ONVIF®

Security Configuration Device Test Specification

Version 25.06

June 2025

REVISION HISTORY

Vers.	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	New tests were added: Create PKCS#10 – negative test, Delete Certificate – CA – Preserve Public Key, Upload certificate – delete linked key (negative test), Upload certificate – Upload malformed certificate (negative test), Upload certificate – Upload expired certificate, TLS Server Certificate - self-signed, TLS Server Certificate – CA. Annex A.21, A.22 were added.
14.12	Dec, 2014	The following test cases were added: Create PKCS#10 – Subject Test, Create self-signed certificate – Subject Test, Upload Passphrase, Delete Passphrase, Upload PKCS8 – no key pair exists, 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. The following test cases were modified with ID change: Basic TLS Handshake, Basic TLS Handshake after Replace Server Certificate Assignment.

		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.
15.06	Jun, 2015	The following test cases were added: ADVANCED_SECURITY-3-2-5 Basic TLS Handshake with Replace Server Certification Path and PKCS#12 ADVANCED_SECURITY-6-3-4 Upload PKCS12 - verify key and certificate ADVANCED_SECURITY-2-1-27 CreateSelfSignedCertificate with PKCS#12 ADVANCED_SECURITY-2-1-28 Create PKCS#10 request with PKCS#12 ADVANCED_SECURITY-8-1-1 Upload CRL ADVANCED_SECURITY-8-1-2 Delete CRL ADVANCED_SECURITY-8-1-3 Get CRL ADVANCED_SECURITY-8-1-4 Create certification path validation policy ADVANCED_SECURITY-8-1-5 Get certification path validation policy ADVANCED_SECURITY-8-1-6 Delete certification path validation policy ADVANCED_SECURITY-3-3-1 TLS client authentication – self-signed TLS server certificate with on-device RSA key pair ADVANCED_SECURITY-3-3-2 CRL processing with on-device RSA key pair ADVANCED_SECURITY-3-3-3 Replace certification path validation policy assignment The following test cases were modified: ADVANCED_SECURITY-1-1-3 Check private Key status for an RSA private key ADVANCED_SECURITY-2-1-1 Create PKCS#10 certification requests ADVANCED_SECURITY-2-1-2 Create self-signed certificate ADVANCED_SECURITY-2-1-3 Upload certificate – Keystore contains private key ADVANCED_SECURITY-2-1-6 Get certificate – self-signed ADVANCED_SECURITY-2-1-8 Get all certificates – self signed ADVANCED_SECURITY-2-1-10 Delete Certificate – self signed ADVANCED_SECURITY-2-1-12 Create Certification Path – self-signed

		ADVANCED_SECURITY-2-1-13 Create Certification Path – CA ADVANCED_SECURITY-2-1-14 Get Certification Path – self-signed ADVANCED_SECURITY-2-1-15 Get Certification Path – CA ADVANCED_SECURITY-2-1-16 Get All Certification Paths – self-signed ADVANCED_SECURITY-2-1-17 Get All Certification Paths – CA ADVANCED_SECURITY-2-1-18 Delete Certification Path – self-signed ADVANCED_SECURITY-2-1-19 Delete Certification Path - CA ADVANCED_SECURITY-2-1-20 Create PKCS#10 (negative test) ADVANCED_SECURITY-2-1-22 Upload certificate – delete linked key (negative test) ADVANCED_SECURITY-3-1-1 Add Server Certificate Assignment – self-signed ADVANCED_SECURITY-3-1-2 Add Server Certificate Assignment – CA ADVANCED_SECURITY-3-1-3 Replace Server Certificate Assignment – self-signed ADVANCED_SECURITY-3-1-4 Replace Server Certificate Assignment – CA ADVANCED_SECURITY-3-1-5 Get Assigned Server Certificates – self-signed ADVANCED_SECURITY-3-1-6 Get Assigned Server Certificates – CA ADVANCED_SECURITY-3-1-7 Remove Server Certificate Assignment – self-signed ADVANCED_SECURITY-3-1-8 Remove Server Certificate Assignment – CA ADVANCED_SECURITY-3-2-3 Basic TLS Handshake ADVANCED_SECURITY-3-2-4 Basic TLS Handshake after Replace Server Certificate Assignment ADVANCED_SECURITY-4-1-1 TLS Server Certificate - self-signed ADVANCED_SECURITY-4-1-2 TLS Server Certificate – CA ADVANCED_SECURITY-5-1-1 Advanced Security Service Capabilities ADVANCED_SECURITY-5-1-2 Get Services and Get Advanced Security Service Capabilities Consistency The following annexes were added: A.36, A.37, A.38, A.39, A.40, A.41, A.42, A.43, A.44, A.45, A.46 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.
17.06	Jun 22, 2017	The document formating were updated.
18.06	Mar 14, 2018	timeout1 variable was replaced by operationDelay variable.
18.06	Mar 15, 2018	The following were updated according to #1562: Annex A.47 Remove Server Certificate Assignment (added) Annex A.48 Restore Server Certificate Assignment (added) ADVANCED_SECURITY-3-1-1 Add Server Certificate Assignment – self-signed (steps 3 and 8 were added) ADVANCED_SECURITY-3-1-2 Add Server Certificate Assignment – CA (steps 3 and 8 were added) ADVANCED_SECURITY-3-1-3 Replace Server Certificate Assignment – self-signed (steps 3 and 18 were added) ADVANCED_SECURITY-3-1-4 Replace Server Certificate Assignment – CA (steps 3 and 25 were added) ADVANCED_SECURITY-3-1-5 Get Assigned Server Certificates – self-signed (steps 3 and 12 were added) ADVANCED_SECURITY-3-1-6 Get Assigned Server Certificates – CA (steps 3 and 14 were added) ADVANCED_SECURITY-3-1-7 Remove Server Certificate Assignment – self-signed (steps 3 and 17 were added) ADVANCED_SECURITY-3-1-8 Remove Server Certificate Assignment – CA (steps 3 and 18 were added) ADVANCED_SECURITY-3-2-3 Basic TLS Handshake (steps 3 and 23 were added) ADVANCED_SECURITY-3-2-4 Basic TLS Handshake after Replace Server Certificate Assignment (steps 3 and 39 were added) ADVANCED_SECURITY-3-2-5 Basic TLS Handshake with Replace Server Certification Path and PKCS#12 (steps 3 and 40 were added) ADVANCED_SECURITY-3-3-1 TLS client authentication – self-signed TLS server certificate with on-device RSA key pair (steps 3 and 38 were added) ADVANCED_SECURITY-3-3-2 CRL processing with on-device RSA key pair (steps 3 and 40 were added) ADVANCED_SECURITY-4-1-1 TLS Server Certificate - self-signed (steps 3 and 22 were added) ADVANCED_SECURITY-4-1-2 TLS Server Certificate – CA (steps 3 and 35 were added)
18.06	Mar 15, 2018	The following were updated according to #1586: ADVANCED_SECURITY-1-1-1 Create RSA Key Pair, status through polling (steps 4.3, 4.3.1, 4.4 were updated)

