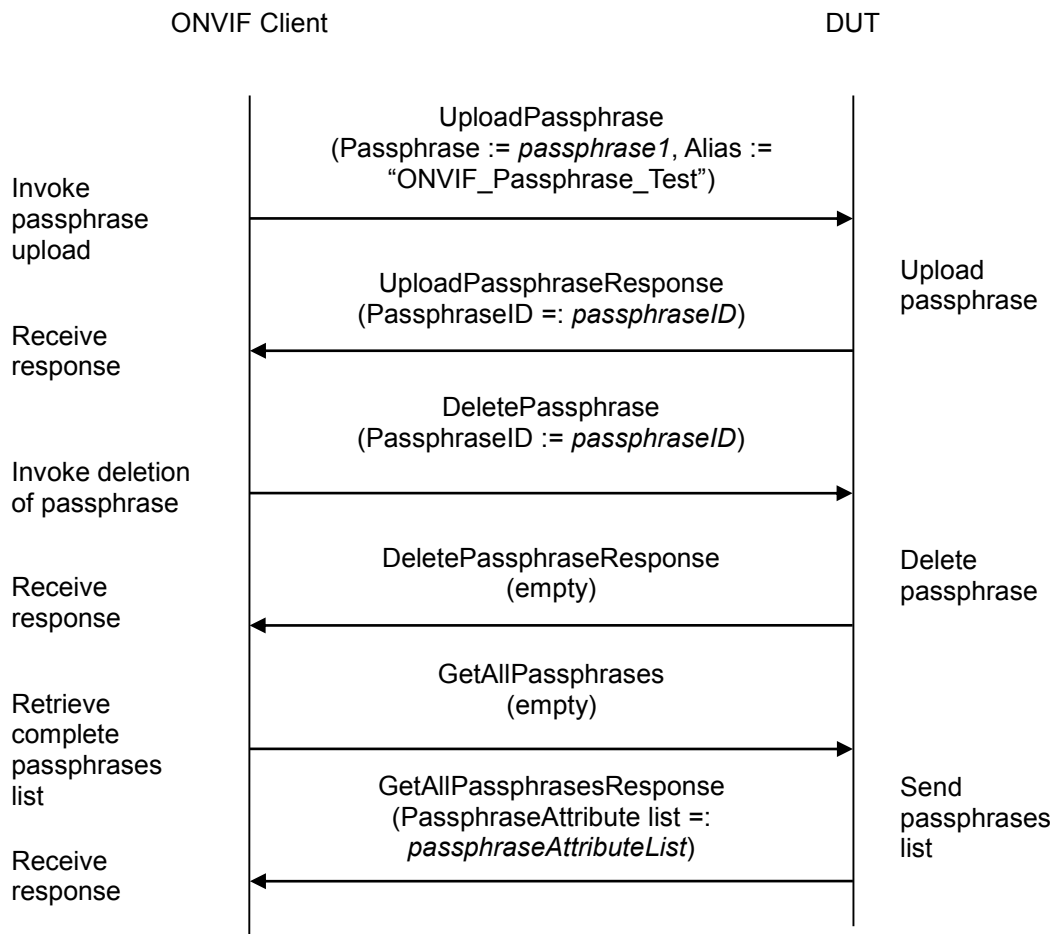
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To verify that a passphrase can be deleted correctly.

**Pre-requisite:** Advanced Security Service is received from the DUT. Passphrase handling is supported by the DUT as indicated by the MaximumNumberOfPassphrases > 0 capability. The DUT shall have enough free storage capacity for one additional passphrase.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**





1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client generates an encryption passphrase *passphrase1* (see Annex A.24).
4. ONVIF Client invokes **UploadPassphrase** with parameters
  - Passphrase := *passphrase1*
  - KeyAlias := "ONVIF\_Passphrase\_Test"
5. The DUT responds with a **UploadPassphraseResponse** message with parameters
  - PassphraseID =: *passphraseID*
6. ONVIF Client invokes **DeletePassphrase** with parameters
  - PassphraseID := *passphraseID*
7. The DUT responds with a **DeletePassphraseResponse** message.
8. ONVIF Client invokes **GetAllPassphrases**.
9. The DUT responds with a **GetAllPassphrasesResponse** message with parameters
  - PassphraseAttribute list =: *passphraseAttributeList*
10. If *passphraseAttributeList* contains passphrase with PassphraseID equal to *passphraseID*, FAIL the test.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **UploadPassphraseResponse** message.

The DUT did not send **GetAllPassphrasesResponse** message.

The DUT did not send **DeletePassphraseResponse** message.



## **4.6.2 Key Management**

### **4.6.2.1 Upload PKCS8 – no key pair exists**

**Test Label:** Upload PKCS8 Verification (No Key Pair Exists)

**Test Case ID:** ADVANCED\_SECURITY-6-2-1

**ONVIF Core Specification Coverage:** Advanced Security, Key Management

**Command Under Test:** UploadKeyPairInPKCS8 (for Advanced Security Service)

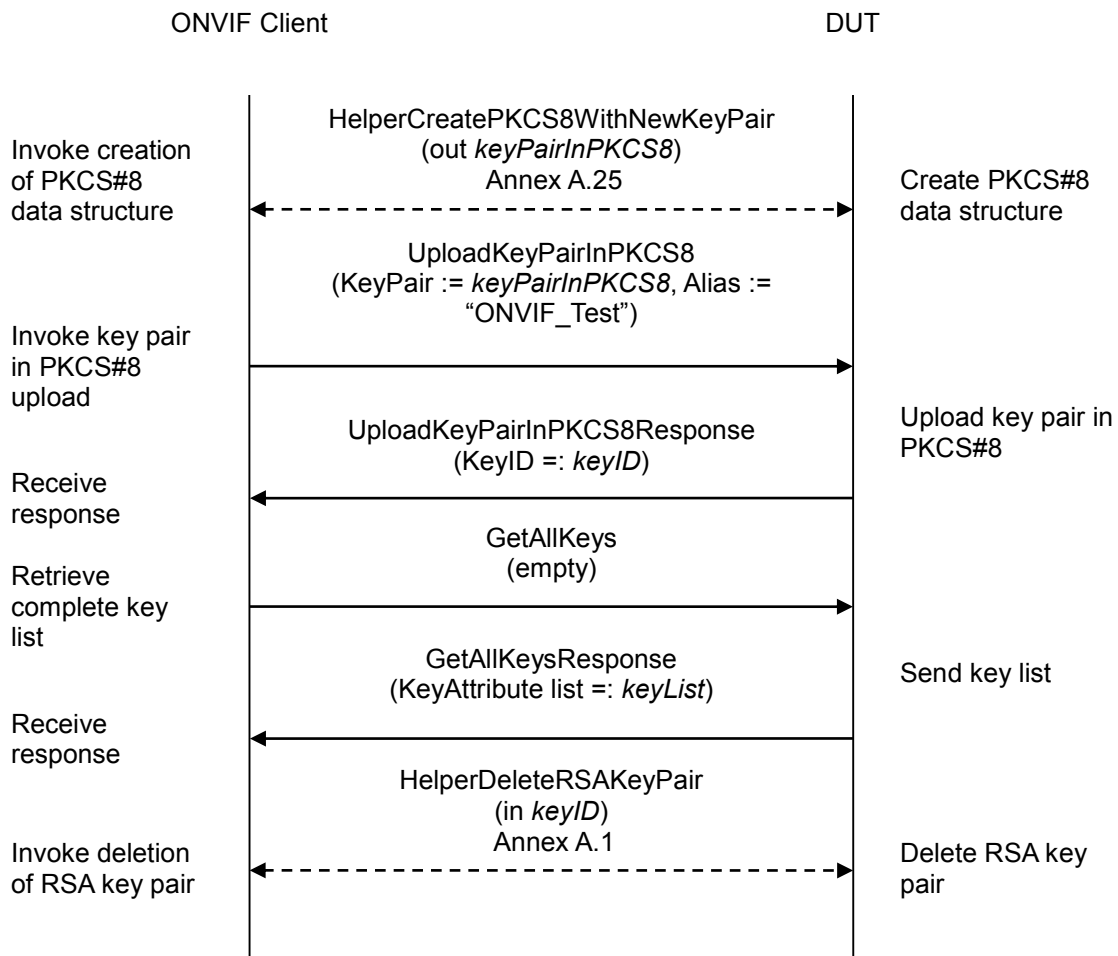
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To verify that a PKCS#8 data structure with new public key and private key can be uploaded correctly.

**Pre-requisite:** Advanced Security Service is received from the DUT. RSA key pair in a PKCS#8 data structure upload is supported by the DUT as indicated by the PKCS8RSAKeyPairUpload capability. The DUT shall have enough free storage capacity for one additional RSA key pair.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client generates a PKCS#8 data structure (out *keyPairInPKCS8*) with new RSA key pair (pubic key (out *publicKey*) and private key (out *privateKey*)) by following the procedure mentioned in Annex A.25.
4. ONVIF Client invokes **UploadKeyPairInPKCS8** with parameters
  - *KeyPair* := *keyPairInPKCS8*
  - *Alias* := "ONVIF\_Test"
  - *EncryptionPassphraseID* skipped
5. The DUT responds with a **UploadKeyPairInPKCS8Response** message with parameters
  - *KeyID* =: *keyID*



6. ONVIF Client invokes **GetAllKeys**.
7. The DUT responds with a **GetAllKeysResponse** message with parameters
  - KeyAttribute list =: *keyList*
8. If *keyList* does not contain KeyAttribute.KeyID =: *keyID*, FAIL the test, and go to the step 11.
9. If KeyAttribute from *keyList* with KeyAttribute.KeyID =: *keyID* has KeyAttribute.hasPrivateKey element that is not equal to *true* or missed, FAIL the test, and go to the step 11.
10. If KeyAttribute from *keyList* with KeyAttribute.KeyID =: *keyID* has KeyAttribute.KeyStatus value other than "ok", FAIL the test, and go to the step 11.
11. ONVIF Client deletes the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.1 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **UploadKeyPairInPKCS8Response** message.

The DUT did not send **GetAllKeysResponse** message.



**4.6.2.2 Upload PKCS8 – decryption fails**

**Test Label:** Upload PKCS8 Verification (Decryption Fails)

**Test Case ID:** ADVANCED\_SECURITY-6-2-2

**ONVIF Core Specification Coverage:** Advanced Security, Key Management

**Command Under Test:** UploadKeyPairInPKCS8 (for Advanced Security Service)

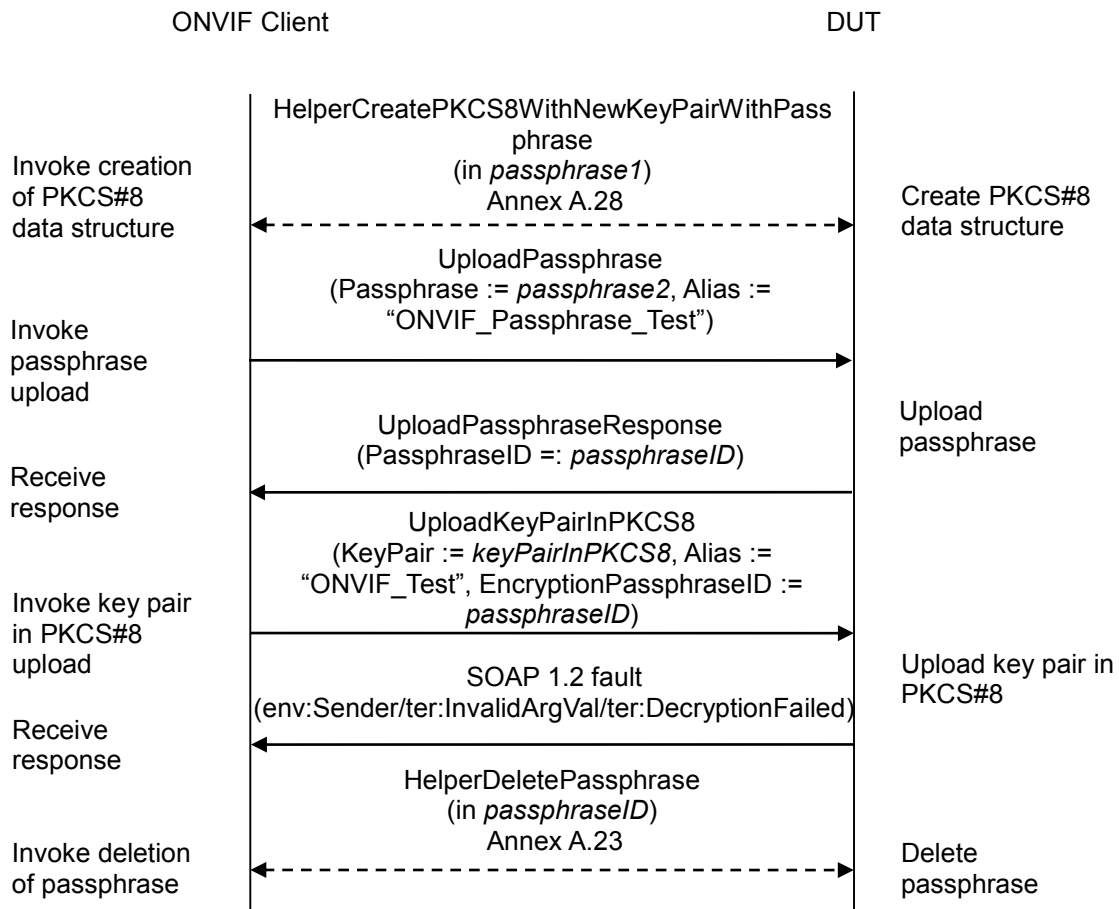
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** To verify that a DecryptionFailed fault is produced when wrong decryption passphrase is used.

**Pre-requisite:** Advanced Security Service is received from the DUT. RSA key pair in a PKCS#8 data structure upload is supported by the DUT as indicated by the PKCS8RSAKeyPairUpload capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional passphrase.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client generates an encryption passphrases *passphrase1* and *passphrase2* (see Annex A.24)
4. ONVIF Client generates a PKCS#8 data structure (out *keyPairInPKCS8*) with new RSA key pair (public key (out *publicKey*) and private key (out *privateKey*)) with encryption passphrase (in *passphrase1*) by following the procedure mentioned in Annex A.28.
5. ONVIF Client invokes **UploadPassphrase** with parameters
  - Passphrase := *passphrase2*
  - KeyAlias := "ONVIF\_Passphrase\_Test"
6. The DUT responds with a **UploadPassphraseResponse** message with parameters
  - PassphraseID =: *passphraseID*
7. ONVIF Client invokes **UploadKeyPairInPKCS8** with parameters
  - KeyPair := *keyPairInPKCS8*
  - Alias := "ONVIF\_Test"
  - EncryptionPassphraseID := *passphraseID*
8. The DUT returns env:Sender/ter:InvalidArgVal/ter:DecryptionFailed SOAP 1.2 fault.
9. ONVIF Client deletes the passphrase (in *passphraseID*) by following the procedure mentioned in Annex A.23 to restore DUT configuration.
10. If the DUT responds with an **UploadKeyPairInPKCS8Response** message with KeyID =: *keyID*, delete the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.1 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **UploadPassphrase** message.

The DUT sent **UploadKeyPairInPKCS8** message.

The DUT did not send the env:Sender/ter:InvalidArgVal/ter:DecryptionFailed SOAP 1.2 fault message.



**4.6.3 Certificate Management**

**4.6.3.1 Upload PKCS12 – no key pair exists**

**Test Label:** Upload PKCS12 Verification (No Key Pair Exists)

**Test Case ID:** ADVANCED\_SECURITY-6-3-1

**ONVIF Core Specification Coverage:** Advanced Security, Certificate Management

**Command Under Test:** UploadCertificateWithPrivateKeyInPKCS12 (for Advanced Security Service)

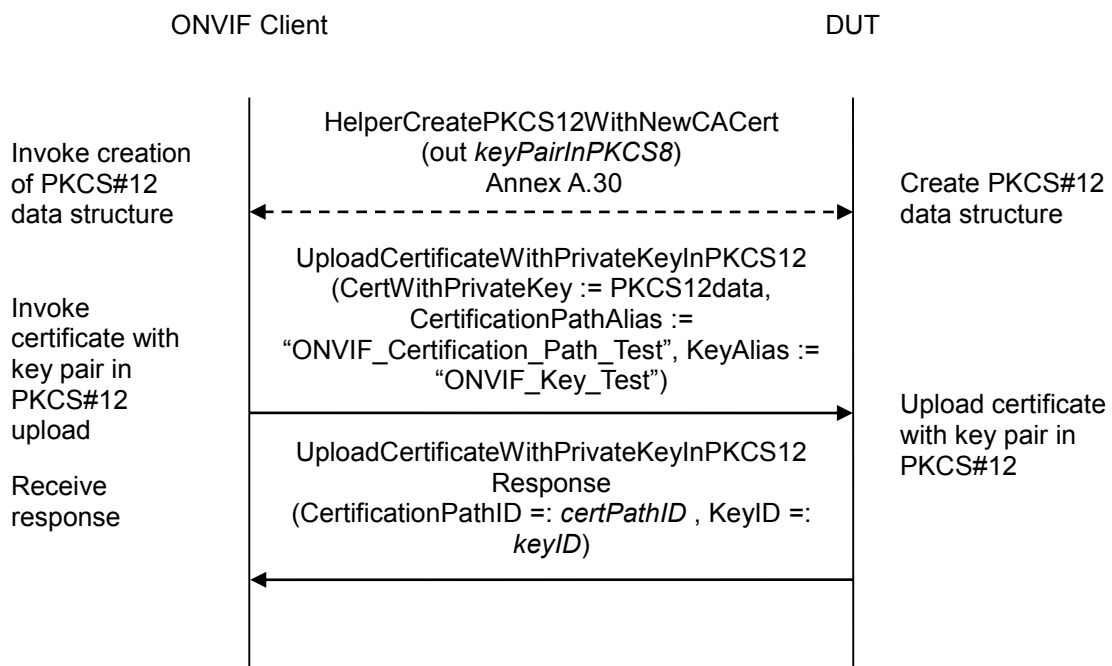
**WSDL Reference:** advancedsecurity.wsdl

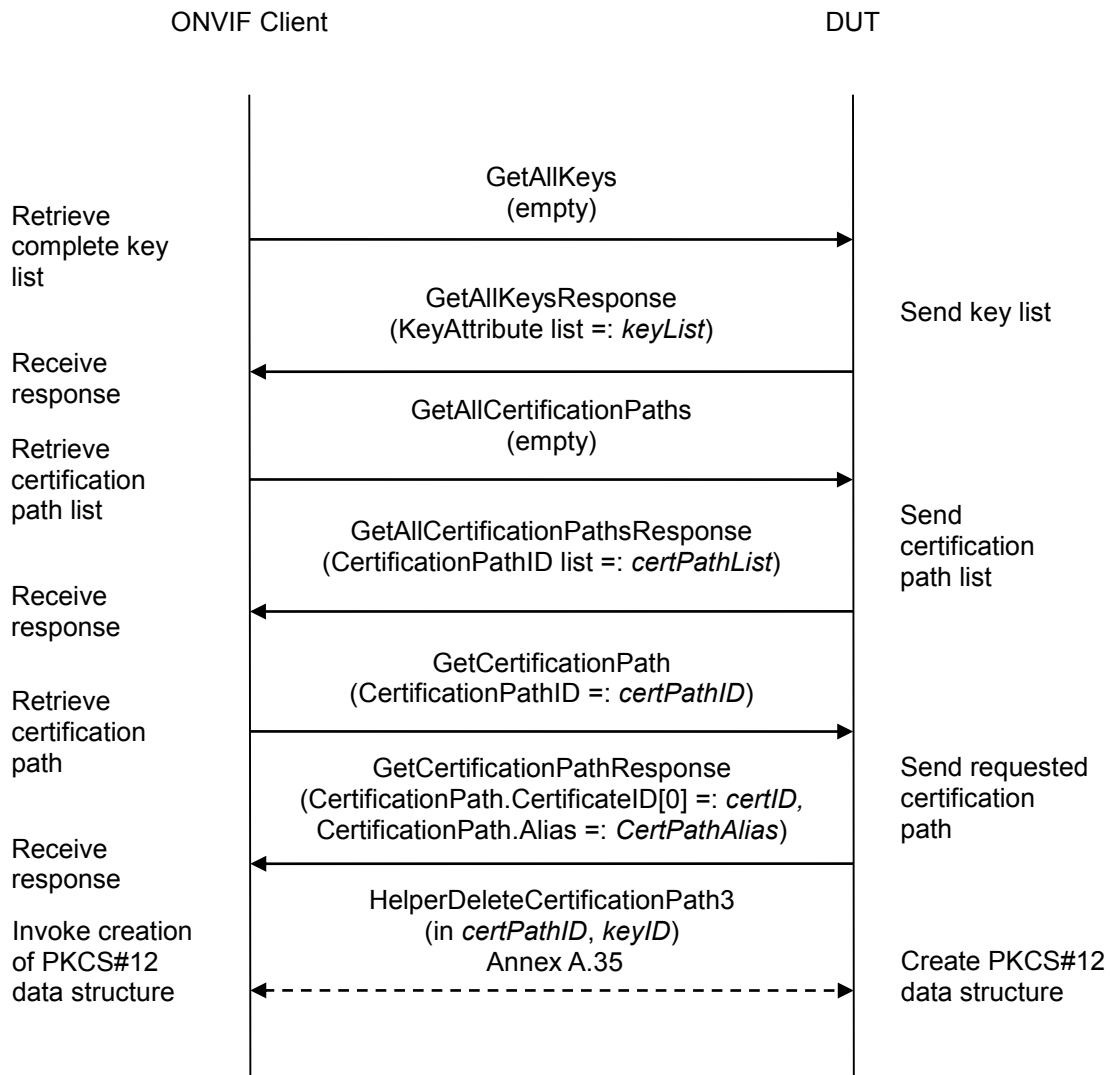
**Test Purpose:** To verify that a PKCS#12 data structure with new public key and private key can be uploaded correctly.