		ADVANCED_SECURITY-1-1-2 Create RSA Key Pair, status through event (steps 6.3, 6.4 were updated) Annex A.7 Create an RSA key pair (steps 4, 4.1, 5 were updated)
18.06	Apr 17, 2018	The following were updated according to #1615: Annex A.4 Provide CA certificate (step 1 added, step 3 updated) Annex A.22 Provide expired CA certificate (step 1 added, step 4 updated)
18.06	Apr 23, 2018	The following were updated according to #1594: ADVANCED_SECURITY-5-1-1 Advanced Security Service Capabilities (step 13.1 were updated)
18.06	May 03, 2018	The following were updated according to #1593: ADVANCED_SECURITY-3-1-1 Add Server Certificate Assignment – self-signed (step 7 was added) ADVANCED_SECURITY-3-1-2 Add Server Certificate Assignment – CA (step 7 was added) ADVANCED_SECURITY-3-1-3 Replace Server Certificate Assignment – self-signed (steps 7, 14 were added) ADVANCED_SECURITY-3-1-4 Replace Server Certificate Assignment – CA (steps 11, 21 were added) ADVANCED_SECURITY-3-1-6 Get Assigned Server Certificates – CA (step 9 was added) ADVANCED_SECURITY-3-1-7 Remove Server Certificate Assignment – self-signed (step 13 was added) ADVANCED_SECURITY-3-1-8 Remove Server Certificate Assignment – CA (steps 9, 16 were added) ADVANCED_SECURITY-3-2-3 Basic TLS Handshake (step 12 was added) ADVANCED_SECURITY-3-2-4 Basic TLS Handshake after Replace Server Certificate Assignment (steps 12, 25 were added) ADVANCED_SECURITY-3-2-5 Basic TLS Handshake with Replace Server Certification Path and PKCS#12 (steps 13, 27, 37 were added) ADVANCED_SECURITY-4-1-1 TLS Server Certificate - self-signed (step 13 was added) ADVANCED_SECURITY-4-1-2 TLS Server Certificate – CA (steps 7, 20 were added) Annex A.12 Remove server certificate assignment with corresponding certification path, certificate and RSA key pair (step 3 was added) Annex A.13 Add server certificate assignment with corresponding certification path, self-signed certificate and RSA key pair (step 4 was added)

		Annex A.20 Remove server certificate assignment with corresponding certification path, certificates and RSA key pairs (step 3 was added) Annex A.47 Remove Server Certificate Assignment (step 8.4 was added) Annex A.48 Restore Server Certificate Assignment (step 1.3 was added)
18.06	May 07, 2018	The following were updated according to #1632: ADVANCED_SECURITY-2-1-20 CreatePKCS10CSR – negative test (step 11 was updated)
18.06	May 16, 2018	The following were updated according to #1619: Annex A.45 Provide CRL for specified certificate (step 1 was changed)
18.06	Jun 21, 2018	Reformatting document using new template
18.12	Oct 1, 2018	The following were updated in the scope of #1599: ADVANCED_SECURITY-2-1-27 CreateSelfSignedCertificate with PKCS#12 (Pre-Requisite updated with new item) ADVANCED_SECURITY-2-1-28 Create PKCS#10 request with PKCS#12 (Pre-Requisite updated with new item) ADVANCED_SECURITY-3-1-3 Replace Server Certificate Assignment - Self-Signed (test procedure updates to create new RSA key pair for second certificate) ADVANCED_SECURITY-3-1-4 Replace Server Certificate Assignment - CA (test procedure updates to create new RSA key pair for second CA-signed certificate) ADVANCED_SECURITY-3-2-4 Basic TLS Handshake after Replace Server Certificate Assignment (test procedure updates to create new RSA key pair for second certificate)
18.12	Nov 12, 2018	The following were updated in the scope of #1653: Title was updated (Advanced Security Test Specification replaced with Security Configuration Device Test Specification) Introduction section was updated (ONVIF Advanced Security Test Specification was replaces with ONVIF Security Configuration Device Test Specification) Scope section was updated (ONVIF Advanced Security Test Specification was replaces with ONVIF Security Configuration Device Test Specification, ONVIF Advanced Security Service replaced with ONVIF Security Configuration Service) Normative references was updated ([ONVIF Advanced Security Service] ONVIF Advanced Security Specifications: replaced with [ONVIF Security Configuration Service] ONVIF Security Configuration Specifications:) Definition was updated (ONVIF Advanced Security Service replaced with ONVIF Security Configuration Service)

		Test Overview\Test Policy\Keystore was updated (Advanced Security Service was replaced with ONVIF Security Configuration Service)
		Test Overview\Test Policy\Certificate Management was updated (Advanced Security Service was replaced with ONVIF Security Configuration Service)
		Test Overview\Test Policy\TLS Server was updated (Advanced Security Service was replaced with ONVIF Security Configuration Service)
		Test Overview\Test Policy\Referential Integrity was updated (Advanced Security Service was replaced with ONVIF Security Configuration Service)
		Test Overview\Test Policy\Capabilities was updated (Advanced Security Service was replaced with ONVIF Security Configuration Service)
		Test Overview\Test Policy\Off-Device Key Generation Operations was updated (Advanced Security Service was replaced with ONVIF Security Configuration Service)
		Test Overview\Test Policy\Certificate-based Client Authentication was updated (Advanced Security Service was replaced with ONVIF Security Configuration Service)
		Advanced Security Test Cases chapter title was updated with Security Configuration Test Cases
		For all test cases Pre-Requisites were updated ("Advanced Security Service is received from the DUT." was replaced with "Security Configuration Service is received from the DUT.")
		For all test cases WSDL References were updated ("advancedsecurity.wsdl" was replaced with "security.wsdl")
		For all test cases Specification Coverage were updated ("Advanced Security, Keystore – Key Management" was replaced with "Key Management (ONVIF Security Configuration Service Specification)")
		For all test cases Specification Coverage were updated ("Advanced Security, Key Management" was replaced with "Key Management (ONVIF Security Configuration Service Specification)")
		For all test cases Specification Coverage were updated ("Advanced Security, Keystore - Certificate Management" was replaced with "Certificate Management (ONVIF Security Configuration Service Specification)")
		For all test cases Specification Coverage were updated ("Advanced Security, Certificate Management" was replaced with "Certificate Management (ONVIF Security Configuration Service Specification)")
		For all test cases Specification Coverage were updated ("Advanced Security, TLS Server" was replaced with "TLS Server (ONVIF Security Configuration Service Specification)")
		For all test cases Specification Coverage were updated ("Advanced Security, Capabilities" was replaced with "Capabilities (ONVIF Security Configuration Service Specification)")

		For all test cases Specification Coverage were updated ("Advanced Security, Passphrase Management" was replaced with "Passphrase Management (ONVIF Security Configuration Service Specification)") For all test cases Specification Coverage were updated ("Advanced Security, Upload Certificate Revocation List" was replaced with "CRL Management (ONVIF Security Configuration Service Specification)") For all test cases Specification Coverage were updated ("Advanced Security, Delete Certificate Revocation List" was replaced with "CRL Management (ONVIF Security Configuration Service Specification)") For all test cases Specification Coverage were updated ("Advanced Security, 7.6.2 Get Certificate Revocation List" was replaced with "CRL Management (ONVIF Security Configuration Service Specification)") For all test cases Specification Coverage were updated ("Advanced Security, Create Certification Path Validation Policy" was replaced with "Certification Path Validation Policy Management (ONVIF Security Configuration Service Specification)") For all test cases Specification Coverage were updated ("Advanced Security, Get Certification Path Validation Policy" was replaced with "Certification Path Validation Policy Management (ONVIF Security Configuration Service Specification)") For all test cases Specification Coverage were updated ("Advanced Security, Delete Certification Path Validation Policy" was replaced with "Certification Path Validation Policy Management (ONVIF Security Configuration Service Specification)") For all test cases Specification Coverage were updated ("Advanced Security, Replace Certification Path Validation Policy" was replaced with "Certification Path Validation Policy Management (ONVIF Security Configuration Service Specification)") Test ADVANCED_SECURITY-5-1-1 was renamed ("Advanced Security Service Capabilities" was replaced with "Security Configuration Service Capabilities") For test ADVANCED_SECURITY-5-1-1 test purpose was updated ("To verify DUT Advanced Security Service Capabilities." was replaced with "To verify DUT Security Configuration Service Capabilities.") Test ADVANCED_SECURITY-5-1-2 was renamed ("Get Services and Get Advanced Security Service Capabilities Consistency" was replaced with "Get Services and Get Security Configuration Service Capabilities Consistency") For test ADVANCED_SECURITY-5-1-2 test purpose was updated ("To verify Get Services and Advanced Security Service Capabilities consistency." was replaced with "To verify Get Services and Security Configuration Service Capabilities consistency.") Feature under test field for tests ADVANCED_SECURITY-5-1-1 and ADVANCED_SECURITY-5-1-2 were updated ("GetServiceCapabilities (for Advanced Security Service)" was replaced with "GetServiceCapabilities (for Security Configuration Service)") Other minor changes in description related to renaming of Advanced Security Service to Security Configuration Service.

18.12	Dec 21, 2018	Switching Hub description in 'Network Configuration for DUT' section was updated according to #1737
19.06	May 06, 2019	The following were updated in the scope of #1799: Security Configuration Test Cases\TLS Versions (added) ADVANCED_SECURITY-9-1-1 TLS Version Management (added) ADVANCED_SECURITY-9-1-2 Disable TLS Version (added) Annex A.49 Configuring HTTPS if Required (added) Annex A.50 Configuring HTTPS using Security Configuration Service (added) Annex A.51 Add server certificate assignment with corresponding certification path, CA certificate and RSA key pair (added) Annex A.52 Basic TLS Handshake With Protocol Version Alert (added) Scope\TLS Versions (added) Test Policy\TLS Versions (added)
20.06	May 13, 2020	Pre-Requisite of the following test cases updated with adding of Pull-Point Notification feature according to #1999: ADVANCED_SECURITY-1-1-2 Create RSA Key Pair, status through event
20.12	Dec 08, 2020	Pre-Requisites of the following test cases were updated with adding of Network Configuration feature according to #2094 ADVANCED_SECURITY-3-1-1 Add Server Certificate Assignment – self-signed ADVANCED_SECURITY-3-1-2 Add Server Certificate Assignment – CA ADVANCED_SECURITY-3-1-3 Replace Server Certificate Assignment – self-signed ADVANCED_SECURITY-3-1-4 Replace Server Certificate Assignment – CA ADVANCED_SECURITY-3-1-5 Get Assigned Server Certificates – self-signed ADVANCED_SECURITY-3-1-6 Get Assigned Server Certificates – CA ADVANCED_SECURITY-3-1-7 Remove Server Certificate Assignment – self-signed ADVANCED_SECURITY-3-1-8 Remove Server Certificate Assignment – CA ADVANCED_SECURITY-3-2-3 Basic TLS Handshake ADVANCED_SECURITY-3-2-4 Basic TLS Handshake after Replace Server Certificate Assignment

		ADVANCED_SECURITY-3-2-5 Basic TLS Handshake with Replace Server Certification Path and PKCS#12 ADVANCED_SECURITY-3-3-1 TLS client authentication – self-signed TLS server certificate with on-device RSA key pair ADVANCED_SECURITY-3-3-2 CRL processing with on-device RSA key pair ADVANCED_SECURITY-4-1-1 TLS Server Certificate - self-signed ADVANCED_SECURITY-4-1-2 TLS Server Certificate – CA ADVANCED_SECURITY-9-1-2 Disable TLS Version
23.06	Feb 20, 2023	The following was updated according #10 ADVANCED_SECURITY-3-2-5 Basic TLS Handshake with Replace Server Certification Path and PKCS#12 (step 8 was added, steps 9, 10, 23, 24 were updated) ADVANCED_SECURITY-6-3-1 Upload PKCS12 – no key pair exists (step 3 was added, steps 4, 5 were updated) ADVANCED_SECURITY-6-3-4 (step 3 was added, steps 4, 5 were updated) Annex HelperCreatePKCS12WithNewCACert was removed and replaced with Annex HelperCreatePKCS12WithNewCACertWithPassphrase Annex HelperCreatePKCS12WithExistingCACert was removed and replaced with Annex HelperCreatePKCS12WithPassphrase Annex HelperUploadPKCS12 (step 1 was added, steps 2, 3 were updated)
23.06	Feb 27, 2023	The following was updated according #11: ADVANCED_SECURITY-3-1-5 Get Assigned Server Certificates – self-signed (step 9 was updated, steps 4, 5, 10 were removed, note was removed) ADVANCED_SECURITY-3-1-6 Get Assigned Server Certificates – CA (step 12 was updated, steps 4, 5, 13 were removed, note was removed) ADVANCED_SECURITY-3-1-7 Remove Server Certificate Assignment – self-signed (step 17 was updated, steps 4,5, 7, 8, 9, 10 were removed, note was removed) ADVANCED_SECURITY-3-1-8 Remove Server Certificate Assignment – CA (step 20 was updated, steps 4,5, 10, 11, 12, 13 were removed, note was removed)
23.06	May 15, 2023	The following test was updated according #116: ADVANCED_SECURITY-5-1-1 Security Configuration Service Capabilities (steps 8.1 and 9.1 were removed) The following tests were removed according #116: ADVANCED_SECURITY-6-2-2 Upload PKCS8 – decryption fails ADVANCED_SECURITY-6-3-2 Upload PKCS12 – decryption fails

		The following tests were added according #116: ADVANCED_SECURITY-6-2-3 Upload Encrypted PKCS8 ADVANCED_SECURITY-6-2-4 Upload Encrypted PKCS8 – Using Passphrase Management ADVANCED_SECURITY-6-3-5 Upload PKCS12 – Decryption Failed ADVANCED_SECURITY-6-3-6 Upload PKCS12 – Using Passphrase Storage
23.06	Jun 22, 2023	The following test was updated according #135: ADVANCED_SECURITY-5-1-1 Security Configuration Service Capabilities (steps 14.4 about MaximumNumberOfCRLs presence was removed)
24.06	Feb 16, 2024	The following test and annexes were updated according #195: ADVANCED_SECURITY-3-3-3 Replace certification path validation policy assignment (step 3-4 was updated) Annex A.40 (step 1 was updated) Annex A.39 (steps 2.1 and 4.1 were updated) Annex A.34 (steps 2 was updated) Annex A.30 (steps 1 was updated)
25.06	Jun 11, 2024	The following test and annexes were updated according #209: Annex HelperSignatureAlgorithmSelection (new annex) Annex HelperCreateSelfSignedCertificate (new input parameter was added, step 2 was added) Annex HelperCreateCertificationPath_SelfSigned (input parameter for capabilities was added, step 1 was updated to use capabilities) Annex HelperPrepareCertificate (input parameter for capabilities was added, step 1 was removed, step 2.1 was updated to use capabilities) Annex HelperAddServerCertAssign_SSCertificate (input parameter for capabilities was added, step 1 was updated to use capabilities) Annex HelperCreateCertPathValidationPolicy (input parameter for capabilities was added, step 1 was updated to use capabilities) Annex HelperConfigureHTTPS (input parameter for capabilities was added, step 4.1 was updated to use capabilities, step 6 was updated to use capabilities) ADVANCED_SECURITY-2-1-6 Get certificate – self-signed (step 3 was added, step 4 was updated to use capabilities) ADVANCED_SECURITY-2-1-8 Get all certificates – self signed (step 3 was added, step 6 was updated to use capabilities) ADVANCED_SECURITY-2-1-10 Delete Certificate – self signed (step 3 was added, step 6 was updated to use capabilities) ADVANCED_SECURITY-2-1-12 Create Certification Path – self-signed (step 3 was added, step 4 was updated to use capabilities)

		ADVANCED_SECURITY-2-1-14 Get Certification Path – self-signed (step 3 was added, step 4 was updated to use capabilities) ADVANCED_SECURITY-2-1-16 Get All Certification Paths – self-signed (step 3 was added, step 4 was updated to use capabilities) ADVANCED_SECURITY-2-1-18 Delete Certification Path – self-signed (step 3 was added, step 6 was updated to use capabilities) ADVANCED_SECURITY-3-1-1 Add Server Certificate Assignment – self-signed (step 3 was added, step 5 was updated to use capabilities) ADVANCED_SECURITY-3-1-3 Replace Server Certificate Assignment – self-signed (step 3 was added, steps 5 and 6 were updated to use capabilities) ADVANCED_SECURITY-3-1-5 Get Assigned Server Certificates – self-signed (step 3 was added, step 5 was updated to use capabilities) ADVANCED_SECURITY-3-1-7 Remove Server Certificate Assignment – self-signed (step 3 was added, step 5 was updated to use capabilities) ADVANCED_SECURITY-3-2-3 Basic TLS Handshake (step 3 was added, step 8 was updated to use capabilities) ADVANCED_SECURITY-3-2-4 Basic TLS Handshake after Replace Server Certificate Assignment (step 3 was added, steps 8 and 17 were updated to use capabilities) ADVANCED_SECURITY-3-3-1 TLS client authentication – self-signed TLS server certificate with on-device RSA key pair (step 3 was added, step 11 was updated to use capabilities) ADVANCED_SECURITY-3-3-2 CRL processing with on-device RSA key pair (step 3 was added, step 11 was updated to use capabilities) ADVANCED_SECURITY-3-3-3 Replace certification path validation policy assignment (step 3 was added, steps 4 and 5 were updated to use capabilities) ADVANCED_SECURITY-4-1-1 TLS Server Certificate - self-signed (step 3 was added, step 5 was updated to use capabilities) ADVANCED_SECURITY-8-1-4 Create certification path validation policy (step 3 was added, step 4 was updated to use capabilities) ADVANCED_SECURITY-8-1-5 Get certification path validation policy (step 3 was added, step 4 was updated to use capabilities) ADVANCED_SECURITY-8-1-6 Delete certification path validation policy (step 3 was added, step 4 was updated to use capabilities) ADVANCED_SECURITY-9-1-2 Disable TLS Version (step 5 was updated to use capabilities) Summary: Instead of using fixed signature algorithm, the one from Device capabilities to be used (see Annex HelperSignatureAlgorithmSelection usage). For almost all annexes capabilities were added as input parameter.