**Pre-requisite:** Advanced Security Service is received from the DUT. Certificate along with an RSA private key in a PKCS#12 data structure upload is supported by the DUT as indicated by the PKCS12CertificateWithRSAPrivateKeyUpload capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding public key (out *publicKey*) in the certificate along with the corresponding private key (out *privateKey*) in the form of a PKCS#12 file (out *PKCS12data*) by following the procedure described in Annex A.30.
4. ONVIF Client invokes **UploadCertificateWithPrivateKeyInPKCS12** with parameters
  - CertWithPrivateKey := *PKCS12data*
  - CertificationPathAlias := "ONVIF\_Certification\_Path\_Test"
  - KeyAlias := "ONVIF\_Key\_Test"





- IgnoreAdditionalCertificates := false
  - IntegrityPassphraseID skipped
  - EncryptionPassphraseID skipped
5. The DUT responds with a **UploadCertificateWithPrivateKeyInPKCS12Response** message with parameters
    - CertificationPathID =: *certPathID*
    - KeyID =: *keyID*
  6. ONVIF Client invokes **GetAllKeys**.
  7. The DUT responds with a **GetAllKeysResponse** message with parameters
    - KeyAttribute list =: *keyList*
  8. If *keyList* does not contain key with KeyID equal to *keyID*, FAIL the test, and go to the step 19.
  9. If key with KeyID equal to *keyID* from *keyList* has hasPrivateKey equal to false or has skipped hasPrivateKey, FAIL the test, and go to the step 19.
  10. If key with KeyID equal to *keyID* from *keyList* has Alias skipped or other than "ONVIF\_Key\_Test", FAIL the test, and go to the step 19.
  11. If key with KeyID equal to *keyID* from *keyList* has KeyStatus other than "ok", FAIL the test, and go to the step 19.
  12. ONVIF Client invokes **GetAllCertificationPaths**.
  13. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters
    - CertificationPathID list =: *certPathList*
  14. If *certPathList* does not contain certification path with CertificationPathID equal to *certPathID*, FAIL the test, and go to the step 19.
  15. ONVIF Client invokes **GetCertificationPath** message with parameters
    - CertificationPathID =: *certPathID*
  16. The DUT responds with a **GetCertificationPathResponse** message with parameters
    - CertificationPath.CertificateID[0] =: *certID*
    - CertificationPath.Alias =: *CertPathAlias*
  17. If *CertPathAlias* Alias skipped or other than "ONVIF\_CertificationPath\_Test", FAIL the test, and go to the step 20.
  18. If received CertificationPath contains more than one CertificateID item, FAIL the test, and go to the step 20.
  19. ONVIF Client deletes the certification path (in *certPathID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.35 to restore DUT configuration and finish the test.



20. ONVIF Client deletes the certification path (in *certPathID*) and related CA certificate (in *certID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.5 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **UploadCertificateWithPrivateKeyInPKCS12Response** message.

The DUT did not send **GetAllKeysResponse** message.

The DUT did not send **GetAllCertificationPathsResponse** message.

The DUT did not send **GetCertificationPathResponse** message.



**4.6.3.2 Upload PKCS12 – decryption fails**

**Test Label:** Upload PKCS12 Verification (Decryption Fails)

**Test Case ID:** ADVANCED\_SECURITY-6-3-2

**ONVIF Core Specification Coverage:** Advanced Security, Certificate Management

**Command Under Test:** UploadCertificateWithPrivateKeyInPKCS12 (for Advanced Security Service)

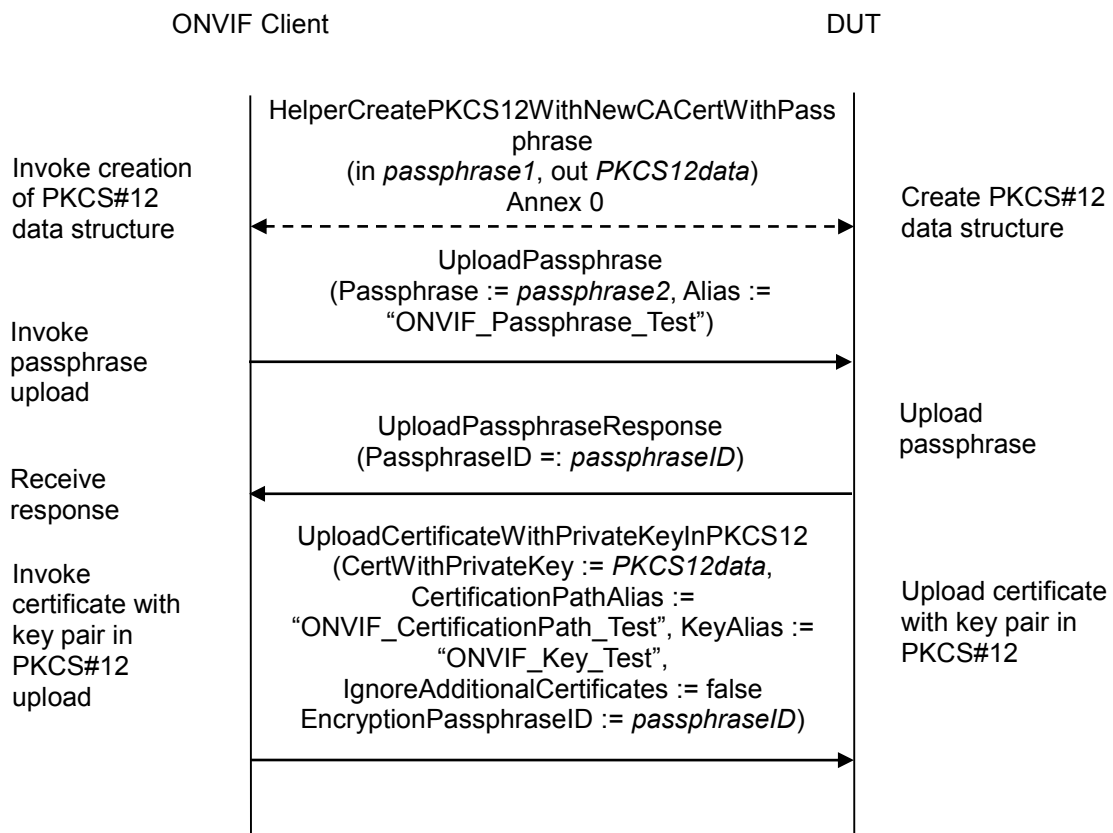
**WSDL Reference:** advancedsecurity.wsdl

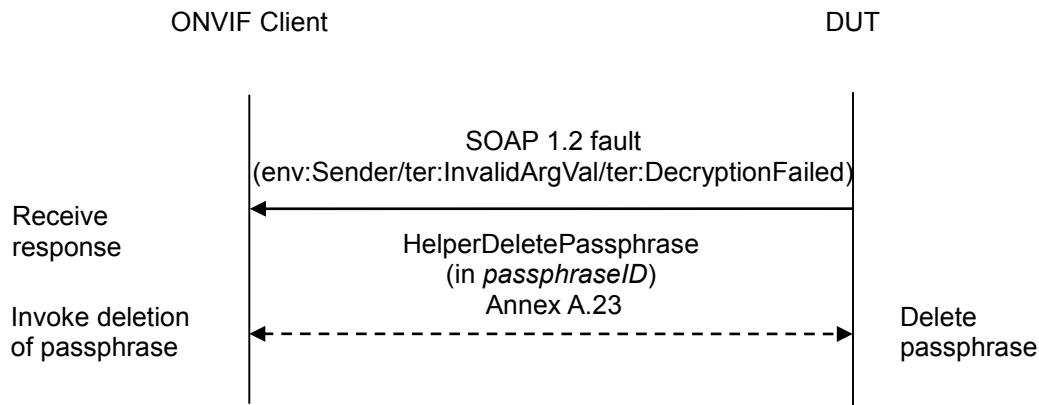
**Test Purpose:** To verify that a DecryptionFailed fault is produced when wrong decryption passphrase is used.

**Pre-requisite:** Advanced Security Service is received from the DUT. Certificate along with an RSA private key in a PKCS#12 data structure upload is supported by the DUT as indicated by the PKCS12CertificateWithRSAPrivateKeyUpload capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. The DUT shall have enough free storage capacity for one additional passphrase.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





### Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client generates an encryption passphrases *passphrase1* and *passphrase2* (see Annex A.24)
4. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding public key (out *publicKey*) in the certificate along with the corresponding private key (out *privateKey*) in the form of a PKCS#12 file (out *PKCS12data*) encrypted with passphrase (in *passphrase1*) by following the procedure described in Annex A.31.
5. ONVIF Client invokes **UploadPassphrase** with parameters
  - Passphrase := *passphrase2*
  - KeyAlias := "ONVIF\_Passphrase\_Test"
6. The DUT responds with a **UploadPassphraseResponse** message with parameters
  - PassphraseID := *passphraseID*
7. ONVIF Client invokes **UploadCertificateWithPrivateKeyInPKCS12** with parameters
  - CertWithPrivateKey := *PKCS12data*
  - CertificationPathAlias := "ONVIF\_CertificationPath\_Test"
  - KeyAlias := "ONVIF\_Key\_Test"
  - IgnoreAdditionalCertificates := false
  - IntegrityPassphraseID skipped
  - EncryptionPassphraseID := *passphraseID*
8. The DUT returns env:Sender\ter:InvalidArgVal\ter:DecryptionFailed SOAP 1.2 fault.
9. ONVIF Client deletes the passphrase (in *passphraseID*) by following the procedure mentioned in



Annex A.23 to restore DUT configuration.

10. If the DUT responds with a **UploadCertificateWithPrivateKeyInPKCS12Response** message (CertificationPathID =: *certPathID*, KeyID =: *keyID*), ONVIF Client deletes the certification path (in *certPathID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.35 to restore DUT configuration and finish the test.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **UploadPassphrase** message.

The DUT sent **UploadCertificateWithPrivateKeyInPKCS12** message.

The DUT did not send **env:SenderInter:InvalidArgValInter:DecryptionFailed** SOAP 1.2 fault.



### 4.6.3.3 Upload PKCS12 - verify key and certificate

**Test Label:** Upload PKCS12 - Verify Key and Certificate

**Test Case ID:** ADVANCED\_SECURITY-6-3-4

**ONVIF Core Specification Coverage:** Advanced Security, Certificate Management

**Command Under Test:** UploadCertificateWithPrivateKeyInPKCS12, GetKeyStatus, GetCertificate, GetAllCertificates

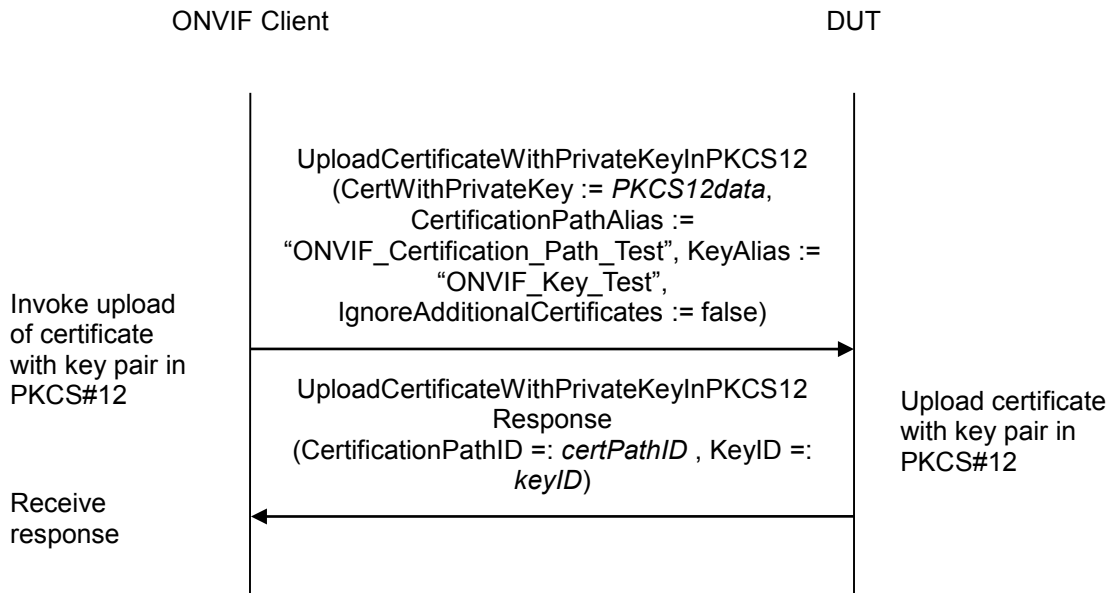
**WSDL Reference:** advancedsecurity.wsdl

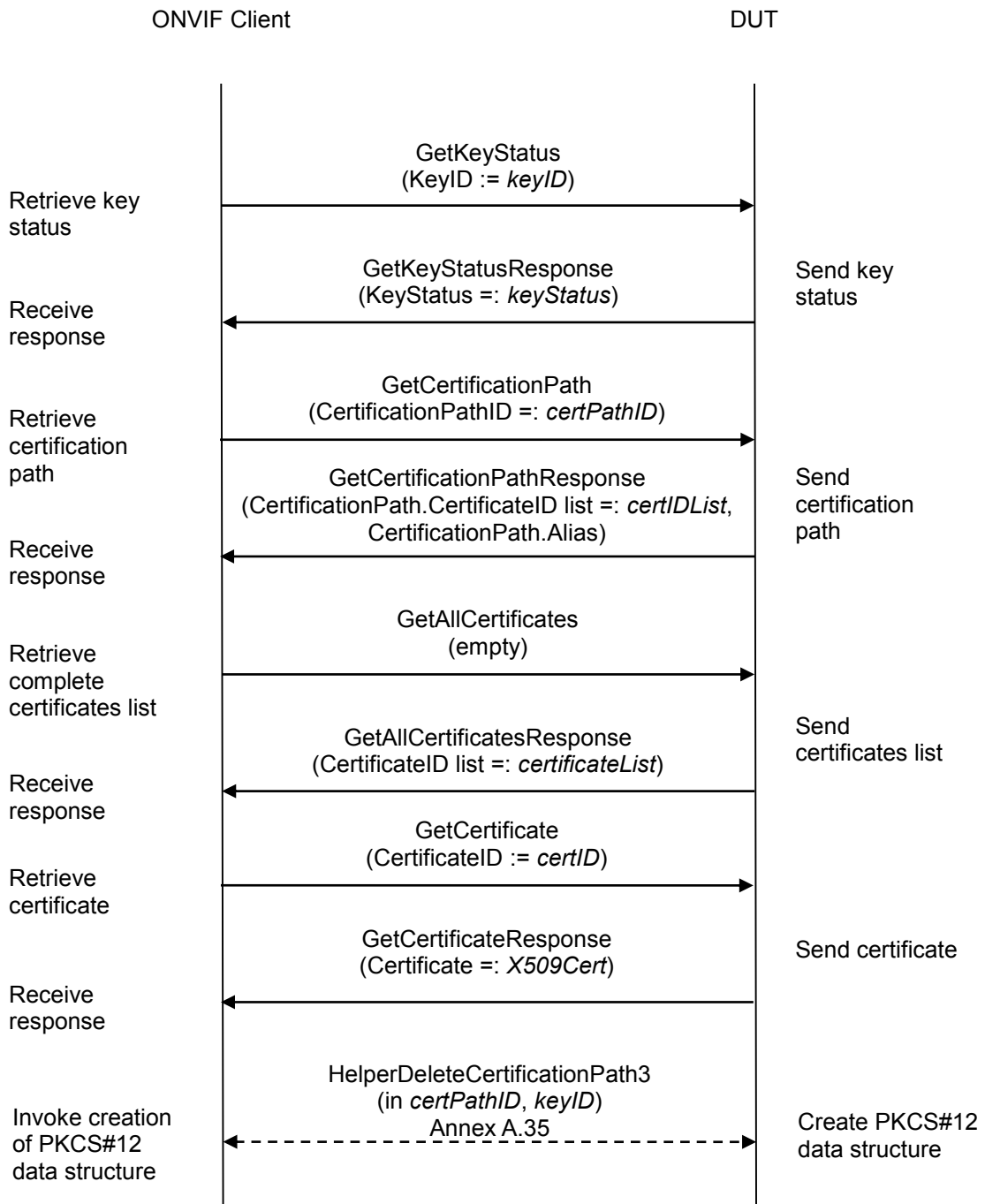
**Test Purpose:** Verify that the DUT correctly integrates keys and certificates, which have been uploaded in a PKCS#12 data structure, into the keystore.

**Pre-requisite:** Advanced Security Service is received from the DUT. Certificate along with an RSA private key in a PKCS#12 data structure upload is supported by the DUT as indicated by the PKCS12CertificateWithRSAPrivateKeyUpload capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. The DUT shall have enough free storage capacity for one additional certification path.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding public key (out



*publicKey*) in the certificate along with the corresponding private key (out *privateKey*) in the form of a PKCS#12 file (out *PKCS12data*) by following the procedure described in Annex A.30.

4. ONVIF Client invokes **UploadCertificateWithPrivateKeyInPKCS12** with parameters
  - CertWithPrivateKey := *PKCS12data*
  - CertificationPathAlias := "ONVIF\_Certification\_Path\_Test"
  - KeyAlias := "ONVIF\_Key\_Test"
  - IgnoreAdditionalCertificates := false
  - IntegrityPassphraseID skipped
  - EncryptionPassphraseID skipped
5. The DUT responds with a **UploadCertificateWithPrivateKeyInPKCS12Response** message with parameters
  - CertificationPathID =: *certPathID*
  - KeyID =: *keyID*
6. ONVIF Client invokes **GetKeyStatus** with parameters
  - KeyID := *keyID*
7. The DUT responds with **GetKeyStatusResponse** message with parameters
  - KeyStatus =: *keyStatus*
8. If *keyStatus* is not equal to "ok", FAIL the test, and go to the step 23.
9. ONVIF Client invokes **GetCertificationPath** message with parameters
  - CertificationPathID =: *certPathID*
10. The DUT responds with a **GetCertificationPathResponse** message with parameters
  - CertificationPath.CertificateID list =: *certIDList*
  - CertificationPath.Alias
11. If *certIDList* contains more item than one, FAIL the test and go to the step 23.
12. ONVIF Client invokes **GetAllCertificates**.
13. The DUT responds with a **GetAllCertificatesResponse** message with parameters
  - CertificateID list =: *certificateList*
14. If *certificateList* does not contain certificate with Certificate.CertificateID equal to *certIDList*[0], FAIL the test and go to the step 23.
15. Set:
  - *certificateList*.Certificate[CertificateID = *certIDList*[0]] =: *X509Cert*





16. If *X509Cert.KeyID* is not equal to *keyID*, FAIL the test and go to the step 23.
17. If *X509Cert.CertificateContent* is not equal to *CAcert*, FAIL the test and go to the step 23.
18. ONVIF Client invokes **GetCertificate** message with parameters
  - *CertificateID* := *certIDList[0]*
19. The DUT responds with a **GetCertificateResponse** message with parameters
  - *Certificate* =: *X509Cert*
20. If *X509Cert.CertificateID* is not equal to *certIDList[0]*, FAIL the test and go to the step 23.
21. If *X509Cert.KeyID* is not equal to *keyID*, FAIL the test and go to the step 23.
22. If *X509Cert.CertificateContent* is not equal to *CAcert*, FAIL the test and go to the step 23.
23. ONVIF Client deletes the certification path (in *certPathID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.35 to restore DUT configuration and skip other steps.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **UploadCertificateWithPrivateKeyInPKCS12Response** message.

The DUT did not send **GetKeyStatusResponse** message.

The DUT did not send **GetCertificationPathResponse** message.

The DUT did not send **GetAllCertificatesResponse** message.

The DUT did not send **GetCertificateResponse** message.