		Capabilities invoke/Annex HelperGetServiceCapabilities_AdvancedSecurity using was removed from all annexes and moved to the tests to prevent multiple capabilities request.
25.06	Oct 15, 2024	ADVANCED_SECURITY-2-1-1 Create PKCS#10 certification requests (step 4 was added, step 5 was updated to use selected algorithm) ADVANCED_SECURITY-2-1-2 Create self-signed certificate (step 4 was added, step 5 was updated to use selected algorithm) ADVANCED_SECURITY-2-1-20 CreatePKCS10CSR – negative test (step 10 was added, step 11 was updated to use selected algorithm) ADVANCED_SECURITY-2-1-27 CreateSelfSignedCertificate with PKCS#12 (steps 3 and 5 were added, step 6 was updated to use selected algorithm, step 4 was updated to use capabilities) ADVANCED_SECURITY-2-1-28 Create PKCS#10 request with PKCS#12 (steps 3 and 4 were added, steps 6 and 8 were updated to use selected algorithm, steps 5 and 9 were updated to use capabilities) ADVANCED_SECURITY-5-1-1 Security Configuration Service Capabilities (steps 9.12, 9.13, 10.6, 10.7, 11.4, 11.5 were removed) Annex HelperAddServerCertAssign_CACertificate (input parameter for capabilities was added, step 2 was added, steps 3 and 6 were updated to use selected algorithm, step 5 was updated to use capabilities) Annex HelperCreateCRLForCertificate (input parameter for capabilities was added, step 1 was added, step 2 was updated to use selected algorithm) ADVANCED_SECURITY-3-3-2 CRL processing with on-device RSA key pair (step 16 was updated to use capabilities) Annex HelperCreateSignedCertificate (input parameter for capabilities was added, step 1 was added, step 2 was updated to use selected algorithm) ADVANCED_SECURITY-3-3-1 TLS client authentication – self-signed TLS server certificate with on-device RSA key pair (steps 12, 14, 15 and 16 were updated to use capabilities) ADVANCED_SECURITY-3-3-2 CRL processing with on-device RSA key pair (steps 12, 14 and 15 were updated to use capabilities) Annex HelperCreateCRL (input parameter for capabilities was added, step 1 was added, step 2 was updated to use selected algorithm) ADVANCED_SECURITY-8-1-1 Upload CRL (step 3 was added, step 4 was updated to use capabilities) ADVANCED_SECURITY-8-1-2 Delete CRL (step 3 was added, step 4 was updated to use capabilities) ADVANCED_SECURITY-8-1-3 Get CRL (step 3 was added, step 4 was updated to use capabilities)

		Annex HelperCreateExpiredCACertificate (input parameter for capabilities was added, step 2 was added, step 5 was updated to use selected algorithm) ADVANCED_SECURITY-2-1-24 Upload certificate – Upload expired certificate (step 3 was added, step 4 was updated to use capabilities) Annex HelperCreateCASignedCertificate (input parameter for capabilities was added, step 2 was added, step 3 was updated to use selected algorithm, step 5 was updated to use capabilities) ADVANCED_SECURITY-2-1-3 Upload certificate – Keystore contains private key (step 3 was added, steps 4 and 5 were updated to use capabilities) ADVANCED_SECURITY-2-1-22 Upload certificate – delete linked key (negative test) (step 3 was added, steps 4 and 5 were updated to use capabilities) Annex HelperUploadCASignedCertificate (input parameter for capabilities was added, step 1 was updated to use capabilities) ADVANCED_SECURITY-2-1-13 Create Certification Path – CA (step 3 was added, steps 3 and 5 were updated to use capabilities) ADVANCED_SECURITY-3-1-4 Replace Server Certificate Assignment – CA (step 3 was added, steps 5, 7 and 13 were updated to use capabilities) Annex HelperCreateCertificationPath_CACertificates (input parameter for capabilities was added, steps 1 and 2 were updated to use capabilities) ADVANCED_SECURITY-4-1-2 TLS Server Certificate – CA (step 3 was added, step 5 was updated to use capabilities) ADVANCED_SECURITY-3-1-8 Remove Server Certificate Assignment – CA (step 3 was added, step 5 was updated to use capabilities) ADVANCED_SECURITY-3-1-6 Get Assigned Server Certificates – CA (step 3 was added, step 5 was updated to use capabilities) ADVANCED_SECURITY-3-1-2 Add Server Certificate Assignment – CA (step 3 was added, step 5 was updated to use capabilities) ADVANCED_SECURITY-2-1-19 Delete Certification Path - CA (step 3 was added, step 6 was updated to use capabilities) ADVANCED_SECURITY-2-1-17 Get All Certification Paths – CA (step 3 was added, step 6 was updated to use capabilities) ADVANCED_SECURITY-2-1-15 Get Certification Path – CA (step 3 was added, step 4 was updated to use capabilities) Annex HelperCreateCACertificate (input parameter for capabilities was added, step 4 was updated to use capabilities) ADVANCED_SECURITY-2-1-4 Upload certificate – Keystore contains private key (negative test) (step 3 was added, step 4 was updated to use capabilities)

		ADVANCED_SECURITY-2-1-5 Upload certificate – Keystore does not contain private key (step 3 was added, step 4 was updated to use capabilities) ADVANCED_SECURITY-2-1-7 Get certificate – CA (step 3 was added, step 4 was updated to use capabilities) ADVANCED_SECURITY-2-1-9 Get All Certificate – CA (step 3 was added, step 6 was updated to use capabilities) ADVANCED_SECURITY-2-1-11 Delete Certificate – CA (step 3 was added, step 6 was updated to use capabilities) ADVANCED_SECURITY-2-1-21 Delete Certificate – CA – Preserve Public Key (step 3 was added, step 4 was updated to use capabilities) ADVANCED_SECURITY-2-1-23 Upload certificate – Upload malformed certificate (negative test) (step 3 was added, step 4 was updated to use capabilities) Annex HelperCreatePKCS12WithNewCACertWithPassphrase (input parameter for capabilities was added, step 1 was updated to use capabilities) Annex HelperPrepareCertificate (steps 1.1 and 3.1 were updated to use capabilities) ADVANCED_SECURITY-3-2-5 Basic TLS Handshake with Replace Server Certification Path and PKCS#12 (step 3 was added, steps 10 and 24 were updated to use capabilities) ADVANCED_SECURITY-6-3-1 Upload PKCS12 – no key pair exists (step 3 was added, step 5 was updated to use capabilities) ADVANCED_SECURITY-6-3-4 Upload PKCS12 - verify key and certificate (step 3 was added, step 5 was updated to use capabilities) ADVANCED_SECURITY-6-3-5 Upload PKCS12 – Decryption Failed (step 3 was added, step 5 was updated to use capabilities) ADVANCED_SECURITY-6-3-6 Upload PKCS12 – Using Passphrase Storage (step 3 was added, step 5 was updated to use capabilities) Annex HelperUploadPKCS12 (input parameter for capabilities was added, step 2 was updated to use capabilities) Annex HelperCreateCertificateFromPKCS10CSR (input parameter for capabilities was added, step 1 was added, step 2 was updated to use selected algorithm)
25.06	Dec 1, 2024	The following test and annexes were updated according #271: Annex HelperDeleteRSAKeyPair was renamed and replaced with Annex HelperDeleteKeyPair Annex HelperCreateCertificateFromPKCS10CSR (step 1 was updated to use key pair algorithm) Annex HelperCreateCACertificate (input parameter for key pair algorithm was added, key pair was added as output parameter and public key and private key were removed from output parameters, step 1 was updated to use key pair algorithm, step 2 was updated

		to use Annex HelperGenerateKeyPair, step 3 was updated to use key pair) Annex HelperDeleteCertificationPath (title was updated, pre-requisite was updated to use both RSA and ECC) Annex HelperDetermineRSAKeyLength was removed and replaced with Annex HelperDetermineKeyPairGenerationParams Annex HelperCreateRSAKeyPair removed and replaced with Annex HelperCreateKeyPair Annex HelperCreateSelfSignedCertificate (pre-requisite was updated to use both RSA and ECC, input parameter for key pair algorithm was added, step 1 was updated to use Annex HelperCreateKeyPair, step 2 was updated to use key pair algorithm) Annex HelperDeleteCertWithKey (title was updated, pre-requisite was updated to use both RSA and ECC) Annex HelperCreateCertificationPath_SelfSigned (pre-requisite was updated to use both RSA and ECC, input parameter for key pair algorithm was added, step 1 was updated to use key pair algorithm) Annex HelperRemoveServerCertificateAssignment (title was updated, pre-requisite was updated to use both RSA and ECC) Annex HelperAddServerCertAssign_SSCertificate (title was updated, pre-requisite was updated to use both RSA and ECC, input parameter for key pair algorithm was added, step 1 was updated to use key pair algorithm) Annex HelperCreateCASignedCertificate (title was updated, input parameter for CSR key pair algorithm was added, input parameter *privateKey*was renamed to *caPrivateKey* step 1 was updated to use Annex HelperCreateKeyPair, step 2 was updated to use CSR key pair algorithm) Annex HelperUploadCertificate (pre-requisite was updated to use both RSA and ECC) Annex HelperUploadCASignedCertificate (pre-requisite was updated to use both RSA and ECC, input parameter for Certificate key pair algorithm was added, input parameter *privateKey* was renamed to *caPrivateKey* step 1 was updated to use Certificate key pair algorithm) Annex HelperDeleteCertificationPath2 (title was updated, pre-requisite was updated to use both RSA and ECC) Annex HelperCreateCertificationPath_CACertificates (pre-requisite was updated to use both RSA and ECC, input parameter for Certificate key pair algorithm was added, input parameter for CA key pair algorithm was added, step 1 was updated to use key pair as output parameter instead of private key, step 2 was updated to use key pair instead of private key and to use Certificate key algorithm) Annex HelperRemoveServerCertificateAssignment2 (title was updated, pre-requisite was updated to use both RSA and ECC) Annex HelperCreateExpiredCACertificate (input parameter for key pair algorithm was added, step 1 was replaced by step 2 which was updated to use key pair algorithm, step 2 was updated to use Annex HelperGenerateKeyPair with capabilities and key pair algorithm as

		input parameters and key pair as output parameter, step 4 was updated to use generated key pair) Annex HelperCreatePKCS8WithNewKeyPair (input parameters for capabilities and key pair algorithm was added, step 1 was updated to use Annex HelperGenerateKeyPair with capabilities and key pair algorithm as input parameters and key pair as output parameter, step 2 was updated to use key pair) Annex HelperGenerateRSAKeyPair was removed and replaced with Annex HelperGenerateKeyPair Annex HelperCreatePKCS8WithExistingKeyPair (public key and private key input parameters was replaced by key pair input parameter, step 1 was updated to use key pair algorithm, private key and public key from the key pair) Annex HelperCreatePKCS8WithNewKeyPairWithPassphrase (input parameters for capabilities and key pair algorithm were added, public key and private key output parameters was replaced by key pair output parameter, step 1 was updated to use Annex HelperGenerateKeyPair with capabilities and key pair algorithm as input parameters and key pair as output parameter, step 2 was updated to use key pair instead of public key and private key) Annex HelperCreatePKCS8WithExistingKeyPairWithPassphrase (public key and private key input parameters was replaced by key pair input parameter, step 1 was updated to use key pair algorithm, private key and public key from the key pair) Annex HelperCreatePKCS12WithNewCACertWithPassphrase (input parameter for key pair algorithm was added, step 1 was updated to use key pair algorithm and key pair, step 2 was updated to use key pair instead of private key and public key) Annex HelperCreatePKCS12WithPassphrase (public key and private key input parameters was replaced by key pair input parameter, step 1 was updated to use key pair algorithm, private key and public key from the key pair) Annex HelperDeleteCertificationPath3 (title was updated, pre-requisite was updated to use both RSA and ECC) Annex HelperUploadPKCS12 (pre-requisite was updated to use both RSA and ECC, input parameter for key pair algorithm was added, step 2 was updated to use key pair algorithm) Annex HelperCreateCRL (input parameter for key pair algorithm was added, step 1 was updated to use key pair algorithm) Annex HelperPrepareCertificate (pre-requisite was updated to use both RSA and ECC, input parameter for key pair algorithm was added, steps 1, 2, 3 and 4 were updated to use key pair algorithm) Annex HelperCreateCertPathValidationPolicy (input parameter for key pair algorithm was added, step 1 was updated to use key pair algorithm) Annex HelperCreateSignedCertificate (input parameter for key pair algorithm was added, step 1 was added to use Annex HelperGenerateKeyPair with capabilities and key pair algorithm as input parameters and key pair as output parameter, step 2 was updated to use key pair algorithm of the private key, step 3 was updated to use key pair and signature algorithm)

		Annex HelperCreateCRLForCertificate (step 1 was updated to use key pair algorithm of the private key, step 2 was updated to use signature algorithm) Annex HelperCheckAndConfigureHTTPS (optional input parameters for Certificate key pair algorithm and CA key pair algorithm were added, step 5 was updated to use Certificate key pair algorithm and CA key pair algorithm) Annex HelperConfigureHTTPS (input parameters for Certificate key pair algorithm and CA key pair algorithm were added, step 4.1 was updated to use Certificate key pair algorithm, step 5 was updated to use Certificate key pair algorithm and CA key pair algorithm) Annex HelperAddServerCertAssign_CACertificate (title was updated, input parameters for Certificate key pair algorithm and CA key pair algorithm were added, step 1 was updated to use Annex HelperCreateKeyPair with capabilities and Certificate key pair algorithm as input parameters and key pair ID as output parameter, step 2 was updated to use CA key pair algorithm, step 5 was updated to use CA key pair algorithm) Annex HelperSignatureAlgorithmSelection (input parameter for key pair algorithm was added, step 1 was updated to algorithm selection for RSA key pair algorithm, step 2 was added to algorithm selection for ECC, step 3 was added) Annex HelperGenerateKeyPair (new annex) Summary: For almost all annexes key pair algorithm was added as input parameter. Almost all annexes are updated to work with both RSA and ECC key pair algorithms. For almost all annexes key pair was added as output parameter and public with private keys are removed from output parameters.
25.06	Dec 6, 2024	The following test cases were updated to use updated annexes and with pre-requisites was updated to use new capabilities according #271: ADVANCED_SECURITY-1-1-1 Create RSA Key Pair, status through polling (steps 4.3.5, 4.4 and 4.5 were updated) ADVANCED_SECURITY-1-1-2 Create RSA Key Pair, status through event (steps 6.3.5, 6.4 and 6.5 were updated) ADVANCED_SECURITY-1-1-3 Check private Key status for an RSA private key (step 3 was added, step 4 was updated to use capabilities and RSA key pair algorithm, step 8 was updated) ADVANCED_SECURITY-1-1-4 Get all keys (step 3 was added, step 6 was updated to use capabilities and RSA key pair algorithm, step 11 was updated) ADVANCED_SECURITY-1-1-5 Delete Key (step 3 was added, step 6 was updated to use capabilities and RSA key pair algorithm, steps 9 and 10 were updated) ADVANCED_SECURITY-2-1-1 Create PKCS#10 certification requests (steps 3 and 4 were updated to use RSA key pair algorithm)