## 4.7 Certificate-based Client Authentication

### 4.7.1 Upload CRL

**Test Label:** Upload CRL

**Test Case ID:** ADVANCED\_SECURITY-8-1-1

**ONVIF Core Specification Coverage:** Advanced Security, Upload Certificate Revocation List

**Command Under Test:** UploadCRL, GetAllCRLs

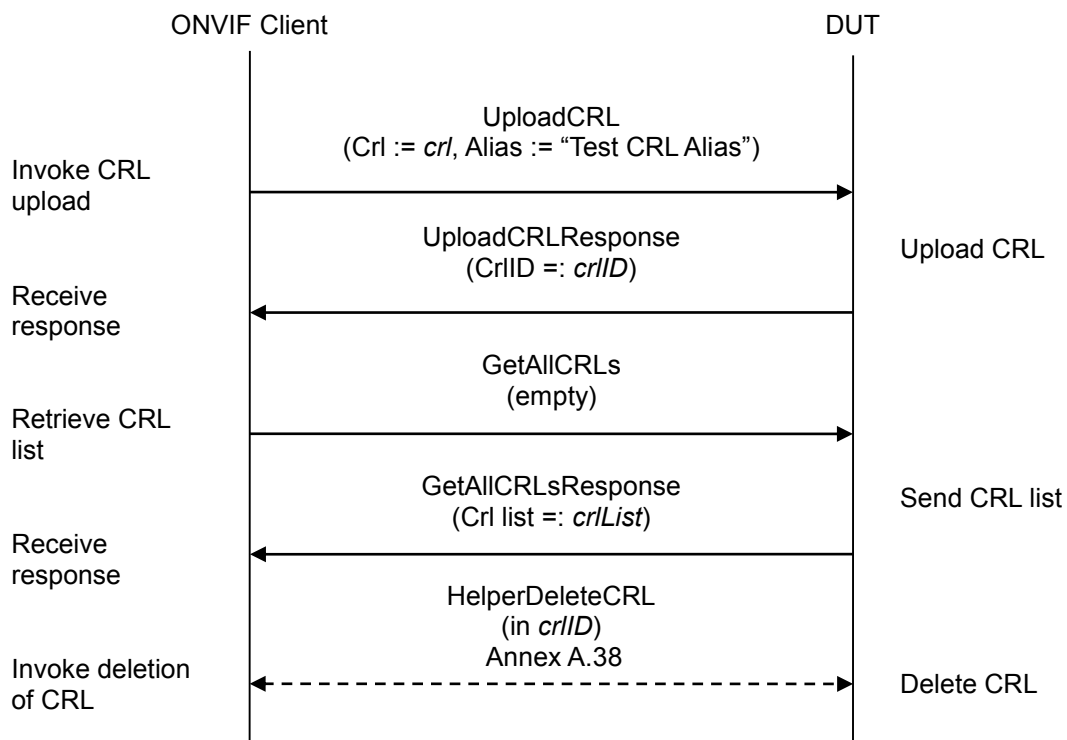
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** Verify that CRLs can be uploaded to the DUT.

**Pre-requisite:** Advanced Security Service is received from the DUT. CRLs supported by the DUT as indicated by the MaximumNumberOfCRLs capability. The DUT shall have enough free storage capacity for one additional CRL.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.



3. ONVIF Client creates a CRL (out *crI*) by following the procedure mentioned in Annex A.37.
4. ONVIF Client invokes **UploadCRL** with parameter
  - *CrI* := *crI*
  - *Alias* := "ONVIF\_CRL\_Test"
  - *anyParameters* skipped
5. The DUT responds with **UploadCRLResponse** message with parameters
  - *CrIID* =: *crIID*
6. ONVIF Client invokes **GetAllCRLs**.
7. The DUT responds with a **GetAllCRLsResponse** message with parameters
  - *CrI list* =: *crIList*
8. If *crIList* does not contain *crIID*, FAIL the test, and go to the step 11.
9. If *crIList*[*CRLID* = *crIID*].*Alias* is not equal to "ONVIF\_CRL\_Test", FAIL the test, and go to the step 11.
10. If *crIList*[*CRLID* = *crIID*].*CRLContent* is not equal to *crI*, FAIL the test, and go to the step 11.
11. ONVIF Client deletes the CRL (in *crIID*) by following the procedure mentioned in Annex A.38 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **UploadCRLResponse** message.

The DUT did not send **GetAllCRLsResponse** message.



**4.7.2 Delete CRL**

**Test Label:** Delete CRL

**Test Case ID:** ADVANCED\_SECURITY-8-1-2

**ONVIF Core Specification Coverage:** Advanced Security, Delete Certificate Revocation List

**Command Under Test:** DeleteCRL, GetAllCRLs

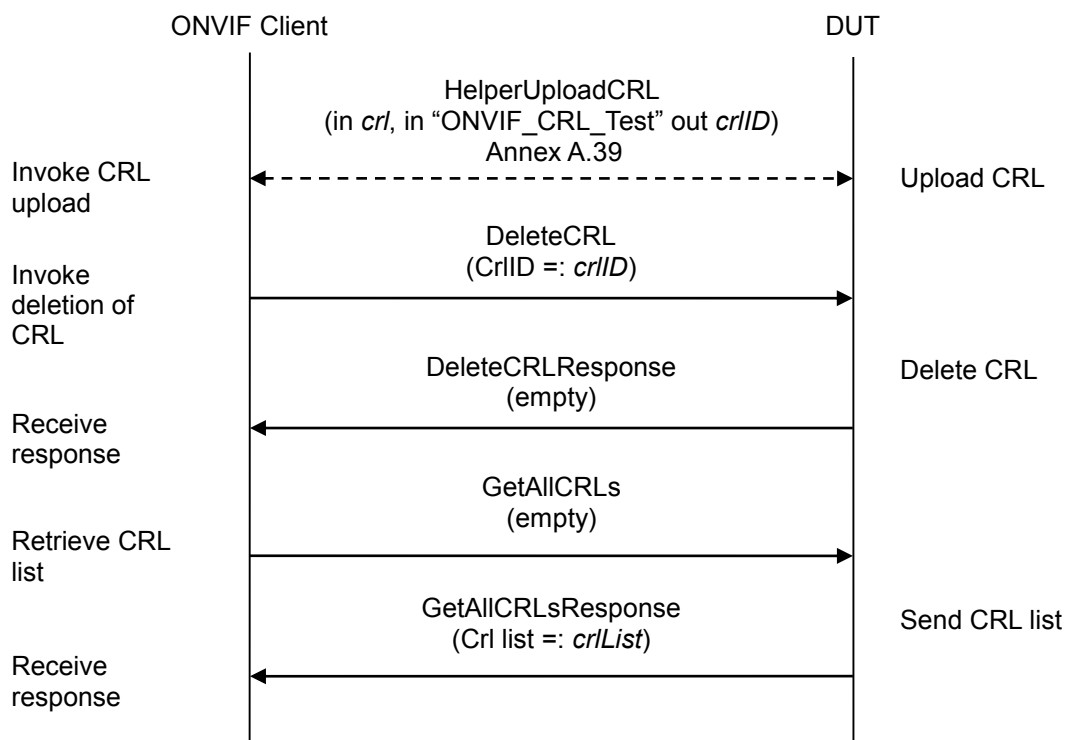
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** Verify that CRLs can be deleted from the DUT.

**Pre-requisite:** Advanced Security Service is received from the DUT. CRLs supported by the DUT as indicated by the MaximumNumberOfCRLs capability. The DUT shall have enough free storage capacity for one additional CRL.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a CRL (out *crl*) by following the procedure mentioned in Annex A.37.



4. ONVIF Client uploads a CRL (in *crI*) with alias (in “ONVIF\_CRL\_Test”) identifier (out *crIID*) by following the procedure described in Annex A.39.
5. ONVIF Client invokes **DeleteCRL** with parameters
  - *CrIID* =: *crIID*
6. The DUT responds with a **DeleteCRLResponse** message.
7. ONVIF Client invokes **GetAllCRLs**.
8. The DUT responds with a **GetAllCRLsResponse** message with parameters
  - *CrI list* =: *crIList*
9. If *crIList* contains *crIID*, FAIL the test.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **DeleteCRLResponse** message.

The DUT did not send **GetAllCRLsResponse** message.



**4.7.3 Get CRL**

**Test Label:** Get CRL

**Test Case ID:** ADVANCED\_SECURITY-8-1-3

**ONVIF Core Specification Coverage:** Advanced Security, 7.6.2 Get Certificate Revocation List

**Command Under Test:** GetCRL

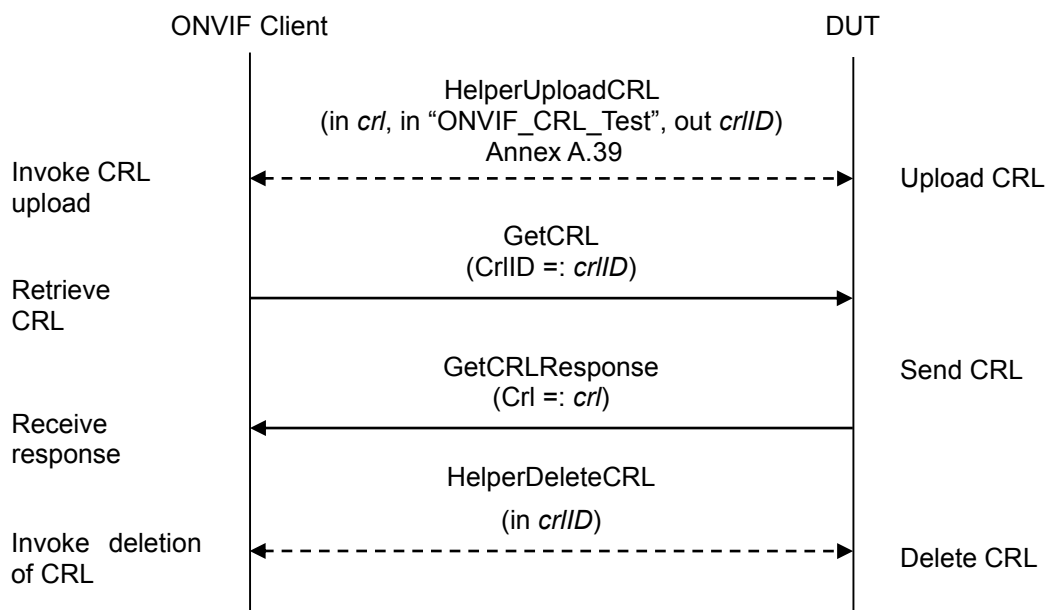
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** Verify that CRLs can be retrieved from the DUT.

**Pre-requisite:** Advanced Security Service is received from the DUT. CRLs supported by the DUT as indicated by the MaximumNumberOfCRLs capability. The DUT shall have enough free storage capacity for one additional CRL.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates a CRL (out *crI*) by following the procedure mentioned in Annex A.37.
4. ONVIF Client uploads a CRL (in *crI*) with alias (in "ONVIF\_CRL\_Test") identifier (out *crIID*) by following the procedure described in Annex A.39.



5. ONVIF Client invokes **GetCRL** with parameters
  - CrlID =: *crlID*
6. The DUT responds with a **GetCRLResponse** message with parameters
  - Crl =: *crl*
7. If *crl.CRLID* is not equal to *crlID*, FAIL the test, and go to the step 10.
8. If *crl.Alias* is not equal to "ONVIF\_CRL\_Test", FAIL the test, and go to the step 10.
9. If *crl.CRLContent* is not equal to *crl*, FAIL the test, and go to the step 10.
10. ONVIF Client deletes the CRL (in *crlID*) by following the procedure mentioned in Annex A.38 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetCRLResponse** message.



**4.7.4 Create certification path validation policy**

**Test Label:** Create Certification Path Validation Policy

**Test Case ID:** ADVANCED\_SECURITY-8-1-4

**ONVIF Core Specification Coverage:** Advanced Security, Create Certification Path Validation Policy

**Command Under Test:** CreateCertPathValidationPolicy, GetAllCertPathValidationPolicies

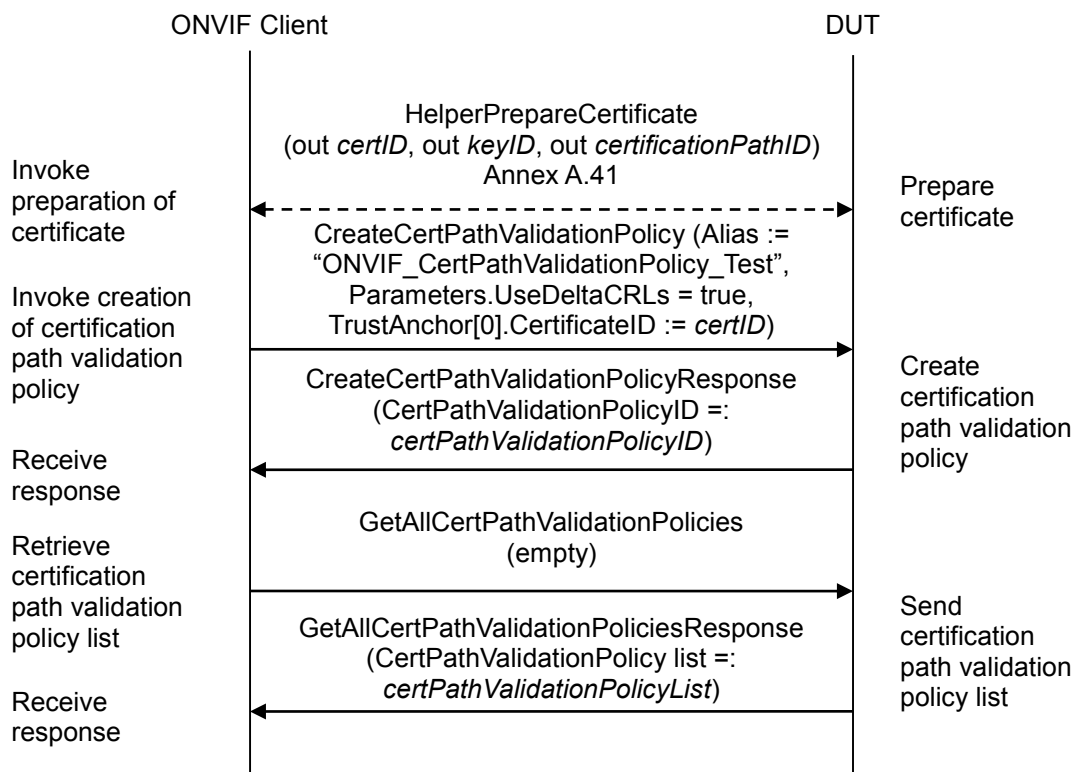
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** Verify that a certification path validation policy can be created on the DUT.

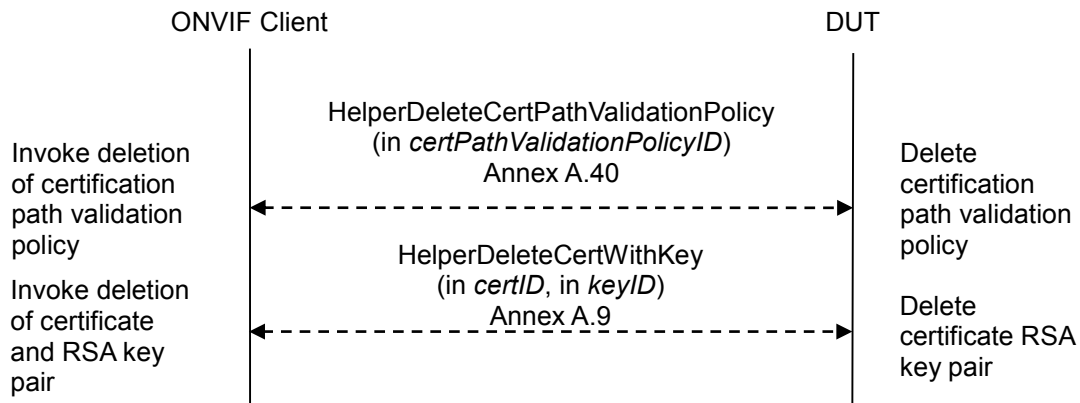
**Pre-requisite:** Advanced Security Service is received from the DUT. Certification path validation policy supported by the DUT as indicated by the MaximumNumberOfCertificationPathValidationPolicies capability. The DUT shall have enough free storage capacity for one additional certification path validation policy. The DUT shall have enough free storage capacity for one additional certificate. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**







**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client prepares certificate (out *certID*), related RSA key pair (out *keyID*) and certification path if any (out *certificationPathID*) by following the procedure mentioned in Annex A.41.
4. ONVIF Client invokes **CreateCertPathValidationPolicy** with parameter
  - Alias := "ONVIF\_CertPathValidationPolicy\_Test"
  - Parameters.RequireTLSWWWClientAuthExtendedKeyUsage skipped
  - Parameters.UseDeltaCRLs = true
  - Parameters.anyParameters skipped
  - TrustAnchor[0].CertificateID := *certID*
  - anyParameters skipped
5. The DUT responds with **CreateCertPathValidationPolicyResponse** message with parameters
  - CertPathValidationPolicyID =: *certPathValidationPolicyID*
6. ONVIF Client invokes **GetAllCertPathValidationPolicies**.
7. The DUT responds with a **GetAllCertPathValidationPoliciesResponse** message with parameters
  - CertPathValidationPolicy list =: *certPathValidationPolicyList*
8. If *certPathValidationPolicyList* does not contain *certPathValidationPolicyID*, FAIL the test, and go to the step 13.
9. If *certPathValidationPolicyList*[CertPathValidationPolicyID = *certPathValidationPolicyID*].Alias is not equal to "ONVIF\_CertPathValidationPolicy\_Test", FAIL the test, and go to the step 13.



10. If  $\text{certPathValidationPolicyList}[\text{CertPathValidationPolicyID} = \text{certPathValidationPolicyID}].\text{Parameters.RequireTLSWWWClientAuthExtendedKeyUsage}$  is equal to true, FAIL the test, and go to the step 13.
11. If  $\text{certPathValidationPolicyList}[\text{CertPathValidationPolicyID} = \text{certPathValidationPolicyID}].\text{Parameters.UseDeltaCRLs}$  is not equal to true, FAIL the test, and go to the step 13.
12. If  $\text{certPathValidationPolicyList}[\text{CertPathValidationPolicyID} = \text{certPathValidationPolicyID}].\text{TrustAnchor}$  does not contain one and only one element with `CertificateID` equal to *certID*, FAIL the test, and go to the step 13.
13. ONVIF Client deletes the certification path validation policy (in *certPathValidationPolicyID*) by following the procedure mentioned in Annex A.40 to restore DUT configuration.
14. If *certificationPathID* is null:
  - 14.1. ONVIF Client deletes the self-signed certificate (in *certID*) and related the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.9.
  - 14.2. Skip other steps.
15. If *certificationPathID* is not null:
  - 15.1. ONVIF Client deletes the certification path (in *certificationPathID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.35 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **CreateCertPathValidationPolicyResponse** message.

The DUT did not send **GetAllCertPathValidationPoliciesResponse** message.



**4.7.5 Get certification path validation policy**

**Test Label:** Get Certification Path Validation Policy

**Test Case ID:** ADVANCED\_SECURITY-8-1-5

**ONVIF Core Specification Coverage:** Advanced Security, Get Certification Path Validation Policy

**Command Under Test:** CreateCertPathValidationPolicy, GetCertPathValidationPolicy

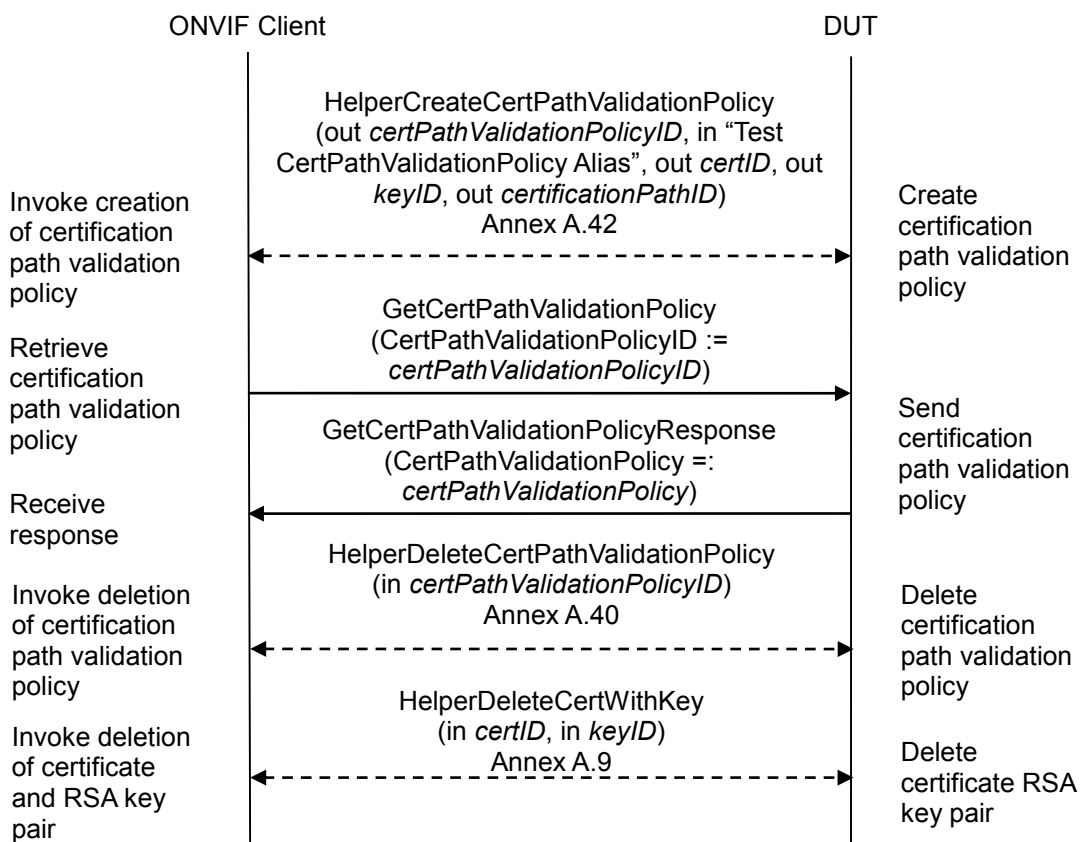
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** Verify that certification path validation policies can be retrieved from the DUT.

**Pre-requisite:** Advanced Security Service is received from the DUT. Certification path validation policy supported by the DUT as indicated by the MaximumNumberOfCertificationPathValidationPolicies capability. The DUT shall have enough free storage capacity for one additional certification path validation policy. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**





**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates certification path validation policy identifier (out *certPathValidationPolicyID*) with specified alias (in “Test CertPathValidationPolicy Alias”), related certificate (out *certID*), RSA key pair (out *keyID*) and certification path if any (out *certificationPathID*) by following the procedure mentioned in Annex A.42.
4. ONVIF Client invokes **GetCertPathValidationPolicy** with parameters
  - *CertPathValidationPolicyID* := *certPathValidationPolicyID*
5. The DUT responds with a **GetCertPathValidationPolicyResponse** message with parameters
  - *CertPathValidationPolicy* =: *certPathValidationPolicy*
6. If *certPathValidationPolicy.CertPathValidationPolicyID* is not equal to *certPathValidationPolicyID*, FAIL the test, and go to the step 11.
7. If *certPathValidationPolicy.Alias* is not equal to “Test CertPathValidationPolicy Alias”, FAIL the test, and go to the step 11.
8. If *certPathValidationPolicy.Parameters.RequireTLSWWWClientAuthExtendedKeyUsage* is equal to true, FAIL the test, and go to the step 11.
9. If *certPathValidationPolicy.Parameters.UseDeltaCRLs* is not equal to true, FAIL the test, and go to the step 11.
10. If *certPathValidationPolicy.TrustAnchor* does not contain one and only one element with *CertificateID* equal to *certID*, FAIL the test, and go to the step 11.
11. ONVIF Client deletes the certification path validation policy (in *certPathValidationPolicyID*) by following the procedure mentioned in Annex A.40 to restore DUT configuration.
12. If *certificationPathID* is null:
  - 12.1. ONVIF Client deletes the self-signed certificate (in *certID*) and related the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.9.
  - 12.2. Skip other steps.
13. If *certificationPathID* is not null:
  - 13.1. ONVIF Client deletes the certification path (in *certificationPathID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.35 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetCertPathValidationPolicyResponse** message.



#### **4.7.6 Delete certification path validation policy**

**Test Label:** Delete Certification Path Validation Policy

**Test Case ID:** ADVANCED\_SECURITY-8-1-6

**ONVIF Core Specification Coverage:** Advanced Security, Delete Certification Path Validation Policy

**Command Under Test:** DeleteCertPathValidationPolicy, GetAllCertPathValidationPolicies

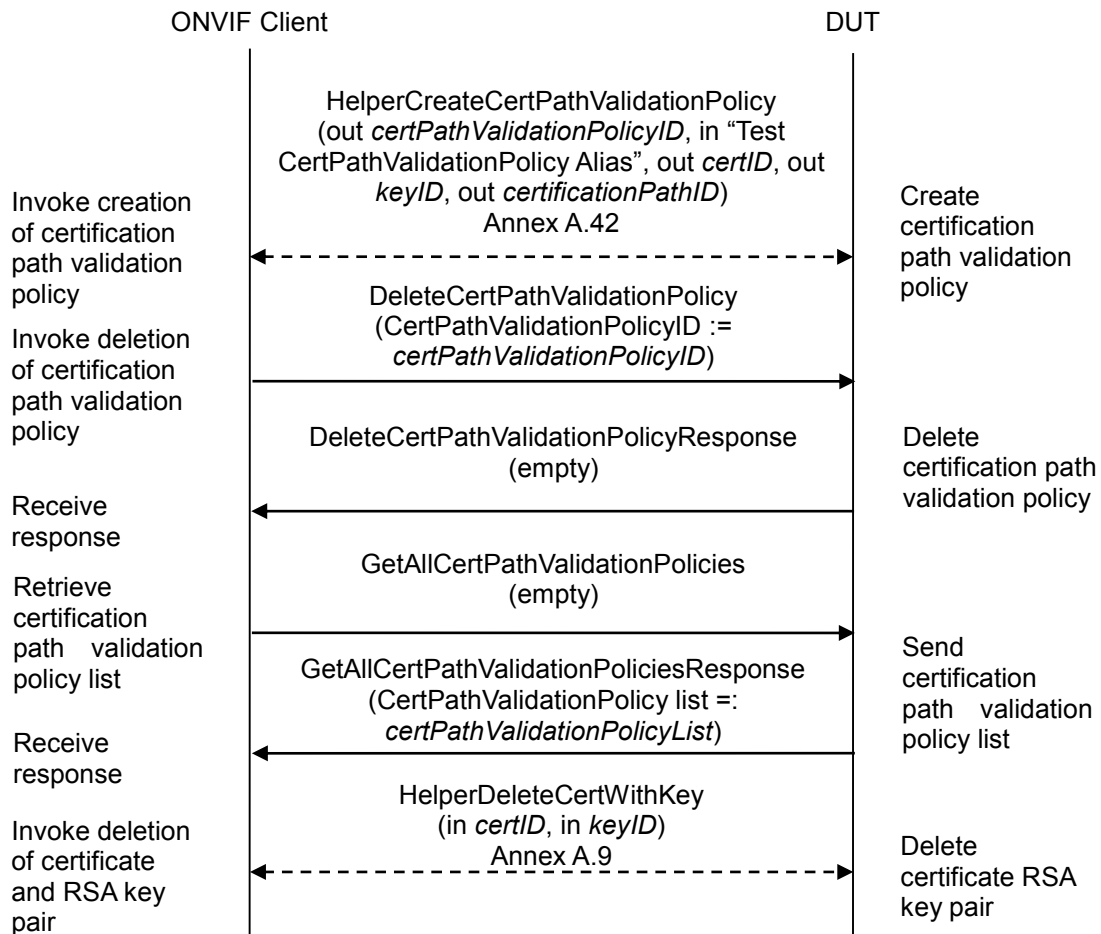
**WSDL Reference:** advancedsecurity.wsdl

**Test Purpose:** Verify that a certification path validation policy can be deleted from DUT.

**Pre-requisite:** Advanced Security Service is received from the DUT. Certification path validation policy supported by the DUT as indicated by the MaximumNumberOfCertificationPathValidationPolicies capability. The DUT shall have enough free storage capacity for one additional certification path validation policy. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**



**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates certification path validation policy identifier (out *certPathValidationPolicyID*) with specified alias (in "Test CertPathValidationPolicy Alias"), related certificate (out *certID*), RSA key pair (out *keyID*) and certification path if any (out *certificationPathID*) by following the procedure mentioned in Annex A.42.
4. ONVIF Client invokes **DeleteCertPathValidationPolicy** with parameters
  - *CertPathValidationPolicyID := certPathValidationPolicyID*
5. DUT responds with a **DeleteCertPathValidationPolicyResponse** message.
6. ONVIF Client invokes **GetAllCertPathValidationPolicies**.
7. The DUT responds with a **GetAllCertPathValidationPoliciesResponse** message with parameters



- CertPathValidationPolicy list =: *certPathValidationPolicyList*
8. If *certPathValidationPolicyList* contains *certPathValidationPolicyID*, FAIL the test, and go to the step 9.
  9. If *certificationPathID* is null:
    - 9.1. ONVIF Client deletes the self-signed certificate (in *certID*) and related the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.9.
    - 9.2. Skip other steps.
  10. If *certificationPathID* is not null:
    - 10.1. ONVIF Client deletes the certification path (in *certificationPathID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.35 to restore DUT configuration.

**Test Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **DeleteCertPathValidationPolicyResponse** message.

The DUT did not send **GetAllCertPathValidationPoliciesResponse** message.



## Annex A

This section describes the meaning of the following definitions. These definitions are used in the test case description.

### A.1 Delete an RSA key pair

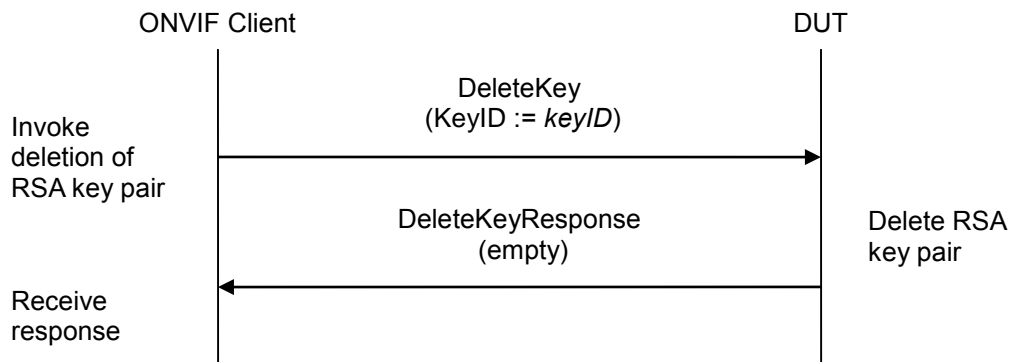
**Name:** HelperDeleteRSAKeyPair

**Procedure Purpose:** Helper procedure to delete an RSA key pair.

**Pre-requisite:** Advanced Security Service is received from the DUT. On-board RSA key pair generation is supported by the DUT as indicated by the RSAKeyPairGeneration capability.

**Input:** The identifier of the key pair (*keyID*) to delete.

**Returns:** None



#### Procedure:

1. ONVIF Client invokes **DeleteKey** with parameters
  - KeyID := *keyID*
2. DUT responds with a **DeleteKeyResponse** message.

#### Procedure Result:

##### PASS –

The DUT passed all assertions.

##### FAIL –

The DUT did not send **DeleteKeyResponse** message.





## **A.2 Subject for a server certificate**

Use the following subject for test cases:

- Subject.Country := "US"
- Subject.CommonName := <DUT IP-address>

## **A.3 Creating a certificate from a PKCS#10 request**

**Name:** HelperCreateCertificateFromPKCS10CSR

**Procedure Purpose:** Helper procedure to create an X.509 certificate from a PKCS#10 certification request.

**Pre-requisite:** None.

**Input:** PKCS#10 request (*pkcs10*) and associated CA certificate (*CAcert*) and a corresponding private key (*privateKey*).

**Returns:** An [RFC5280] compliant X.509 certificate (*certResult*) from the PKCS#10 request signed with the public key in the CA certificate.

**Procedure:**

1. Create an [RFC5280] compliant X.509 certificate (*certResult*) from the PKCS#10 request (*pkcs10*) with the following properties:
  - version:= v3
  - signature := sha1-WithRSAEncryption
  - subject := subject from the PKCS#10 request (*pkcs10*)
  - subject public key := subject public key in the PKCS#10 request (*pkcs10*)
  - validity := not before 19700101000000Z and not after 99991231235959Z
  - certificate signature is generated with the private key (*privateKey*) in the CA certificate (*CAcert*)
  - certificate extensions := the X.509v3 extensions from the PKCS#10 request (*pkcs10*)



#### **A.4 Provide CA certificate**

**Name:** HelperCreateCACertificate

**Procedure Purpose:** Helper procedure to create an X.509 CA certificate.

**Pre-requisite:** None.

**Input:** The subject (*subject*) of certificate (optional input parameter, could be skipped).

**Returns:** An X.509 CA certificate (*CAcert*) that is compliant to [RFC5280] and a corresponding private key (*privateKey*) and public key (*publicKey*).

**Procedure:**

1. If *subject* is skipped set:
  - *subject* := "CN=ONVIF TT,C=US"
2. ONVIF Client creates an X.509 self-signed CA certificate that is compliant to [RFC5280] and has the following properties:
  - version:= v3
  - signature := sha1-WithRSAEncryption
  - validity := not before 19700101000000Z and not after 99991231235959Z
  - subject := *subject*

**Note:** ONVIF Client may return the same CA certificate in subsequent invocations of this procedure for the same subject.



**A.5 Delete a certification path with corresponding certificate and RSA key pair**

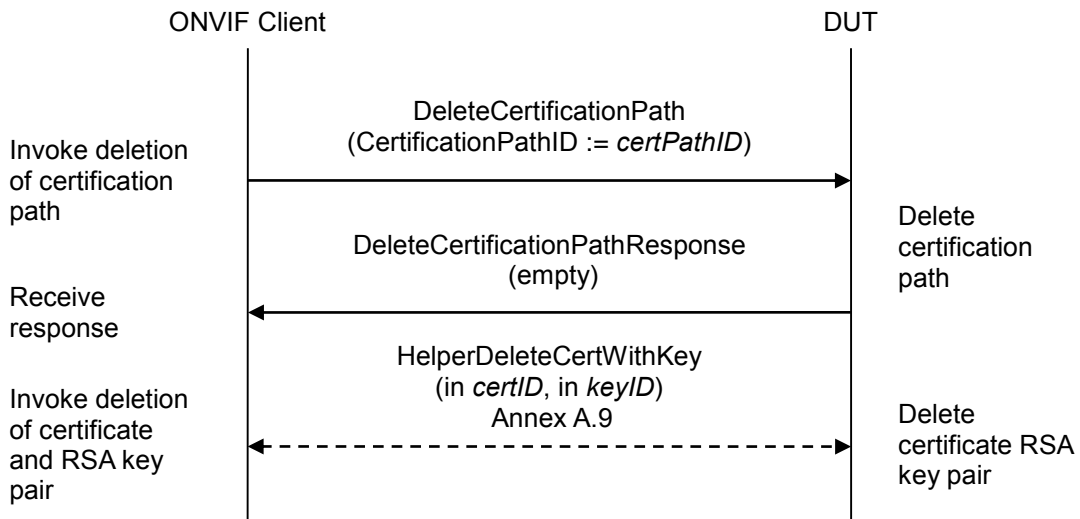
**Name:** HelperDeleteCertificationPath

**Procedure Purpose:** Helper procedure to delete certification path and related certificate and RSA key pair.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate by the DUT as indicated by the SelfSignedCertificateCreationWithRSA or PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSACapability. TLS is supported by the DUT as indicated by the TLSServerSupported capability.

**Input:** The identifier of the certification path (*certPathID*), certificate (*certID*) and RSA key pair (*keyID*) to delete.

**Returns:** None



**Procedure:**

1. ONVIF Client invokes **DeleteCertificationPath** with parameters
  - CertificationPathID := *certPathID*
2. DUT responds with a DeleteCertificationPathResponse message.
3. ONVIF Client deletes the self-signed certificate (in *certID*) and related the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.9.

**Procedure Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**



The DUT did not send **DeleteCertificationPathResponse** message.



## A.6 Determine RSA key length

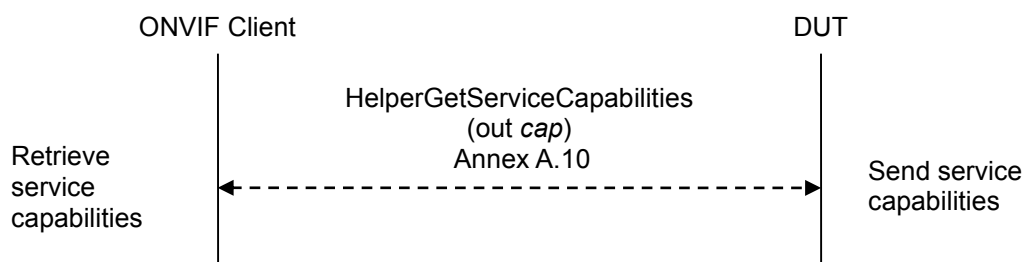
**Name:** HelperDetermineRSAKeyLength

**Procedure Purpose:** Helper procedure to determine the RSA key length to use during testing.

**Pre-requisite:** Advanced Security Service is received from the DUT. On-board RSA key pair generation is supported by the DUT as indicated by the RSAKeyPairGeneration capability.

**Input:** None

**Returns:** The smallest supported RSA key length (*keyLength*).



### Procedure:

1. ONVIF Client gets the service capabilities (out *cap*) by the following the procedure mentioned in Annex A.10.
2. ONVIF Client loops through the supported Key length list (*cap.RSAKeyLengths*) and selects the smallest supported key length (*keyLength*).

### Procedure Result:

#### PASS –

The DUT passed all assertions.

#### FAIL –

No supported key length was found at step 2.



### A.7 Create an RSA key pair

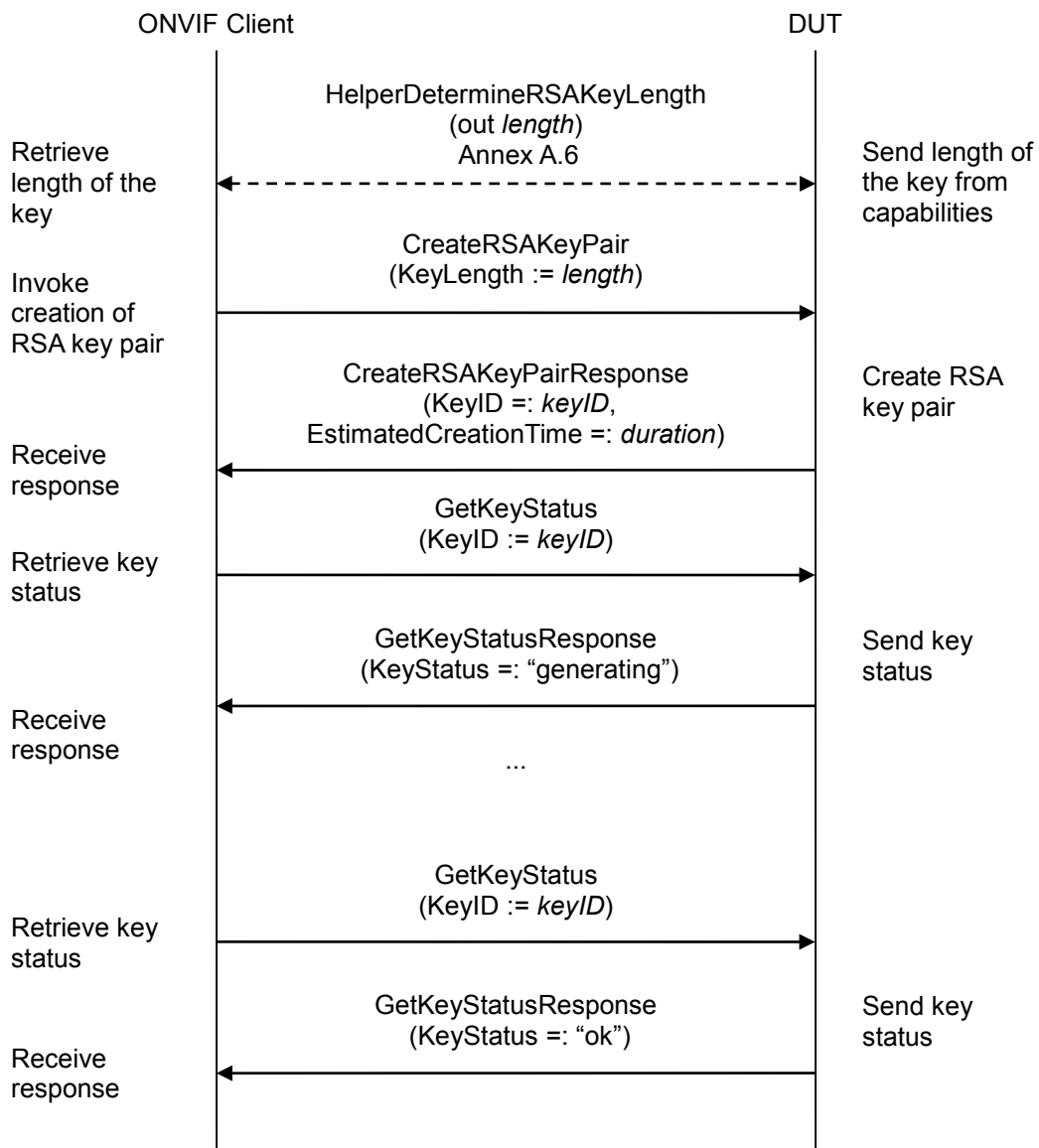
**Name:** HelperCreateRSAKeyPair

**Procedure Purpose:** Helper procedure to create an RSA key pair

**Pre-requisite:** Advanced Security Service is received from the DUT. On-board RSA key pair generation is supported by the DUT as indicated by the RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional RSA key pair.

**Input:** None

**Returns:** The identifier of the new key pair (*keyID*).





**Procedure:**

1. ONVIF Client determines the length of the key to generate (out *length*) by following the procedure mentioned in Annex A.6.
2. ONVIF Client invokes **CreateRSAKeyPair** with parameter
  - KeyLength := *length*
3. The DUT responds with **CreateRSAKeyPairResponse** message with parameters
  - KeyID =: *keyID*
  - EstimatedCreationTime =: *duration*
4. Until *timeout1* expires repeat the following steps:
  - 4.1. ONVIF Client waits for time *duration*.
  - 4.2. ONVIF Client invokes **GetKeyStatus** with parameters
    - KeyID := *keyID*
  - 4.3. The DUT responds with **GetKeyStatusResponse** message with parameters
    - KeyStatus =: *keyStatus*
  - 4.4. If *keyStatus* is equal to "ok", *keyID* will be return as a result of the procedure, other steps will be skipped.
  - 4.5. If *keyStatus* is equal to "corrupt", FAIL the procedure and deletes the RSA key pair (*keyID*) by following the procedure mentioned in Annex A.1.
5. If *timeout1* expires for step 4 and the last *keyStatus* is other than "ok", FAIL the procedure and deletes the RSA key pair (*keyID*) by following the procedure mentioned in Annex A.1.

**Procedure Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **CreateRSAKeyPairResponse** message.

The DUT did not send **GetKeyStatusResponse** message(s).

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.