		ADVANCED_SECURITY-2-1-2 Create self-signed certificate (steps 3 and 4 were updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-3 Upload certificate – Keystore contains private key (steps 4 and 5 were updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-4 Upload certificate – Keystore contains private key (negative test) (step 4 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-5 Upload certificate – Keystore does not contain private key (step 4 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-6 Get certificate – self-signed (step 4 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-7 Get certificate – CA (step 4 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-8 Get all certificates – self signed (step 6 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-9 Get All Certificate – CA (step 6 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-10 Delete Certificate – self signed (step 6 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-11 Delete Certificate – CA (step 6 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-12 Create Certification Path – self-signed (step 4 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-13 Create Certification Path – CA (step 4 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-14 Get Certification Path – self-signed (step 4 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-15 Get Certification Path – CA (step 4 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-16 Get All Certification Paths – self-signed (step 6 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-17 Get All Certification Paths – CA (step 6 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-18 Delete Certification Path – self-signed (step 6 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-19 Delete Certification Path - CA (step 6 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-20 CreatePKCS10CSR – negative test (step 10 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-21 Delete Certificate – CA – Preserve Public Key (step 4 was updated to use RSA key pair algorithm)

		ADVANCED_SECURITY-2-1-22 Upload certificate – delete linked key (negative test) (steps 4 and 5 were updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-23 Upload certificate – Upload malformed certificate (negative test) (step 4 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-24 Upload certificate – Upload expired certificate (step 4 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-27 CreateSelfSignedCertificate with PKCS#12 (steps 4 and 5 were updated to use RSA key pair algorithm) ADVANCED_SECURITY-2-1-28 Create PKCS#10 request with PKCS#12 (steps 4 and 5 were updated to use RSA key pair algorithm) ADVANCED_SECURITY-3-1-1 Add Server Certificate Assignment – self-signed (step 5 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-3-1-2 Add Server Certificate Assignment – CA (step 5 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-3-1-3 Replace Server Certificate Assignment – self-signed (steps 5 and 6 were updated to use RSA key pair algorithm) ADVANCED_SECURITY-3-1-4 Replace Server Certificate Assignment – CA (steps 5 and 7 were updated to use RSA key pair algorithm) ADVANCED_SECURITY-3-1-5 Get Assigned Server Certificates – self-signed (step 5 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-3-1-6 Get Assigned Server Certificates – CA (step 5 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-3-1-7 Remove Server Certificate Assignment – self-signed (step 5 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-3-1-8 Remove Server Certificate Assignment – CA (step 5 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-3-2-3 Basic TLS Handshake (step 8 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-3-2-4 Basic TLS Handshake after Replace Server Certificate Assignment (steps 8 and 17 were updated to use RSA key pair algorithm) ADVANCED_SECURITY-3-2-5 Basic TLS Handshake with Replace Server Certification Path and PKCS#12 (steps 10 and 24 were updated to use RSA key pair algorithm) ADVANCED_SECURITY-3-3-1 TLS client authentication – self-signed TLS server certificate with on-device RSA key pair (steps 11, 12, 14, 15 and 16 were updated to use RSA key pair algorithm) ADVANCED_SECURITY-3-3-2 CRL processing with on-device RSA key pair (steps 11, 12, 14 and 15 were updated to use RSA key pair algorithm)

		ADVANCED_SECURITY-3-3-3 Replace certification path validation policy assignment (steps 4 and 5 were updated to use RSA key pair algorithm) ADVANCED_SECURITY-4-1-1 TLS Server Certificate - self-signed (step 5 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-4-1-2 TLS Server Certificate – CA (step 5 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-5-1-1 Security Configuration Service Capabilities (step 8 was added to check new ECCKeyPairGeneration capability, step 10 was added to check new PKCS8 capability, step 12 was added to check new PKCS12 capability, step 11 was added to check new PKCS10 capability, step 16 was added to check new SelfSignedCertificateCreation capability) ADVANCED_SECURITY-5-1-2 Get Services and Get Security Configuration Service Capabilities Consistency (was updated with new capabilities) ADVANCED_SECURITY-6-2-1 Upload PKCS8 – no key pair exists (step 3 was added, step 4 was updated to use capabilities and RSA key pair algorithm) ADVANCED_SECURITY-6-2-3 Upload Encrypted PKCS8 (step 3 was added, step 5 was updated to use capabilities and RSA key pair algorithm) ADVANCED_SECURITY-6-2-4 Upload Encrypted PKCS8 – Using Passphrase Management (step 3 was added, step 5 was updated to use capabilities and RSA key pair algorithm) ADVANCED_SECURITY-6-3-1 Upload PKCS12 – no key pair exists (step 5 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-6-3-4 Upload PKCS12 - verify key and certificate (step 5 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-6-3-5 Upload PKCS12 – Decryption Failed (step 5 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-6-3-6 Upload PKCS12 – Using Passphrase Storage (step 5 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-8-1-1 Upload CRL (step 4 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-8-1-2 Delete CRL (step 4 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-8-1-3 Get CRL (step 4 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-8-1-4 Create certification path validation policy (step 4 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-8-1-5 Get certification path validation policy (step 4 was updated to use RSA key pair algorithm) ADVANCED_SECURITY-8-1-6 Delete certification path validation policy (step 4 was updated to use RSA key pair algorithm)

		Summary: Almost all test cases was updated to use updated annexes and updated capabilities.
25.06	Dec 9, 2024	The following new testcases were added for ECC key pair algorithm according #271: ADVANCED_SECURITY-1-1-6 Create ECC Key Pair, status through polling ADVANCED_SECURITY-1-1-7 Create ECC Key Pair, status through event ADVANCED_SECURITY-1-1-8 Check private Key status for an ECC private key ADVANCED_SECURITY-1-1-9 Get all keys (ECC) ADVANCED_SECURITY-1-1-10 Delete Key (ECC) ADVANCED_SECURITY-2-1-29 Create PKCS#10 certification requests (ECC) ADVANCED_SECURITY-2-1-30 Create self-signed certificate (ECC) ADVANCED_SECURITY-2-1-31 Upload certificate – Keystore contains private key (ECC) ADVANCED_SECURITY-2-1-32 Upload certificate – Keystore contains private key (ECC, negative test) ADVANCED_SECURITY-2-1-33 Upload certificate – Keystore does not contain private key (ECC) ADVANCED_SECURITY-2-1-34 Get certificate – self-signed (ECC) ADVANCED_SECURITY-2-1-35 Get certificate – CA (ECC) ADVANCED_SECURITY-2-1-36 CreatePKCS10CSR – negative test (ECC) ADVANCED_SECURITY-2-1-37 Delete Certificate – CA – Preserve Public Key (ECC) ADVANCED_SECURITY-2-1-38 Upload certificate – delete linked key (ECC, negative test) ADVANCED_SECURITY-2-1-39 Upload certificate – Upload malformed certificate (ECC, negative test) ADVANCED_SECURITY-2-1-40 Upload certificate – Upload expired certificate (ECC) ADVANCED_SECURITY-2-1-41 CreateSelfSignedCertificate with PKCS#12 (ECC) ADVANCED_SECURITY-2-1-42 Create PKCS#10 request with PKCS#12 (ECC)
25.06	Mar 04, 2025	The following new test cases were added as a part of #301 task: ADVANCED_SECURITY-10-1-1 Authorization Server Configuration Creation (new)

		ADVANCED_SECURITY-10-1-2 Authorization Server Configuration Modification (new) Annex HelperGetAuthorizationServerConfigurations Get Authorization Server Configurations List (new) Annex HelperEmptySpaceForOneAuthorizationServerConfiguration Make Sure That At Least One Authorization Server Configuration Could Be Created (new) Annex HelperConfigureKeyPair Configure Key Pair (new) Annex HelperCreateCertPathValidationPolicyForAuthServer Create a certification path validation policy for authentication server configuration (new) Introduction\Scope\Authorization Server Configuration (new) Test Overview\Test Policy\Authorization Server Configuration (new)
25.06	Jun 5, 2025	The following test cases were removed according #341: ADVANCED_SECURITY-2-1-27 CreateSelfSignedCertificate with PKCS#12 ADVANCED_SECURITY-2-1-28 Create PKCS#10 request with PKCS#12 ADVANCED_SECURITY-2-1-41 CreateSelfSignedCertificate with PKCS#12 (ECC) ADVANCED_SECURITY-2-1-42 Create PKCS#10 request with PKCS#12 (ECC)

Table of Contents

1 Introduction

The goal of the ONVIF test specification set is to make it possible to realize fully interoperable IP physical security 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 Security Configuration Device Test Specification acts as a supplementary document to the [ONVIF Network Interface 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 Security Configuration Device 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 Security Configuration Service, which is defined in [ONVIF Security Configuration Service].