### A.8 Create a self-signed certificate

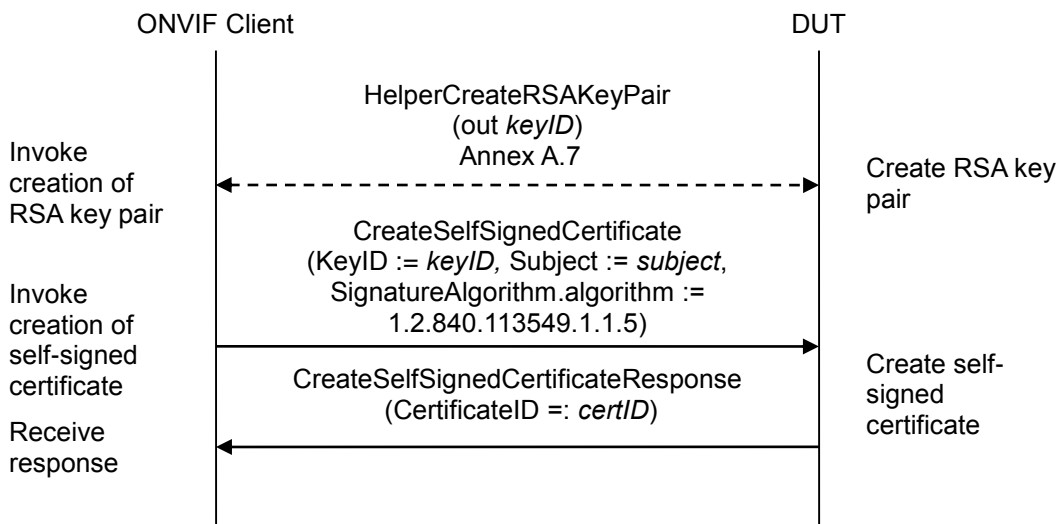
**Name:** HelperCreateSelfSignedCertificate

**Procedure Purpose:** Helper procedure to create a self-signed certificate.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate.

**Input:** None

**Returns:** The identifier of the new certificate (*certID*) and RSA key pair (*keyID*).



**Procedure:**

1. ONVIF Client creates an RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.7.
2. ONVIF Client invokes **CreateSelfSignedCertificate** with parameters
  - X509Version skipped
  - KeyID := *keyID*
  - Subject := *subject* (see Annex A.2)
  - Alias skipped
  - notValidBefore skipped
  - notValidAfter skipped
  - SignatureAlgorithm.algorithm := 1.2.840.113549.1.1.5 (OID of SHA-1 with RSA Encryption)





algorithm)

- SignatureAlgorithm.parameters skipped
- SignatureAlgorithm.anyParameters skipped
- Extension skipped

3. The DUT responds with a **CreateSelfSignedCertificateResponse** message with parameters

- CertificateID =: *certID*

**Procedure Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **CreateSelfSignedCertificateResponse** message.



### A.9 Delete a certificate with corresponding RSA key pair

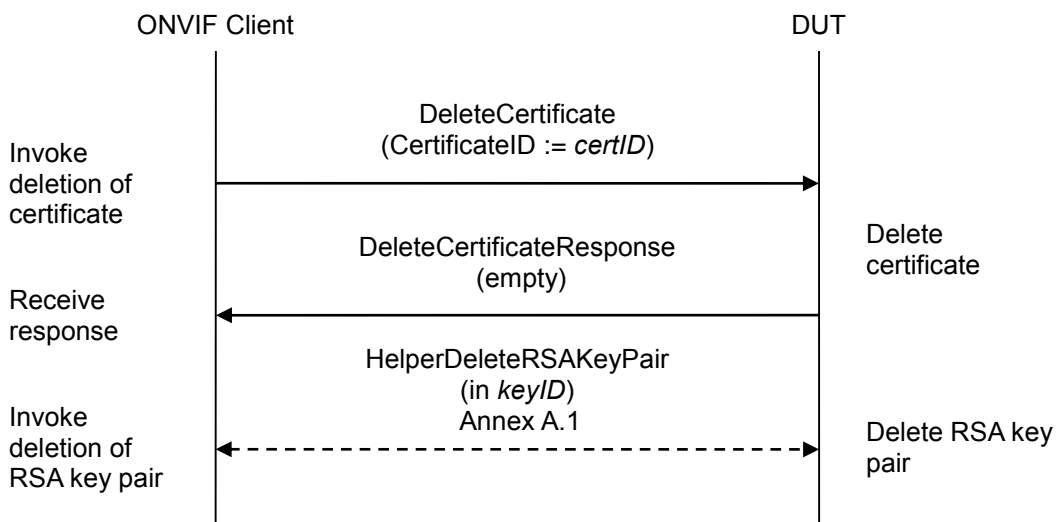
**Name:** HelperDeleteCertWithKey

**Procedure Purpose:** Helper procedure to delete a certificate and related RSA key pair.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate by the DUT as indicated by the SelfSignedCertificateCreationWithRSA or PCKS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSACapability.

**Input:** The identifier of the certificate (*certID*) and RSA key pair (*keyID*) to delete.

**Returns:** None



**Procedure:**

1. ONVIF Client invokes **DeleteCertificate** with parameters
  - CertificateID := *certID*
2. The DUT responds with a **DeleteCertificateResponse** message.
3. ONVIF Client deletes the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.1.

**Procedure Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **DeleteCertificateResponse** message



### A.10 Get service capabilities

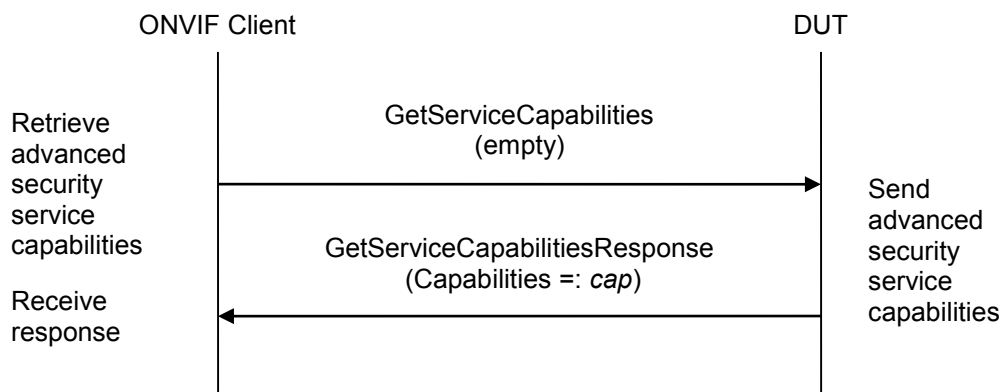
**Name:** HelperGetServiceCapabilities

**Procedure Purpose:** Helper procedure to get service capabilities.

**Pre-requisite:** Advanced Security Service is received from the DUT.

**Input:** None

**Returns:** The service capabilities (*cap*).



**Procedure:**

1. ONVIF Client invokes **GetServiceCapabilities**.
2. The DUT responds with a **GetServiceCapabilitiesResponse** message with parameters
  - Capabilities =: *cap*

**Procedure Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetServiceCapabilitiesResponse** message.



### A.11 Create a certification path based on self-signed certificate

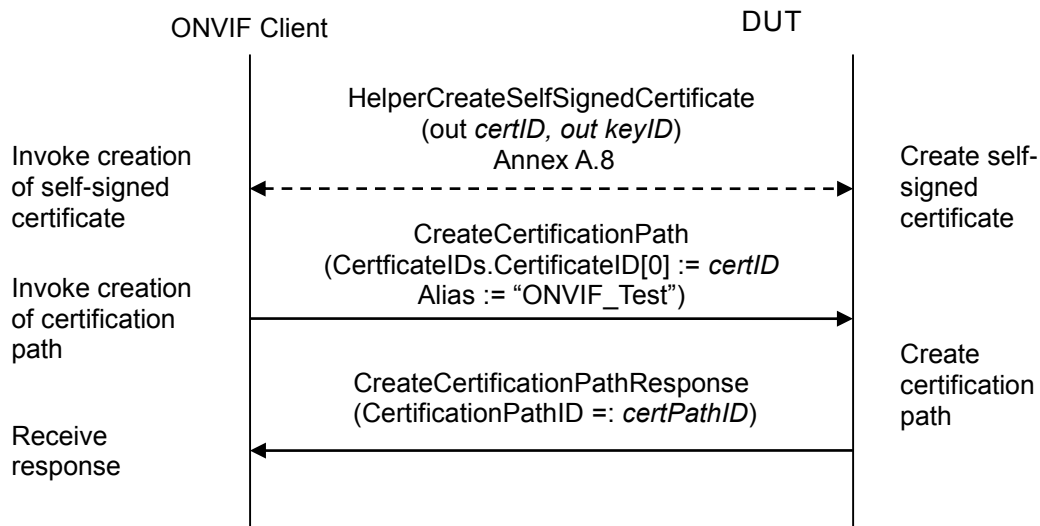
**Name:** HelperCreateCertificationPath\_SelfSigned

**Procedure Purpose:** Helper procedure to create a certification path based on self-signed certificate.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. The DUT shall have enough free storage capacity for one additional certification path.

**Input:** None

**Returns:** The identifier of the new certification path (*certPathID*), certificate (*certID*) and RSA key pair (*keyID*).



**Procedure:**

1. ONVIF Client creates a self-signed certificate (out *certID*) and related RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.8.
2. ONVIF Client invokes **CreateCertificationPath** with parameters
  - CertificateIDs.CertificateID[0] := *certID*
  - Alias := "ONVIF\_Test"
3. The DUT responds with a **CreateCertificationPathResponse** message with parameters
  - CertificationPathID := *certPathID*

**Procedure Result:**



**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **CreateCertificationPathResponse** message.



**A.12 Remove server certificate assignment with corresponding certification path, certificate and RSA key pair**

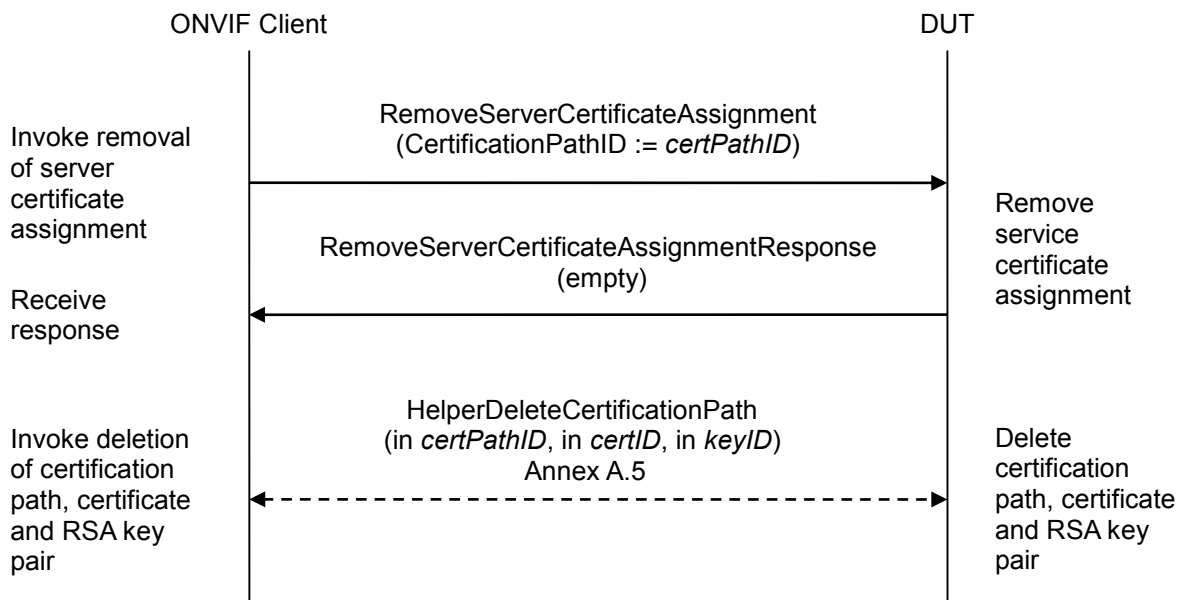
**Name:** HelperRemoveServerCertificateAssignment

**Procedure Purpose:** Helper procedure to remove server certificate assignment with corresponding certification path, certificate and RSA key pair.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate by the DUT as indicated by the SelfSignedCertificateCreationWithRSA or PCKS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSACapability. TLS supported by the DUT as indicated by the TLSServerSupported capability.

**Input:** The identifier of certification path (*certPathID*), certificate (*certID*) and RSA key pair (*keyID*) to delete.

**Returns:** None



**Procedure:**

1. ONVIF Client invokes **RemoveServerCertificateAssignment** with parameters
  - CertificationPathID := *certPathID*
2. The DUT responds with a **RemoveServerCertificateAssignmentResponse** message.
3. ONVIF Client deletes the certification path (in *certPathID*) and related certificate (in *certID*) and the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.5.

**Procedure Result:**

**PASS –**



The DUT passed all assertions.

**FAIL –**

The DUT did not send **RemoveServerCertificateAssignmentResponse** message.



**A.13 Add server certificate assignment with corresponding certification path, self-signed certificate and RSA key pair**

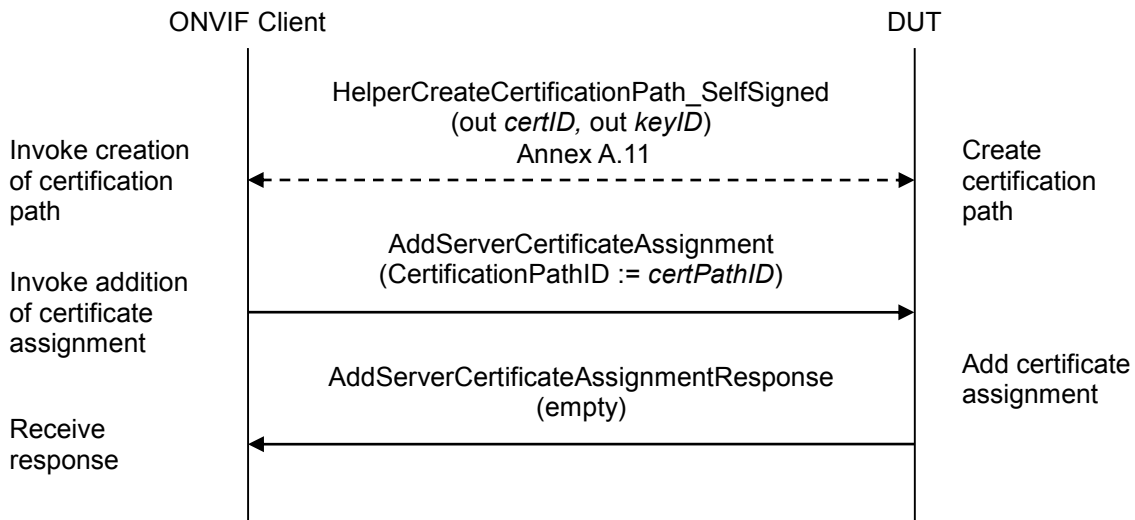
**Name:** HelperAddServerCertAssign\_SSCertificate

**Procedure Purpose:** Helper procedure to add server certificate assignment with corresponding certification path, self-signed certificate and RSA key pair.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. The DUT shall have enough free storage capacity for one additional certification path.

**Input:** None

**Returns:** The identifiers of the new certification path (*certPathID*), certificate (*certID*) and RSA key pair (*keyID*).



**Procedure:**

1. ONVIF Client creates a certification path (out *certPathID*) based on self-signed certificate (out *certID*) and related RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.11.
2. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
  - CertificationPathID := *certPathID*
3. The DUT responds with an **AddServerCertificateAssignmentResponse** message.

**Procedure Result:**

**PASS –**





The DUT passes all assertions.

**FAIL –**

The DUT did not send **AddServerCertificateAssignmentResponse** message.



### A.14 Create a CA-signed certificate for RSA key pair

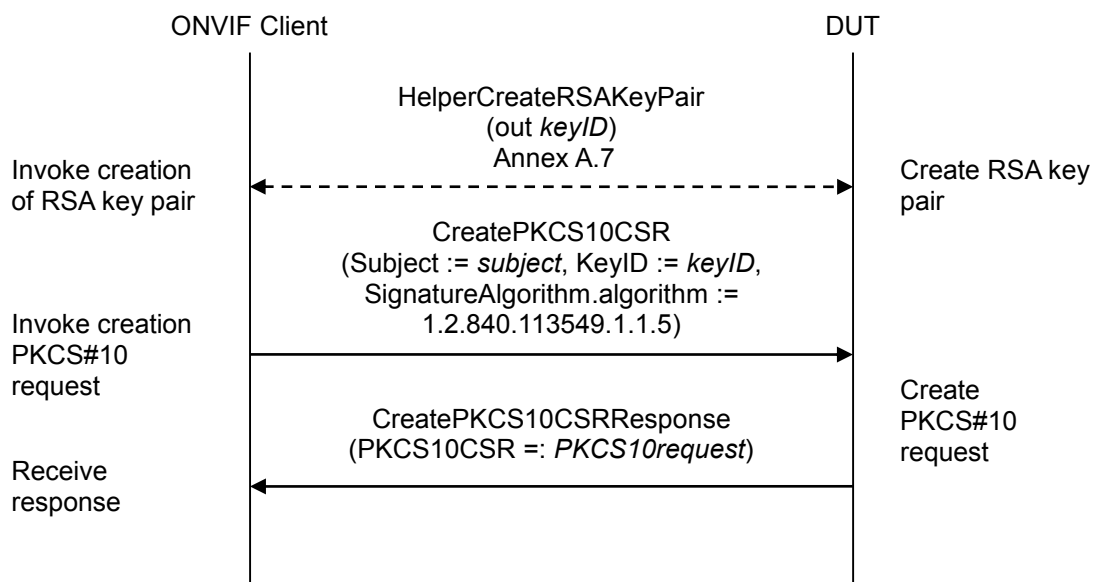
**Name:** HelperCreateCASignedCertificate

**Procedure Purpose:** Helper procedure to create a CA-signed certificate for RSA key pair.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional RSA key pair. Current time of the DUT shall be at least Jan 01, 1970.

**Input:** CA certificate (*CAcert*) and a corresponding private key (*privateKey*).

**Returns:** The identifier of the new key pair (*keyID*), a CA-signed certificate (*cert*).



**Procedure:**

1. ONVIF Client creates an RSA key pair (*keyID*) by following the procedure mentioned in Annex A.7.
2. ONVIF Client invokes **CreatePKCS10CSR** with parameter
  - Subject := *subject* (see Annex A.2)
  - KeyID := *keyID*
  - CSRAttribute skipped
  - SignatureAlgorithm.algorithm := 1.2.840.113549.1.1.5 (OID of SHA-1 with RSA Encryption algorithm)
3. The DUT responds with **CreatePKCS10CSRResponse** message with parameters



- PKCS10CSR =: *PKCS10request*

4. ONVIF Client creates a certificate (out *cert*) from the PKCS#10 request (in *PKCS10request*) and an associated CA certificate (in *CAcert*) with related private key (in *privateKey*) by following the procedure described in Annex A.3.

**Procedure Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **CreatePKCS10CSRResponse** message.



### A.15 Upload a certificate without Private Key Assignment

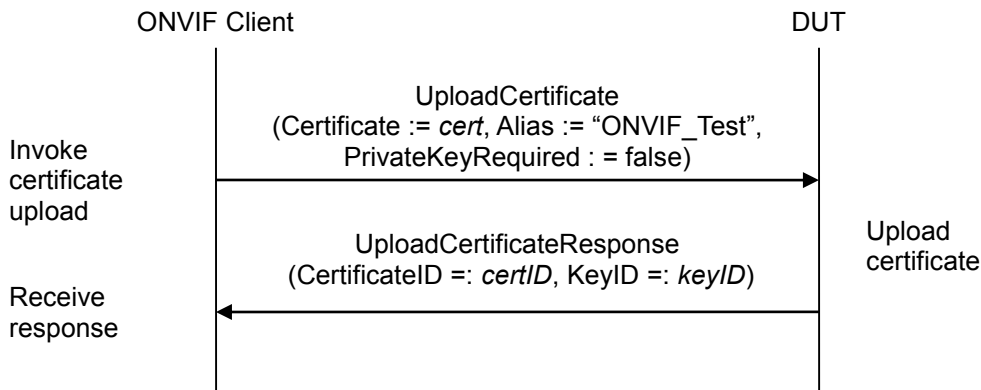
**Name:** HelperUploadCertificate

**Procedure Purpose:** Helper procedure to upload a certificate without private key assignment.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate.

**Input:** Certificate (*cert*).

**Returns:** The identifier of the new RSA key pair (*keyID*) and a certificate identifier (*certID*).



**Procedure:**

1. ONVIF Client invokes **UploadCertificate** with parameters
  - Certificate := *cert*
  - Alias := "ONVIF\_Test"
  - PrivateKeyRequired := false
2. The DUT responds with a **UploadCertificateResponse** message with parameters
  - CertificateID := *certID*
  - KeyID := *keyID*

**Procedure Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **UploadCertificateResponse** message.



**A.16 Create and upload a CA-signed certificate for private key**

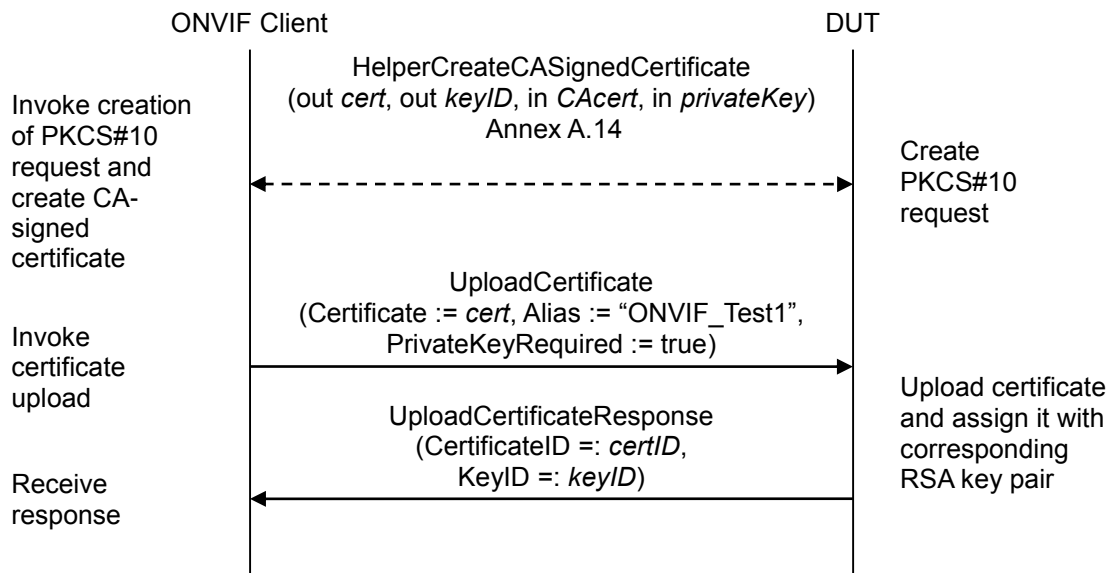
**Name:** HelperUploadCASignedCertificate

**Procedure Purpose:** Helper procedure to create and upload a CA-signed certificate for private key

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. Current time of the DUT shall be at least Jan 01, 1970.

**Input:** CA certificate (*CAcert*) and a corresponding private key (*privateKey*).

**Returns:** The identifier of the new RSA key pair (*keyID*), a certificate identifier (*certID*).



**Procedure:**

1. ONVIF Client creates a certificate (out *cert*) from the PKCS#10 request with RSA key pair (out *keyID*) and associated CA certificate (in *CAcert*) and a corresponding private key (in *privateKey*) by following the procedure described in Annex A.14.
2. ONVIF Client invokes **UploadCertificate** with parameters
  - Certificate := *cert*
  - Alias := "ONVIF\_Test1"
  - PrivateKeyRequired := true
3. The DUT responds with a **UploadCertificateResponse** message with parameters
  - CertificateID := *certID*



- KeyID =: *keyID*

**Procedure Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **UploadCertificateResponse** message.



**A.17 Delete a certification path with corresponding two certificates and RSA key pairs**

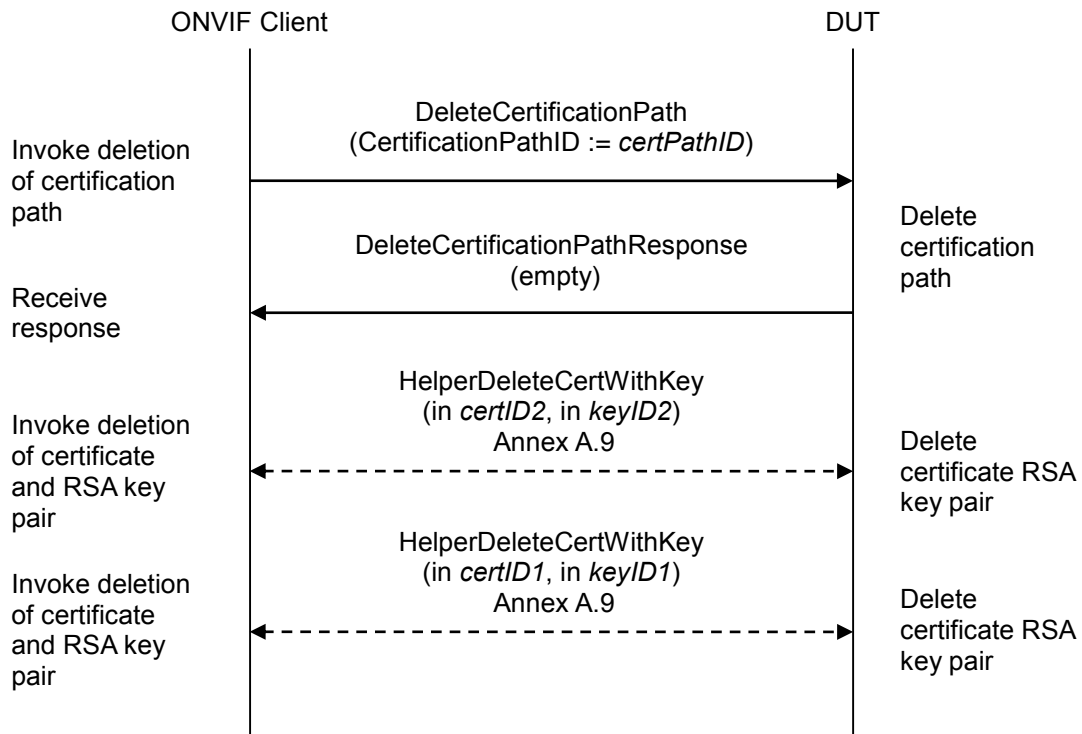
**Name:** HelperDeleteCertificationPath2

**Procedure Purpose:** Helper procedure to delete certification path and related certificates and RSA key pairs.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate by the DUT as indicated by the SelfSignedCertificateCreationWithRSA or PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSACapability. TLS is supported by the DUT as indicated by the TLSServerSupported capability.

**Input:** The identifier of the certification path (*certPathID*), certificate (*certID1*) and RSA key pair (*keyID1*), certificate (*certID2*) and RSA key pair (*keyID2*) to delete.

**Returns:** None



**Procedure:**

1. ONVIF Client invokes **DeleteCertificationPath** with parameters
  - CertificationPathID := *certPathID*
2. The DUT responds with a **DeleteCertificationPathResponse** message.
3. ONVIF Client deletes the CA certificate (*certID2*) and related RSA key pair (*keyID2*) by following the procedure mentioned in Annex A.9.



4. ONVIF Client deletes the CA certificate (*certID1*) and related RSA key pair (*keyID1*) by following the procedure mentioned in Annex A.9.

**Procedure Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **DeleteCertificationPathResponse** message.





**A.18 Create certification path with CA-signed certificate and associated CA certificate**

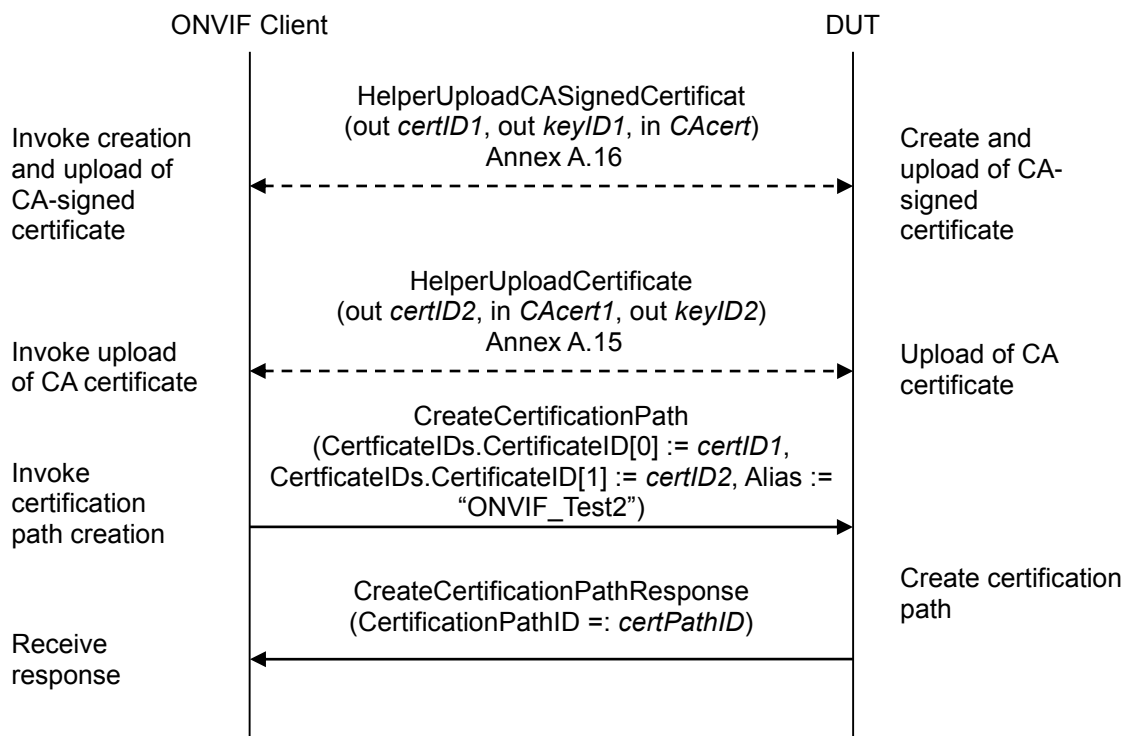
**Name:** HelperCreateCertificationPath\_CACertificates

**Procedure Purpose:** Helper procedure to create a certification path based on CA-signed certificate and associated CA certificate.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability. RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability. The DUT shall have enough free storage capacity for two additional RSA key pairs. The DUT shall have enough free storage capacity for two additional certificates. The DUT shall have enough free storage capacity for one additional certification path. Current time of the DUT shall be at least Jan 01, 1970.

**Input:** None

**Returns:** The identifier of the new certification path (*certPathID*) and two related certificates: CA-signed certificate (*certID1*) and related key (*keyID1*) and associated CA certificate (*certID2*) and related key (*keyID2*).



**Procedure:**

1. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding private key (out *privateKey*) by following the procedure described in Annex A.4.
2. ONVIF Client creates and uploads a CA-signed certificate (out *certID1*) for RSA key pair (out *keyID1*) and associated CA certificate (out *certID2*) and related key (*keyID2*).



*keyID1*) and associated CA certificate (in *CAcert*) and a corresponding private key (in *privateKey*) by following the procedure described in Annex A.16.

3. ONVIF Client uploads a CA certificate (out *certID2*, in *CAcert*) and new RSA key pair with the public key from the CA certificate (out *keyID2*) by following the procedure described in Annex A.15.
4. ONVIF Client invokes **CreateCertificationPath** with parameters
  - `CertificateIDs.CertificateID[0] := certID1`
  - `CertificateIDs.CertificateID[1] := certID2`
  - `Alias := "ONVIF_Test2"`
5. The DUT responds with a **CreateCertificationPathResponse** message with parameters
  - `CertificationPathID := certPathID`

**Procedure Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **CreateCertificationPathResponse** message.



### **A.19 Validate DER encoding**

**Name:** HelperValidateDEREncoding

**Procedure Purpose:** Helper procedure to validate DER encoding.

**Pre-requisite:** None.

**Input:** DER encoded data (*dataDER*).

**Returns:** None.

**Procedure:**

1. ONVIF Client tries to decode DER encoded data *dataDER*. If decoding was failed, then *dataDER* is not valid encoded, FAIL the procedure and skip other steps.
2. ONVIF Client DER encodes the result from previous step (*dataDER2*).
3. ONVIF Client compares *dataDER* and *dataDER2*. If they are not equal, then *dataDER* is not valid encoded, FAIL the procedure.

**Procedure Result:**

**PASS –**

The DUT passed all assertions.



**A.20 Remove server certificate assignment with corresponding certification path, certificates and RSA key pairs**

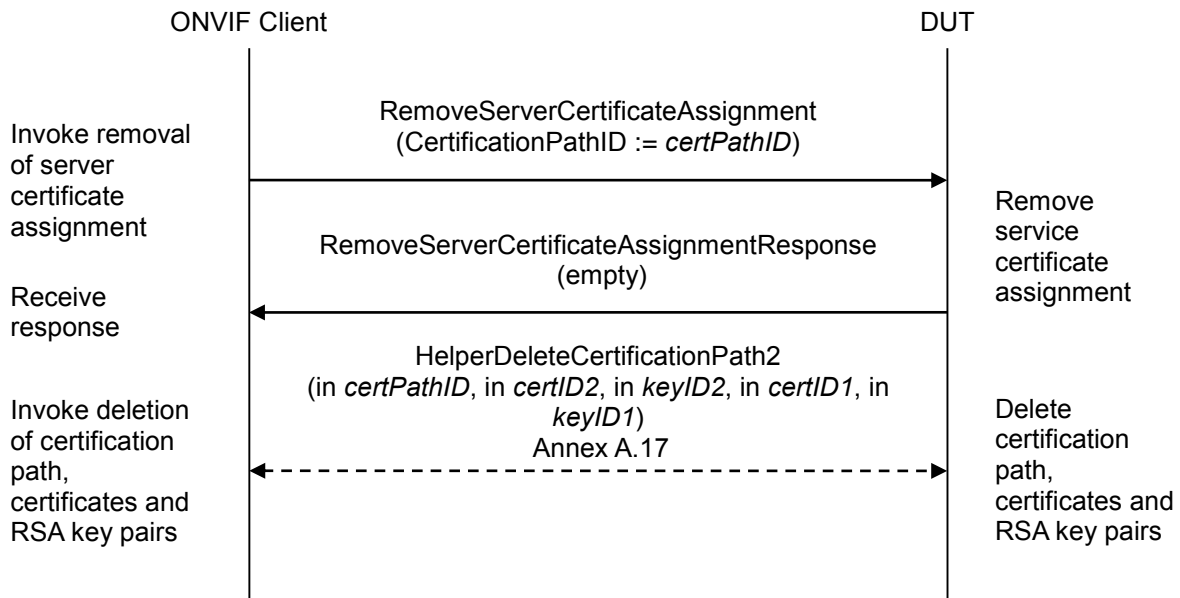
**Name:** HelperRemoveServerCertificateAssignment2

**Procedure Purpose:** Helper procedure to remove server certificate assignment with corresponding certification path, certificates and RSA key pairs.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate by the DUT as indicated by the SelfSignedCertificateCreationWithRSA or PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSACapability. TLS is supported by the DUT as indicated by the TLSServerSupported capability.

**Input:** The identifier of certification path (*certPathID*), certificate (*certID1*) and RSA key pair (*keyID1*), certificate (*certID2*) and RSA key pair (*keyID2*) to delete.

**Returns:** None



**Procedure:**

1. ONVIF Client invokes **RemoveServerCertificateAssignment** with parameters
  - CertificationPathID := *certPathID*
2. DUT responds with a RemoveServerCertificateAssignmentResponse message.
3. ONVIF Client deletes the certification path (in *certPathID*), related the CA certificate (in *certID2*) and the RSA key pair (in *keyID2*) and related the CA-signed certificate (in *certID1*) and the RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.17.

**Procedure Result:**

**PASS –**



The DUT passed all assertions.

**FAIL –**

The DUT did not send **RemoveServerCertificateAssignmentResponse** message.



## A.21 Basic TLS handshake

**Name:** HelperBasicTLShandshakeCheck

**Procedure Purpose:** Helper procedure to verify basic TLS handshake.

**Pre-requisite:** TLS is supported by the DUT as indicated by the TLSServerSupported capability. TLS is configured. HTTPS protocol is enabled.

**Input:** TLS server certification path ID (certPathID).

### Procedure:

1. ONVIF Client invokes **ClientHello** with parameters
  - ClientVersion := 3,1
  - Random number := *ClientRandom[32]*, that is 4-byte number that consists of the client's date and time plus a 28-byte randomly generated number
  - CipherSuites := list of common CipherSuites used by TLS 1.0, SSL 2.0 and 3.0
  - Compression methods list := NONE
  - <SessionID> skipped
2. The DUT TLS server responds with a **ServerHello** message with parameters
  - Version =: the highest version number supported by both sides
  - Random number =: *ServerRandom[32]*, that is 4-byte number that consists of the client's date and time plus a 28-byte randomly generated number
  - CipherSuite =: the strongest cipher that both the client and server support
  - Compression method =: NONE
  - Session ID =: *SessionID*
3. The DUT TLS server responds **Certificate** message with parameters
  - Certificate.CertificateID =: *CertificateID*
  - Certificate.KeyID =: *KeyID*
4. The DUT TLS server responds a **ServerHelloDone** message.
5. ONVIF Client invokes **ClientKeyExchange** message with parameter
  - Premaster Secret := *PreMasterSecret* encrypted with *KeyID*
6. ONVIF Client computes *MasterSecret* using *ClientRandom[32]*, *ServerRandom[32]* and *PreMasterSecret*.
7. The DUT TLS server computes *MasterSecret* using *ClientRandom[32]*, *ServerRandom[32]* and *PreMasterSecret*.
8. ONVIF Client invokes **ChangeCipherSpec** message.



9. ONVIF Client invokes encrypted **Finished** message, containing a hash := *hash1* and MAC := *MAC1* over the previous handshake messages.
10. The DUT TLS server decrypts the client's *Finished* message and verify the hash and MAC.
11. The DUT TLS server responds **ChangeCipherSpec**.
12. The DUT TLS server responds its encrypted **Finished** message, containing a hash =: *hash2* and MAC =: *MAC2* over the previous handshake messages.
13. If *hash1* is not equal to *hash2*, FAIL the test.
14. If *MAC1* is not equal to *MAC2*, FAIL the test.

**Procedure Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT TLS server did not send **ServerHello** message.

The DUT TLS server did not send **Certificate** message.

The DUT TLS server did not send **ServerHelloDone** message.

The DUT TLS server did not send **ChangeCipherSpec** message.

The DUT TLS server did not send **Finished** message.

The DUT TLS server sends Alert Message.



## **A.22 Provide expired CA certificate**

**Name:** HelperCreateExpiredCACertificate

**Procedure Purpose:** Helper procedure to create an expired X.509 CA certificate.

**Pre-requisite:** None.

**Input:** None

**Returns:** An X.509 CA certificate (*CAcert*) that is compliant to [RFC5280] and a corresponding private key (*privateKey*).

### **Procedure:**

1. ONVIF Client invokes **GetSystemDateAndTimeRequest**.
2. The DUT responds with a **GetSystemDateAndTimeResponse** message with parameters
  - `UTCDateTime` =: *DUTCurrentTime*
3. ONVIF Client creates an X.509 self-signed CA certificate that is compliant to [RFC5280] and has the following properties:
  - `version` := v3
  - `signature` := sha1-WithRSAEncryption
  - `validity` := not before 19700101000000Z and not after [*DUTCurrentTime* - 1 day]

**Note:** ONVIF Client may return the same CA certificate in subsequent invocations of this procedure.





### A.23 Delete a passphrase

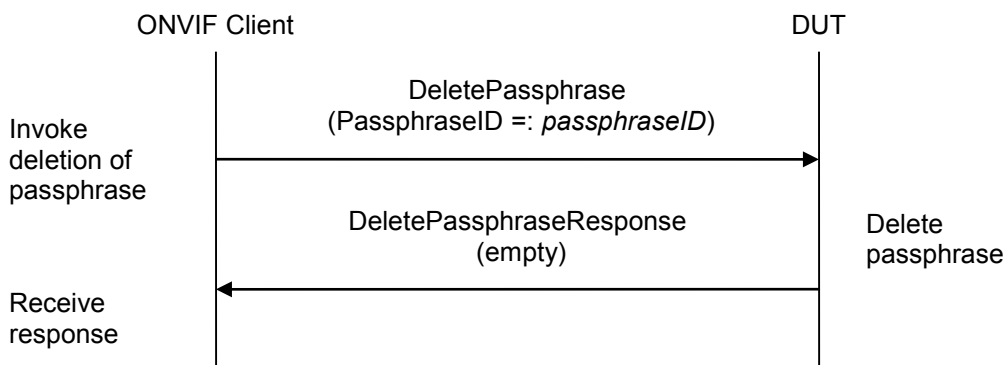
**Name:** HelperDeletePassphrase

**Procedure Purpose:** Helper procedure to delete a passphrase.

**Pre-requisite:** Advanced Security Service is received from the DUT. Passphrase handling is supported by the DUT as indicated by the `MaximumNumberOfPassphrases > 0` capability.

**Input:** The identifier of the passphrase (*passphraseID*) to delete.

**Returns:** None



**Procedure:**

1. ONVIF Client invokes **DeletePassphrase** with parameters
  - `PassphraseID := passphraseID`
2. The DUT responds with a **DeletePassphraseResponse** message.

**Procedure Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **DeletePassphraseResponse** message.



#### **A.24 Passphrases for test cases**

Use the following passphrases for test cases (20 ASCII characters):

- *passphrase1* := "Passphrase for ONVIF"
- *passphrase2* := "AdditionalPassphrase"

#### **A.25 Creating a PKCS#8 data structure with new public key and private key without passphrase**

**Name:** HelperCreatePKCS8WithNewKeyPair

**Procedure Purpose:** Helper procedure to create a PKCS#8 data structure with new public key and private key without passphrase.

**Pre-requisite:** None.

**Input:** None.

**Returns:** A [RFC 5958, RFC 5959] compliant PKCS#8 data structure (*keyPairInPKCS8*) with new public key (*publicKey*) and private key (*privateKey*).

**Procedure:**

1. ONVIF Client generates RSA key pair with public key (out *publicKey*) and private key (out *privateKey*) by following the procedure mentioned in Annex A.26.
2. ONVIF Client generates a PKCS#8 data structure (out *keyPairInPKCS8*) with existing public key (in *publicKey*) and private key (in *privateKey*) by following the procedure mentioned in Annex A.27.



## **A.26 Generating an RSA key pair**

**Name:** HelperGenerateRSAKeyPair

**Procedure Purpose:** Helper procedure to generate an RSA key pair.

**Pre-requisite:** None.

**Input:** None.

**Returns:** A [RFC 3447] compliant RSA key pair with new public key (*publicKey*) and private key (*privateKey*).

### **Procedure:**

1. ONVIF Client determines the length of the key to generate (out *length*) by following the procedure mentioned in Annex A.6.
2. Create an [RFC 3447] compliant RSA key pair with new public key (out *publicKey*) and private key (out *privateKey*) with the following properties:
  - *KeyLength* := *length*



### **A.27 Creating a PKCS#8 data structure with existing public key and private key without passphrase**

**Name:** HelperCreatePKCS8WithExistingKeyPair

**Procedure Purpose:** Helper procedure to create a PKCS#8 data structure with existing public key and private key without passphrase.