The principal intended purposes are:

- Provide self-assessment tool for implementations.
- Provide comprehensive test suite coverage for [ONVIF Network Interface 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 Network Interface Specs]; instead, it will cover its subset.

This ONVIF Security Configuration Device Test Specification covers the ONVIF Security Configuration Service, which is a functional block of [ONVIF Network Interface 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
- Create ECC 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 5958] 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

1.1.8 TLS Versions

The TLS Versions section covers the test cases needed for disabling and enabling TLS versions on an ONVIF device.

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

- Get list of Enabled TLS Versions
- Set list of Enabled TLS Versions

1.1.9 Authorization Server Configuration

The Authorization Server Configuration section covers the test cases needed for authorization server configuration management on an ONVIF device.

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

- Get list of Authorization Server Configurations
- Create Authorization Server Configuration
- Delete Authorization Server Configuration
- Modify Authorization Server Configuration

2 Normative references

- [ONVIF Conformance] ONVIF Conformance Process Specification:

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

- [ONVIF Profile Policy] ONVIF Profile Policy:

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

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

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

- [ONVIF Core Specs] ONVIF Core Specifications:

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

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

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

- [ONVIF Base Test] ONVIF Base Device Test Specifications:

  https://www.onvif.org/profiles/conformance/device-test/

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

  http://www.iso.org/directives

- [ISO 16484-5] ISO 16484-5:2014-09 Annex P:

  https://www.iso.org/obp/ui/#!iso:std:63753:en

- [SOAP 1.2, Part 1] W3C SOAP 1.2, Part 1, Messaging Framework:

  http://www.w3.org/TR/soap12-part1/

- [XML-Schema, Part 1] W3C XML Schema Part 1: Structures Second Edition:

  http://www.w3.org/TR/xmlschema-1/

- [XML-Schema, Part 2] W3C XML Schema Part 2: Datatypes Second Edition:

  http://www.w3.org/TR/xmlschema-2/

- [WS-Security] "Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)", OASIS Standard, February 2006.:

http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

- [RFC 3447] "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", J. Jonsson, B. Kaliski, February 2003.:

  https://www.ietf.org/rfc/rfc3447.txt

- [RFC 5280] "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", D. Cooper et. al., May 2008.:

  http://www.ietf.org/rfc/rfc5280.txt

- [RFC 5958] "Asymmetric Key Packages", S. Turner, August 2010.:

  https://tools.ietf.org/html/rfc5958

- [RFC 5959] "Algorithms for Asymmetric Key Package Content Type", S. Turner, August 2010.:

  https://www.ietf.org/rfc/rfc5959.txt

- [PKCS#12] "Personal Information Exchange Syntax v1.0", RSA Laboratories, June 24, 1999.:

3 Terms and Definitions

3.1 Conventions

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

3.2 Definitions

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

Key	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 both RSA and ECC key pairs as keys.
Key Pair	A key that consists of a public key and (optionally) a private key.
Key Pair Algorithm	A key pair algorithm is used in this specification to indicate to the algorithm of key or key pair. This specification supports both RSA and ECC key pair algorithms.
Key Pair Generation Params	A key pair generation params is used in this specification as representing of generation params for several key pair algorithms. This specification supports both RSA and ECC key pair generation params.
RSA key pair	A key pair that is accepted as input by the RSA algorithm.
ECC key pair	A key pair that is accepted as input by the ECC algorithm.
RSA key pair generation params	A key pair generation params that is accepted as input by the RSA key pair generation algorithm.
ECC key pair generation params	A key pair generation params that is accepted as input by the ECC key pair generation algorithm.
Digital Signature	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.
Certificate	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.
Certification Path	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.
Certification Authority	A certification authority is an entity that issues certificates to subject entities.

Alias	An alias is a name for an object on the device that is chosen by the client and treated transparently by the device.

3.3 Abbreviations

This section describes abbreviations used in this document.

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

4 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.

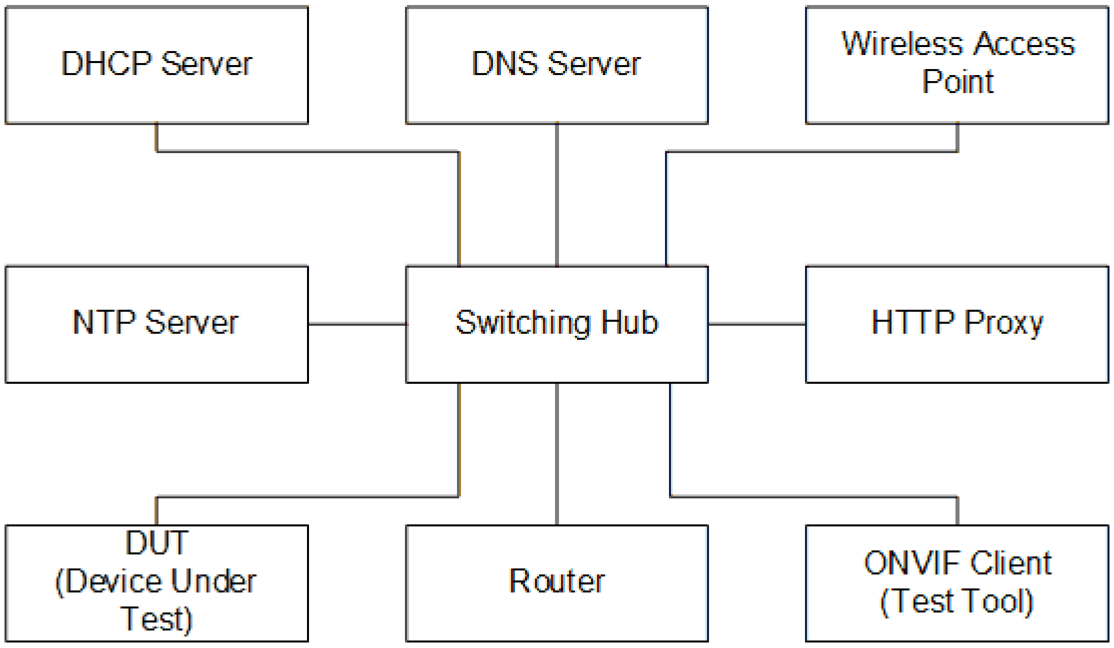
4.1 Test Setup

4.1.1 Network Configuration for DUT

The generic test configuration for the execution of test cases defined in this document is as shown below (Figure 4.1).

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 4.1. 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 behavior of the DUT. It handles both expected and unexpected behavior.

HTTP Proxy: provides facilitation in case of RTP and RTSP tunneling 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.

Switching Hub: provides network connectivity among all the test equipments in the test environment. All devices should be connected to the Switching Hub. When running multiple test instances in parallel on the same network, the Switching Hub should be configured to use filtering in order to avoid multicast traffic being flooded to all ports, because this may affect test stability.

Router: provides router advertisements for IPv6 configuration.

4.2 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).

4.3 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.

4.3.1 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.