**Pre-requisite:** None.

**Input:** A [RFC 3447] compliant RSA key pair with public key (*publicKey*) and private key (*privateKey*).

**Returns:** A [RFC 5958, RFC 5959] compliant PKCS#8 data structure (*keyPairInPKCS8*) for provided RSA key pair.

**Procedure:**

1. Create an [RFC 5958, RFC 5959] compliant PKCS#8 data structure (*keyPairInPKCS8*) with the following properties:
  - PrivateKeyInfo
    - version:= v2
    - privateKeyAlgorithm := rsaEncryption
    - privateKey := *privateKey*
    - attributes
      - publicKey := *publicKey*



### **A.28 Creating a PKCS#8 data structure with new public key and private key with passphrase**

**Name:** HelperCreatePKCS8WithNewKeyPairWithPassphrase

**Procedure Purpose:** Helper procedure to create a PKCS#8 data structure with new public key and private key with passphrase.

**Pre-requisite:** None.

**Input:** The passphrase (*passphrase*) to use in encryption.

**Returns:** A [RFC 5958, RFC 5959] compliant PKCS#8 data structure (*keyPairInPKCS8*) with new public key (*publicKey*) and private key (*privateKey*).

**Procedure:**

1. ONVIF Client generates RSA key pair with public key (out *publicKey*) and private key (out *privateKey*) by following the procedure mentioned in Annex A.26.
2. ONVIF Client generates a PKCS#8 data structure (out *keyPairInPKCS8*) with existing public key (in *publicKey*) and private key (in *privateKey*) with encryption passphrase (in *passphrase*) by following the procedure mentioned in Annex A.29.



### **A.29 Creating a PKCS#8 data structure with existing public key and private key with passphrase**

**Name:** HelperCreatePKCS8WithExistingKeyPairWithPassphrase

**Procedure Purpose:** Helper procedure to create a PKCS#8 data structure with existing public key and private key with passphrase.

**Pre-requisite:** None.

**Input:** A [RFC 3447] compliant RSA key pair with public key (*publicKey*) and private key (*privateKey*). The passphrase (*passphrase*) to use in encryption.

**Returns:** A [RFC 5958, RFC 5959] compliant PKCS#8 data structure (*keyPairInPKCS8*) for provided RSA key pair.

**Procedure:**

1. Use the current PrivateKeyInfo data:
  - PrivateKeyInfo
    - version := v2
    - privateKeyAlgorithm := rsaEncryption
    - privateKey := *privateKey*
    - attributes
      - publicKey := *publicKey*
2. Create an [RFC 5958, RFC 5959] compliant PKCS#8 data structure (*keyPairInPKCS8*) with the following properties:
  - EncryptedPrivateKeyInfo
    - encryptionAlgorithm := pbeWithSHAAnd3-KeyTripleDES-CBC
    - encryptedData := encrypted with *passphrase* PrivateKeyInfo data



### **A.30 Creating a PKCS#12 data structure with new CA-signed certificate signed by new public key and private key without passphrase**

**Name:** HelperCreatePKCS12WithNewCACert

**Procedure Purpose:** Helper procedure to create CA certificate and a corresponding public key in the certificate along with the corresponding private key in the form of a PKCS#12 file.

**Pre-requisite:** None.

**Input:** The subject (*subject*) of certificate (optional input parameter, could be skipped).

**Returns:** A [PKCS#12] compliant PKCS#12 data structure (*PKCS12data*) with CA certificate (*CACert*) and a corresponding public key (*publicKey*) in the certificate along with the corresponding private key (*privateKey*).

**Procedure:**

1. If *subject* is skipped, set:
  - *subject* := "CN=ONVIF TT,C=US"
2. ONVIF Client creates a CA certificate (out *CACert*) with subject (in *subject*) and a corresponding public key (out *publicKey*) in the certificate along with the corresponding private key (out *privateKey*) by following the procedure described in Annex A.4.
3. ONVIF Client creates a CA certificate (in *CACert*) and a corresponding public key (in *publicKey*) in the certificate along with the corresponding private key (in *privateKey*) in the form of a PKCS#12 file (out *PKCS12data*) by following the procedure described in Annex A.32.



### **A.31 Creating a PKCS#12 data structure with new CA-signed certificate signed by new public key and private key with passphrase**

**Name:** HelperCreatePKCS12WithNewCACertWithPassphrase

**Procedure Purpose:** Helper procedure to create CA certificate and a corresponding public key in the certificate along with the corresponding private key and encryption passphrase in the form of a PKCS#12 file.

**Pre-requisite:** None.

**Input:** The passphrase (*passphrase*) to use in encryption.

**Returns:** A [PKCS#12] compliant PKCS#12 data structure (*PKCS12data*) with CA certificate (*CACert*) and a corresponding public key (*publicKey*) in the certificate along with the corresponding private key (*privateKey*) encrypted with passphrase (*passphrase*).

**Procedure:**

1. ONVIF Client creates a CA certificate (out *CACert*) and a corresponding public key (out *publicKey*) in the certificate along with the corresponding private key (out *privateKey*) by following the procedure described in Annex A.4.
2. ONVIF Client creates a CA certificate (in *CACert*) and a corresponding public key (in *publicKey*) in the certificate along with the corresponding private key (in *privateKey*) encrypted with passphrase (in *passphrase*) in the form of a PKCS#12 file (out *PKCS12data*) by following the procedure described in Annex A.33.





### **A.32 Creating a PKCS#12 data structure with existing CA-signed certificate and a corresponding public key and private key without passphrase**

**Name:** HelperCreatePKCS12WithExistingCACert

**Procedure Purpose:** Helper procedure to create a PKCS#12 data structure with existing CA-signed certificate and a corresponding public key and private key without passphrase.

**Pre-requisite:** None.

**Input:** An X.509 CA certificate (*CACert*) that is compliant to [RFC5280] and a corresponding private key (*privateKey*) and public key (*publicKey*).

**Returns:** A [PKCS#12] compliant PKCS#12 data structure (*PKCS12data*).

**Procedure:**

1. Use the current PrivateKeyInfo data:
  - PrivateKeyInfo
    - version := v2
    - privateKeyAlgorithm := rsaEncryption
    - privateKey := *privateKey*
    - publicKey := *publicKey*
2. Create an [PKCS#12] compliant PKCS#12 data structure (*PKCS12data*) with the following properties:
  - version := v3
  - authSafe
    - SafeBag
      - Pkcs-12-KeyBag := PrivateKeyInfo
      - PKCS12AttrSet
        - friendlyName := "testAlias"
    - SafeBag
      - Pkcs-12-CertBag := *CACert*
      - PKCS12AttrSet
        - friendlyName := "testAlias"



### **A.33 Creating a PKCS#12 data structure with existing CA-signed certificate and a corresponding public key and private key with passphrase**

**Name:** HelperCreatePKCS12WithPassphrase

**Procedure Purpose:** Helper procedure to create a PKCS#12 data structure with existing CA-signed certificate and a corresponding public key and private key with passphrase.

**Pre-requisite:** None.

**Input:** An X.509 CA certificate (*CAcert*) that is compliant to [RFC5280] and a corresponding private key (*privateKey*) and public key (*publicKey*), and passphrase (*passphrase*).

**Returns:** A [PKCS#12] compliant PKCS#12 data structure (*PKCS12data*).

**Procedure:**

1. Use the current PrivateKeyInfo data:
  - PrivateKeyInfo
    - version := v2
    - privateKeyAlgorithm := rsaEncryption
    - privateKey := *privateKey*
    - publicKey := *publicKey*
2. Create an EncryptedPrivateKeyInfo data structure with the following properties:
  - EncryptedPrivateKeyInfo
    - encryptionAlgorithm := pbeWithSHAAnd3-KeyTripleDES-CBC
    - encryptedData := encrypted with *passphrase* PrivateKeyInfo data
3. Create an [PKCS#12] compliant PKCS#12 data structure (*PKCS12data*) with the following properties:
  - version := v3
  - authSafe
    - SafeBag
      - Pkcs-12-PKCS9ShroudedKeyBag := EncryptedPrivateKeyInfo
      - PKCS12AttrSet
        - friendlyName := "testAlias"
    - SafeBag
      - Pkcs-12-CertBag := CAcert
      - PKCS12AttrSet



- friendlyName := "testAlias"



### **A.34 Subject for a server certificate (all DN-attributes)**

Use the following subject for test cases:

- Subject.Country := "US"
- Subject.Organization := "ONVIF Test"
- Subject.OrganizationalUnit := "Unit test"
- Subject.DistinguishedNameQualifier := "DN Qualifier Test"
- Subject.StateOrProvinceName := "State Name Test"
- Subject.CommonName := "Common Name Test"
- Subject.SerialNumber := "000000000000"
- Subject.Locality := "LA"
- Subject.Title := "Mr"
- Subject.Surname := "SurnameTest"
- Subject.GivenName := "GivenNameTest"
- Subject.Initials := "AS"
- Subject.Pseudonym := "Pseudonym Test"
- Subject.GenerationQualifier := "GenerationQualifier Test"
- Subject.GenericAttribute.Type := "string"
- Subject.GenericAttribute.Value := "Test GenericAttribute"
- Subject.MultyValueRDN.Attribute.Type := "string"
- Subject.MultyValueRDN.Attribute.Value := "Test MultyValueRDN"



**A.35 Delete a certification path with corresponding certificate and RSA key pair when CertificateID is unknown**

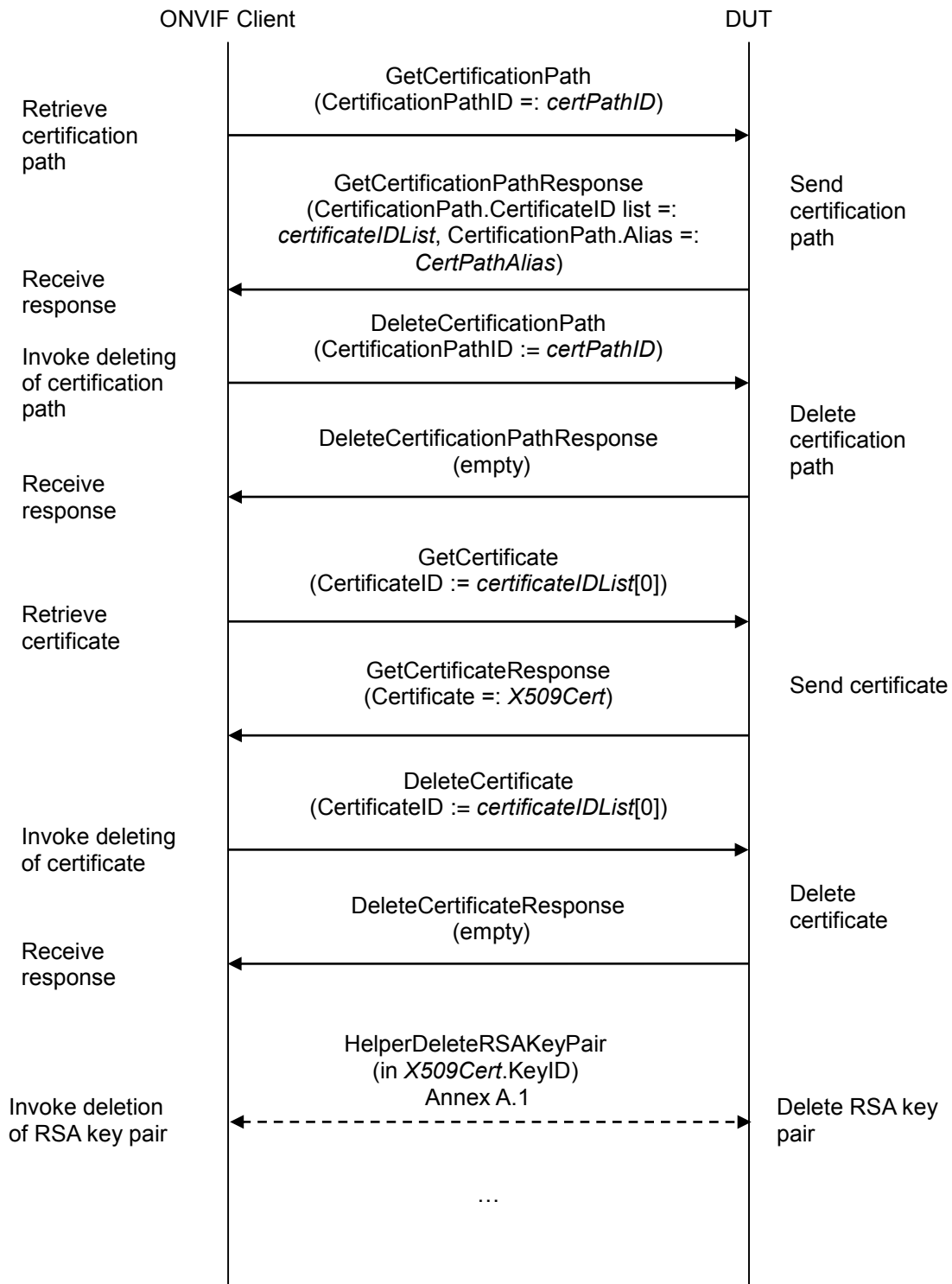
**Name:** HelperDeleteCertificationPath3

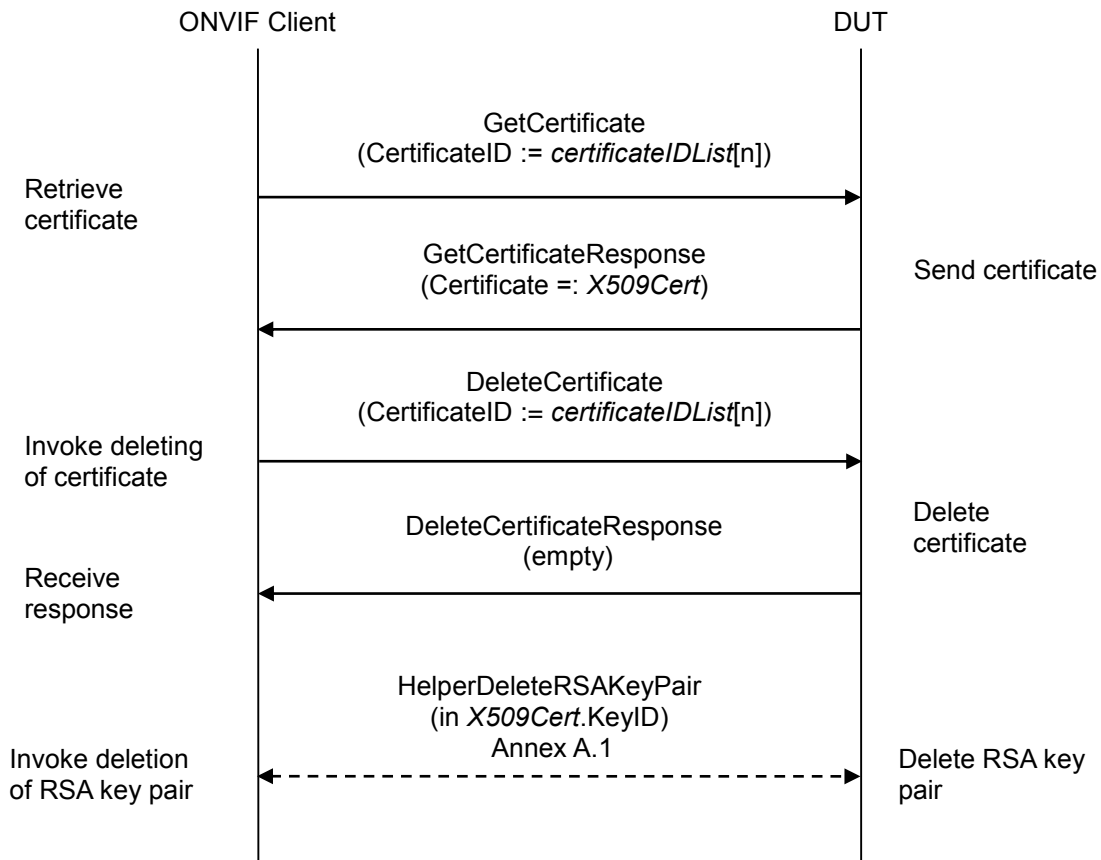
**Procedure Purpose:** Helper procedure to delete certification path and related certificate and RSA key pair when CertificateID is unknown.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate by the DUT as indicated by the SelfSignedCertificateCreationWithRSA or PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSACapability. TLS is supported by the DUT as indicated by the TLSServerSupported capability.

**Input:** The identifier of the certification path (*certPathID*) and RSA key pair (*keyID*) to delete.

**Returns:** None





**Procedure:**

1. ONVIF Client invokes **GetCertificationPath** message with parameters
  - CertificationPathID =: *certPathID*
2. The DUT responds with a **GetCertificationPathResponse** message with parameters
  - CertificationPath.CertificateID list =: *certificateIDList*
  - CertificationPath.Alias =: *CertPathAlias*
3. If the DUT did not send a **GetCertificationPathResponse** message, FAIL the test and go to step 9.
4. ONVIF Client invokes **DeleteCertificationPath** message with parameters
  - CertificationPathID =: *certPathID*
5. The DUT responds with empty **DeleteCertificationPathResponse** message.
6. If the DUT did not send a **DeleteCertificationPathResponse** message, FAIL the test and go to step 8.
7. For each CertificateID (*certificateID*) in *certificateIDList*:



7.1. ONVIF Client invokes **GetCertificate** message with parameters

- CertificateID := *certID*

7.2. The DUT responds with a **GetCertificateResponse** message with parameters

- Certificate =: *X509Cert*

7.3. ONVIF Client invokes **DeleteCertificate** with parameters

- CertificateID := *certificateID*

7.4. The DUT responds with a **DeleteCertificateResponse** message.

7.5. ONVIF Client deletes the RSA key pair (in *X509Cert.KeyID*) by following the procedure mentioned in Annex A.1 to restore DUT configuration.

8. Skip other steps.

9. ONVIF Client deletes the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.1 to restore DUT configuration.

**Procedure Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **GetCertificationPathResponse** message.

The DUT did not send **DeleteCertificationPathResponse** message.

The DUT did not send **DeleteCertificateResponse** message.

The DUT did not send **GetCertificateResponse** message.





### A.36 Upload PKCS#12 – no key pair exists

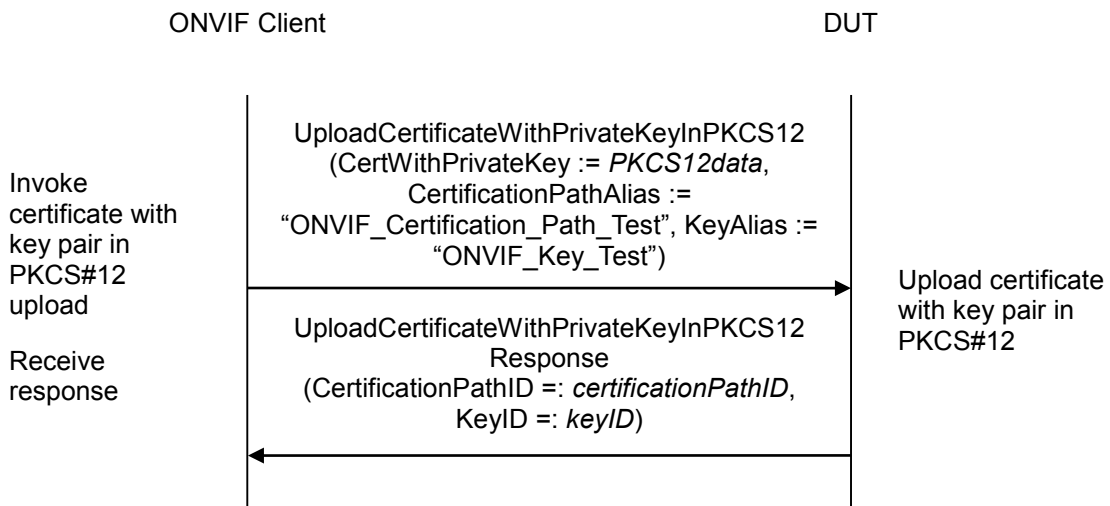
**Name:** HelperUploadPKCS12

**Procedure Purpose:** Helper procedure to create and upload PKCS#12 data structure with new public key and private key.

**Pre-requisite:** Advanced Security Service is received from the DUT. Certificate along with an RSA private key in a PKCS#12 data structure upload is supported by the DUT as indicated by the `PKCS12CertificateWithRSAPrivateKeyUpload` capability. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. The DUT shall have enough free storage capacity for one additional certification path.

**Input:** None

**Returns:** A [PKCS#12] compliant PKCS#12 data structure (*PKCS12data*) with CA certificate (*CAcert*) and a corresponding public key (*publicKey*) in the certificate along with the corresponding private key (*privateKey*) and certification path ID (*certificationPathID*) with corresponding key pair ID (*keyID*) for uploaded PKCS#12 data structure.



**Procedure:**

1. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding public key (out *publicKey*) in the certificate along with the corresponding private key (out *privateKey*) in the form of a PKCS#12 file (out *PKCS12data*) by following the procedure described in Annex A.30.
2. ONVIF Client invokes **UploadCertificateWithPrivateKeyInPKCS12** with parameters
  - `CertWithPrivateKey := PKCS12data`
  - `CertificationPathAlias := "ONVIF_Certification_Path_Test"`
  - `KeyAlias := "ONVIF_Key_Test"`
  - `IgnoreAdditionalCertificates := false`



- IntegrityPassphraseID skipped
  - EncryptionPassphraseID skipped
3. The DUT responds with a **UploadCertificateWithPrivateKeyInPKCS12Response** message with parameters
- CertificationPathID =: *certificationPathID*
  - KeyID =: *keyID*

**Procedure Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **UploadCertificateWithPrivateKeyInPKCS12Response** message.



### **A.37 Provide CRL**

**Name:** HelperCreateCRL

**Procedure Purpose:** Helper procedure to create CRL.

**Pre-requisite:** None.

**Input:** None.

**Returns:** A CRL (*cr*) that is compliant to [RFC5280].

**Procedure:**

1. ONVIF Client creates a CRL that is compliant to [RFC5280] and has the following properties:

- `tbsCertList[0].version` := v2
- `tbsCertList[0].signature.algorithm` := sha1-WithRSAEncryption
- `tbsCertList[0].issuer` := "CN=ONVIF TT,C=US"
- `tbsCertList[0].thisUpdate` := [current time]
- `tbsCertList[0].nextUpdate` skipped
- `tbsCertList[0].revokedCertificates[0].userCertificate` := [any certificate number]
- `tbsCertList[0].revokedCertificates[0].revocationDate` := [current time]
- `signatureAlgorithm.algorithm` := sha1-WithRSAEncryption
- `signatureValue` := sha1-WithRSAEncryption signature

**Note:** ONVIF Client may return the same CRL in subsequent invocations of this procedure.



### A.38 Delete a CRL

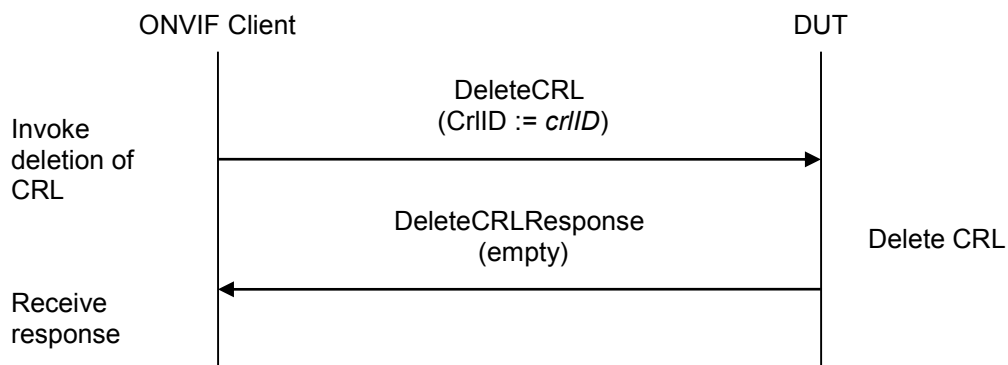
**Name:** HelperDeleteCRL

**Procedure Purpose:** Helper procedure to delete a CRL.

**Pre-requisite:** Advanced Security Service is received from the DUT. CRLs supported by the DUT as indicated by the MaximumNumberOfCRLs capability.

**Input:** The identifier of CRL (*crIID*) to delete.

**Returns:** None



**Procedure:**

1. ONVIF Client invokes **DeleteCRL** with parameters
  - CrIID := *crIID*
2. The DUT responds with a **DeleteCRLResponse** message.

**Procedure Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **DeleteCRLResponse** message.



### A.39 Upload a CRL

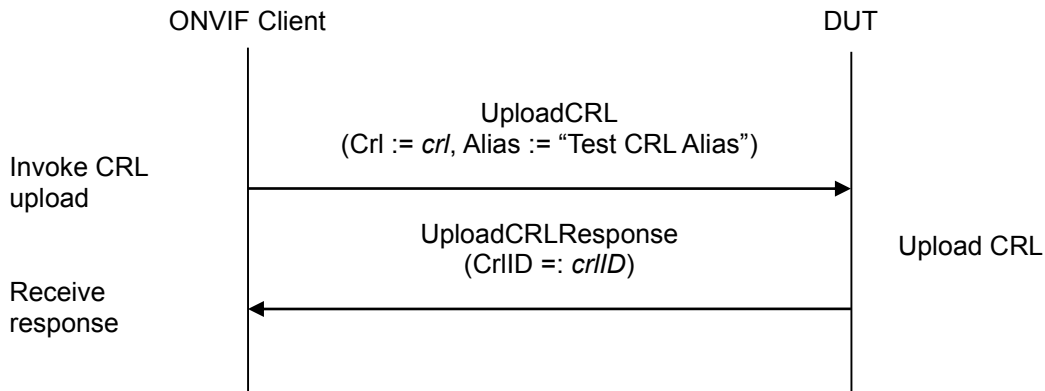
**Name:** HelperUploadCRL

**Procedure Purpose:** Helper procedure to upload a CRL.

**Pre-requisite:** Advanced Security Service is received from the DUT. CRLs supported by the DUT as indicated by the MaximumNumberOfCRLs capability. The DUT shall have enough free storage capacity for one additional CRL.

**Input:** The CRL alias (*alias*) and the CRL (*crf*).

**Returns:** The CRL identifier (*crIID*).



**Procedure:**

1. ONVIF Client invokes **UploadCRL** with parameter
  - CrI := *crf*
  - Alias := "Test CRL Alias"
  - anyParameters skipped
2. The DUT responds with **UploadCRLResponse** message with parameters
  - CrIID =: *crIID*

**Procedure Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **UploadCRLResponse** message.



### A.40 Delete a certification path validation policy

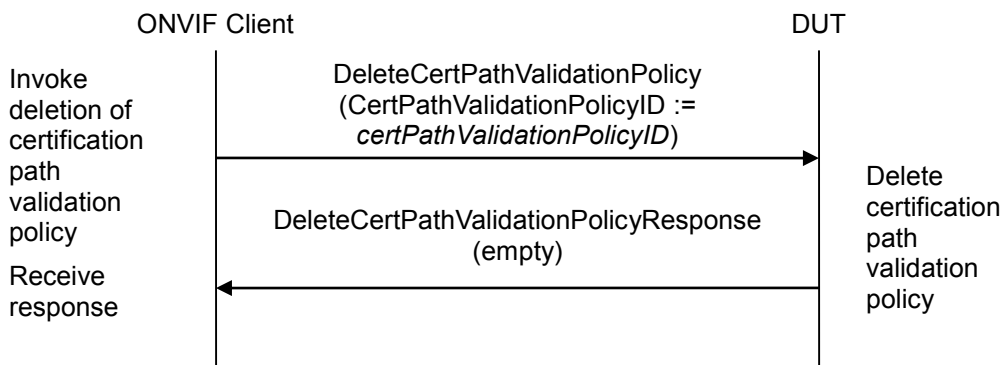
**Name:** HelperDeleteCertPathValidationPolicy

**Procedure Purpose:** Helper procedure to delete a certification path validation policy.

**Pre-requisite:** Advanced Security Service is received from the DUT. Certification path validation policy supported by the DUT as indicated by the `MaximumNumberOfCertificationPathValidationPolicies` capability.

**Input:** The identifier of certification path validation policy (`certPathValidationPolicyID`) to delete.

**Returns:** None



**Procedure:**

1. ONVIF Client invokes **DeleteCertPathValidationPolicy** with parameters
  - `CertPathValidationPolicyID := certPathValidationPolicyID`
2. The DUT responds with a **DeleteCertPathValidationPolicyResponse** message.

**Procedure Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **DeleteCertPathValidationPolicyResponse** message.



**A.41 Prepare certificate on the DUT**

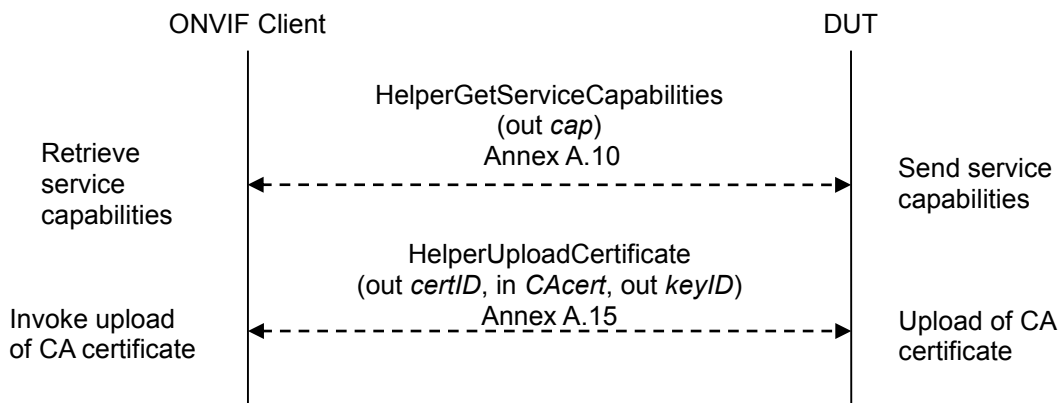
**Name:** HelperPrepareCertificate

**Procedure Purpose:** Helper procedure to create or upload certificate on the DUT.

**Pre-requisite:** Advanced Security Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreationWithRSA capability and RSA key pair generation supported by the DUT as indicated by the RSAKeyPairGeneration capability or create PKCS#10 supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA capability or certificate along with an RSA private key in a PKCS#12 data structure upload is supported by the DUT as indicated by the PKCS12CertificateWithRSAPrivateKeyUpload capability. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate.

**Input:** None.

**Returns:** The identifier of the new certificate (*certID*), RSA key pair (*keyID*) and certification path if any (*certificationPathID*).



**Procedure:**

1. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.10.
2. If *cap.KeystoreCapabilities.PKCS10ExternalCertificationWithRSA*:
  - 2.1. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding private key (out *privateKey*) by following the procedure described in Annex A.4.
  - 2.2. ONVIF Client uploads a CA certificate (out *certID*, in *CAcert*) and new RSA key pair with the public key from the CA certificate (out *keyID*) by following the procedure described in Annex A.15.
  - 2.3. Set:
    - *certificationPathID* := null



2.4. Skip other steps.

3. If `cap.KeystoreCapabilities.SelfSignedCertificateCreationWithRSA` and `cap.KeystoreCapabilities.RSAKeyPairGeneration`:

3.1. ONVIF Client creates a self-signed certificate (out *certID*) and related RSA key pair (out *keyID*) by following the procedure mentioned in Annex A.8.

3.2. Set:

- *certificationPathID* := null

3.3. Skip other steps.

4. If `cap.KeystoreCapabilities.PKCS12CertificateWithRSAPrivateKeyUpload`:

4.1. ONVIF Client creates a CA certificate (out *CAcert*) and a corresponding public key (out *publicKey*) in the certificate along with the corresponding private key (out *privateKey*) in the form of a PKCS#12 file (out *PKCS12data*) and uploads it with certification path ID (out *certificationPathID*) and key pair ID (out *keyID*) by following the procedure described in Annex A.36.

4.2. ONVIF Client invokes **GetCertificationPath** message with parameters

- *CertificationPathID* =: *certificationPathID*

4.3. The DUT responds with a **GetCertificationPathResponse** message with parameters

- *CertificationPath.CertificateID[0]* =: *certID*
- *CertificationPath.Alias*

5. If (`cap.KeystoreCapabilities.PKCS10ExternalCertificationWithRSA` = false or skipped) and (`cap.KeystoreCapabilities.SelfSignedCertificateCreationWithRSA` = false or skipped or `cap.KeystoreCapabilities.RSAKeyPairGeneration` = false or skipped) and (`cap.KeystoreCapabilities.PKCS12CertificateWithRSAPrivateKeyUpload` = false or skipped), FAIL the test and skip other steps.

#### Procedure Result:

#### PASS –

The DUT passed all assertions.

#### FAIL –

The DUT did not send **GetCertificationPathResponse** message.





**A.42 Create a certification path validation policy**

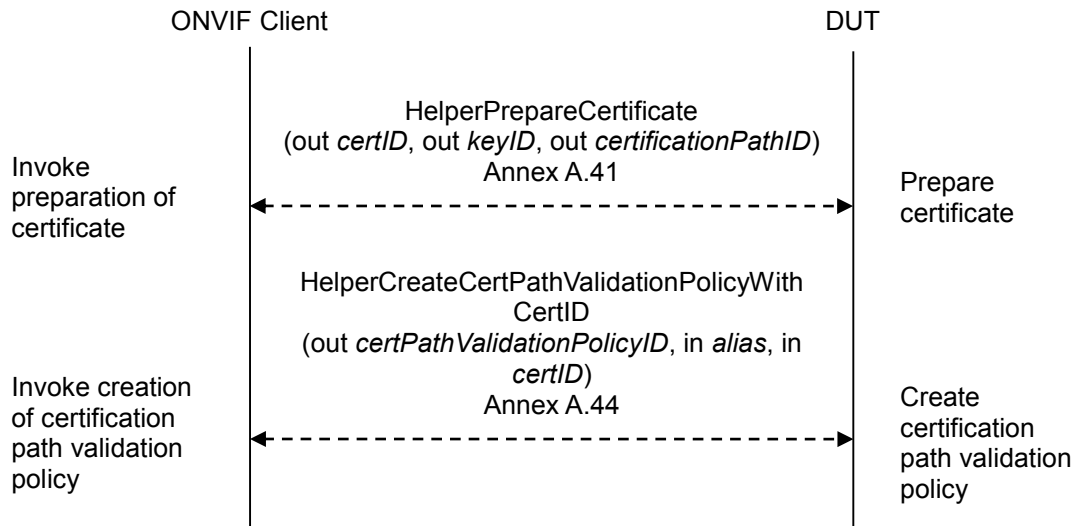
**Name:** HelperCreateCertPathValidationPolicy

**Procedure Purpose:** Helper procedure to create a certification path validation policy.

**Pre-requisite:** Advanced Security Service is received from the DUT. Certification path validation policy supported by the DUT as indicated by the MaximumNumberOfCertificationPathValidationPolicies capability. The DUT shall have enough free storage capacity for one additional certification path validation policy.

**Input:** The certification path validation policy alias (*alias*).

**Returns:** The certification path validation policy identifier (*certPathValidationPolicyID*), related certificate (*certID*), RSA key pair (*keyID*) and certification path if any (out *certificationPathID*).



**Procedure:**

1. ONVIF Client prepares certificate (out *certID*), related RSA key pair (out *keyID*) and certification path if any (out *certificationPathID*) by following the procedure mentioned in Annex A.41.
2. ONVIF Client creates certification path validation policy identifier (out *certPathValidationPolicyID*) with specified alias (in *alias*) and the certificate identifier (in *certID*) for trust anchor by following the procedure mentioned in Annex A.44.

**Procedure Result:**

**PASS –**

The DUT passed all assertions.



### **A.43 Provide certificate signed by private key of other certificate**

**Name:** HelperCreateSignedCertificate

**Procedure Purpose:** Helper procedure to create an X.509 certificate signed by private key of other certificate.

**Pre-requisite:** None.

**Input:** The subject (*subject*) of certificate and private key (*inputPrivateKey*) of the CA-certificate (*cert*).

**Returns:** An X.509 certificate (*cert*) signed by input private key that is compliant to [RFC5280] and a corresponding private key (*privateKey*) and public key (*publicKey*).

**Procedure:**

1. ONVIF Client creates an X.509 certificate signed by *inputPrivateKey* that is compliant to [RFC5280] and has the following properties:
  - version:= v3
  - signature := sha1-WithRSAEncryption
  - validity := validity from *cert*
  - subject := *subject*
  - issuerDN := subjectDN from *cert*

**Note:** ONVIF Client may return the same certificate in subsequent invocations of this procedure for the same subject and private key.



#### A.44 Create a certification path validation policy with provided certificate identifier

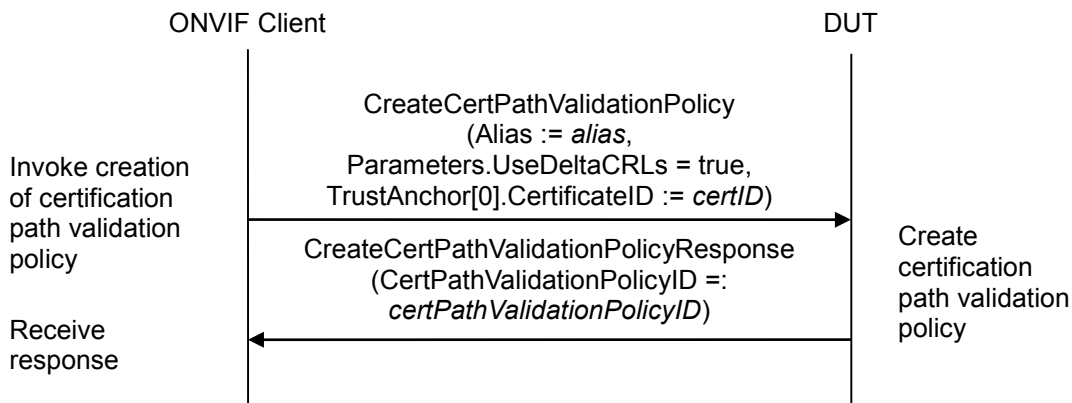
**Name:** HelperCreateCertPathValidationPolicyWithCertID

**Procedure Purpose:** Helper procedure to create a certification path validation policy with provided certificate identifier.

**Pre-requisite:** Advanced Security Service is received from the DUT. Certification path validation policy supported by the DUT as indicated by the `MaximumNumberOfCertificationPathValidationPolicies` capability. The DUT shall have enough free storage capacity for one additional certification path validation policy.

**Input:** The certification path validation policy alias (*alias*) and the certificate identifier (*certID*) for trust anchor.

**Returns:** The certification path validation policy identifier (*certPathValidationPolicyID*).



**Procedure:**

1. ONVIF Client invokes **CreateCertPathValidationPolicy** with parameters
  - Alias := *alias*
  - Parameters.RequireTLSWWWClientAuthExtendedKeyUsage skipped
  - Parameters.UseDeltaCRLs = true
  - Parameters.anyParameters skipped
  - TrustAnchor[0].CertificateID := *certID*
  - anyParameters skipped
2. The DUT responds with **CreateCertPathValidationPolicyResponse** message with parameters
  - CertPathValidationPolicyID := *certPathValidationPolicyID*

**Procedure Result:**

**PASS –**



The DUT passed all assertions.

**FAIL –**

The DUT did not send **CreateCertPathValidationPolicyResponse** message.



#### **A.45 Provide CRL for specified certificate**

**Name:** HelperCreateCRLForCertificate

**Procedure Purpose:** Helper procedure to create CRL for specified certificate signed with specified key.

**Pre-requisite:** None.

**Input:** The certificate for revocation (*cert*) and private key for signature (*privateKey*).

**Returns:** A CRL (*cr*) that is compliant to [RFC5280].

**Procedure:**

1. ONVIF Client creates a CRL that is compliant to [RFC5280] signed signed by private key *privateKey* and has the following properties:
  - `tbsCertList[0].version := v2`
  - `tbsCertList[0].signature.algorithm := sha1-WithRSAEncryption`
  - `tbsCertList[0].issuer := "ONVIF_DTT"`
  - `tbsCertList[0].thisUpdate := [current time]`
  - `tbsCertList[0].nextUpdate` skipped
  - `tbsCertList[0].revokedCertificates[0].userCertificate := cert`
  - `tbsCertList[0].revokedCertificates[0].revocationDate := [current date]`
  - `signatureAlgorithm.algorithm := sha1-WithRSAEncryption`
  - `signatureValue := sha1-WithRSAEncryption signature`

**Note:** ONVIF Client may return the same CRL in subsequent invocations of this procedure.



### A.46 Upload a passphrase

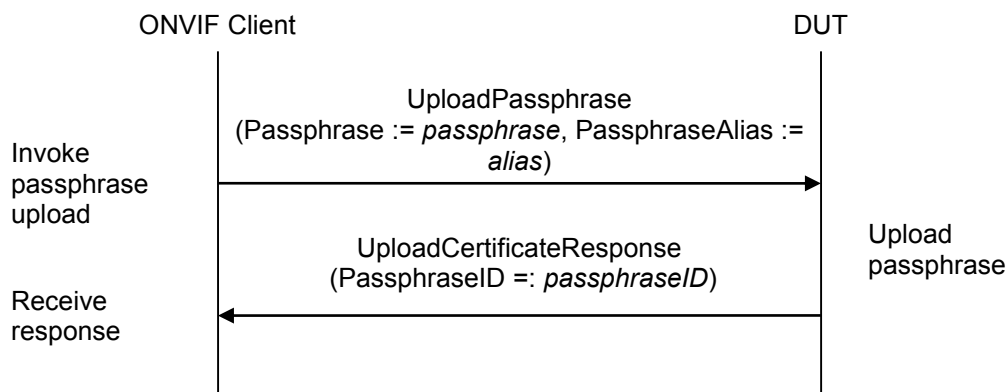
**Name:** HelperUploadPassphrase

**Procedure Purpose:** Helper procedure to upload a passphrase.

**Pre-requisite:** Advanced Security Service is received from the DUT. Passphrase handling is supported by the DUT as indicated by the MaximumNumberOfPassphrases capability. The DUT shall have enough free storage capacity for one additional passphrase.

**Input:** The passphrase (*passphrase*) and passphrase alias (*alias*).

**Returns:** The passphrase identifier (*passphraseID*).



**Procedure:**

1. ONVIF Client invokes **UploadPassphrase** with parameters
  - Passphrase := *passphrase*
  - PassphraseAlias := *alias*
2. The DUT responds with an **UploadPassphraseResponse** message with parameters
  - PassphraseID := *passphraseID*

**Procedure Result:**

**PASS –**

The DUT passed all assertions.

**FAIL –**

The DUT did not send **UploadPassphraseResponse** message.