4.3.2 Keystore

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

- DUT shall give the ONVIF Security Configuration 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 or ECC 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 generates an ECC key pair status handling is done with polling.
  - The DUT generates an ECC 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 5.1 for Keystore Test Cases.

4.3.3 Certificate Management

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

- DUT shall give the ONVIF Security Configuration 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 5.2 for Certificate Management Test Cases.

4.3.4 TLS Server

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

- DUT shall give the ONVIF Security Configuration 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 5.3 for TLS Server Test Cases.

4.3.5 Referential Integrity

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

- DUT shall give the ONVIF Security Configuration 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 5.4 for Referential integrity Test Cases.

4.3.6 Capabilities

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

- DUT shall give the ONVIF Security Configuration 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 5.5 for Capabilities Test Cases.

4.3.7 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 ONVIF Security Configuration 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 5.6 for Off-Device Key Generation Operations Test Cases.

4.3.8 Certificate-Based Client Authentication

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

- DUT shall give the ONVIF Security Configuration 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 5.7 for Certificate-based Client Authentication Test Cases.

4.3.9 TLS Versions

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

- DUT shall give the ONVIF Security Configuration Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.
- The DUT shall support enabling and disabling specific TLS versions.
- The following tests are performed about enabling and disabling specific TLS versions
    - The ONVIF Client can get list of enabled TLS versions.
    - The ONVIF Client can set list of enabled TLS versions.
    - TLS versions out of enabled TLS versions are rejected by the DUT.

Please, refer to Section 5.8 for TLS Versions Test Cases.

4.3.10 Authorization Server Configuration

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

- DUT shall give the ONVIF Security Configuration Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.

- The DUT shall support Authorization Server Configuration. Otherwise, these test cases will be skipped.
- The DUT shall support certification path validation policy.
- The following tests are performed about Authorization Server Configuration
  - The ONVIF Client can get list of Authorization Server Configurations.
  - The ONVIF Client can get Authorization Server Configuration by provided token.
  - The ONVIF Client can create Authorization Server Configuration with all configuration type supported and all authentication method supported.
  - The ONVIF Client can delete Authorization Server Configuration.
  - The ONVIF Client can modify Authorization Server Configuration.

Please, refer to Section 5.9 for Authorization Server Configuration Test Cases.

5 Security Configuration Test Cases

5.1 Keystore

5.1.1 Create RSA Key Pair, status through polling

Test Case ID: ADVANCED_SECURITY-1-1-1

Specification Coverage: Key Management (ONVIF Security Configuration Service Specification)

Feature under test: CreateRSAKeyPair, GetKeyStatus

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.1.
4. For each key length *keyLength* in the RSAKeyLengths 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 *operationDelay* + *duration* expires repeat the following steps:

   4.3.1. ONVIF Client waits for 5 seconds.

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.2 to restore DUT configuration, and skip other steps.

4.4. If *operationDelay* + *duration* 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.2 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.2 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

5.1.2 Create RSA Key Pair, status through event

Test Case ID: ADVANCED_SECURITY-1-1-2

Specification Coverage: Key Management (ONVIF Security Configuration Service Specification)

Feature under test: CreateRSAKeyPair

WSDL Reference:security.wsdl and event.wsdl

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

Pre-Requisite: Security Configuration 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. Device supports Pull-Point Notification feature.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.1.
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 *operationDelay* + *duration* timeout expires repeat the following steps:

   6.3.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 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.2 to restore DUT configuration and go to the step 7.

6.4. If *operationDelay* + *duration* 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.2 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.2 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 passes 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 the **UnsubscribeResponse** message.

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

5.1.3 Check private Key status for an RSA private key

Test Case ID: ADVANCED_SECURITY-1-1-3

Specification Coverage: Key Management (ONVIF Security Configuration Service Specification)

Feature under test: GetAllKeys

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.1.
4. ONVIF Client creates an RSA key pair by following the procedure mentioned in Annex A.3 with the following input and output parameters
   - in *cap* - service capabilities
   - in RSA - key pair algorithm
   - out *keyID* - key pair ID
5. ONVIF Client invokes **GetAllKeys**.
6. The DUT responds with a **GetAllKeysResponse** message with parameters
   - KeyAttribute list =: *keyAttributeList*
7. If *keyAttributeList*[KeyID = *keyID*].hasPrivateKey is not equal to true, FAIL the test and go to the next step.

8. ONVIF Client deletes the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.2 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.1.4 Get all keys

Test Case ID: ADVANCED_SECURITY-1-1-4

Specification Coverage: Key Management (ONVIF Security Configuration Service Specification)

Feature under test: GetAllKeys

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration 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 Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

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

4. ONVIF Client invokes **GetAllKeys**.

5. The DUT responds with a **GetAllKeysResponse** message with parameters

   - KeyAttribute list =: *initialKeyList*

6. ONVIF Client creates an RSA key pair by following the procedure mentioned in Annex A.3 with the following input and output parameters

- in *cap* - service capabilities
- in RSA - key pair algorithm
- out *keyID* - key pair ID

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

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Note: The DUT may return an empty list at step 5.

5.1.5 Delete Key

Test Case ID: ADVANCED_SECURITY-1-1-5

Specification Coverage: Key Management (ONVIF Security Configuration Service Specification)

Feature under test: DeleteKey

WSDL Reference: security.wsdl

Test Purpose: To test deletion of RSA key pairs

Pre-Requisite: Security Configuration 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.1.
4. ONVIF Client invokes **GetAllKeys**.
5. The DUT responds with a **GetAllKeysResponse** message with parameters
   - KeyAttribute list =: *initialKeyList*
6. ONVIF Client creates an RSA key pair by following the procedure mentioned in Annex A.3 with the following input and output parameters
   - in *cap* - service capabilities
   - in RSA - key pair algorithm
   - out *keyID* - key pair ID
7. ONVIF Client invokes **GetAllKeys**.
8. The DUT responds with a **GetAllKeysResponse** message with parameters
   - KeyAttribute list =: *updatedKeyList*
9. 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.2 to restore DUT configuration, and skip other steps.
10. 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.2 to restore DUT configuration, and skip other steps.
11. ONVIF Client invokes **DeleteKey** with parameters
    - KeyID =: *keyID*
12. The DUT responds with a **DeleteKeyResponse** message.
13. ONVIF Client invokes **GetAllKeys**.
14. The DUT responds with a **GetAllKeysResponse** message with parameters

- KeyAttribute list =: *finalKeyList*

15. If *finalKeyList* is not equal *initialKeyList*, FAIL the test.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **DeleteKeyResponse** message.
- DUT did not send **GetAllKeysResponse** message(s).

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

5.1.6 Create ECC Key Pair, status through polling

Test Case ID: ADVANCED_SECURITY-1-1-6

Specification Coverage: Key Management (ONVIF Security Configuration Service Specification)

Feature under test: CreateECCKeyPair, GetKeyStatus

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. On-board ECC key pair generation is supported by the DUT as indicated by the ECCKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional ECC key pair.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.1.
4. For each elliptic curve *ellipticCurve* in the EllipticCurves capability contained in *cap*.KeystoreCapabilities repeat the following steps:

4.1. ONVIF Client invokes **CreateECCKeyPair** with parameter

- EllipticCurve := *ellipticCurve*

4.2. The DUT responds with **CreateECCKeyPairResponse** message with parameters

- KeyID =: *keyID*
- EstimatedCreationTime =: *duration*

4.3. Until *operationDelay* + *duration* expires repeat the following steps:

4.3.1. ONVIF Client waits for 5 seconds.

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 ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.2 to restore DUT configuration, and skip other steps.

4.4. If *operationDelay* + *duration* timeout expires for step 4.3 and the last *keyStatus* is other than "ok", FAIL the test, delete the ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.2 to restore DUT configuration and skip other steps.

4.5. ONVIF Client deletes the ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.2 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

5.1.7 Create ECC Key Pair, status through event

Test Case ID: ADVANCED_SECURITY-1-1-7

Specification Coverage: Key Management (ONVIF Security Configuration Service Specification)

Feature under test: CreateECCKeyPair

WSDL Reference:security.wsdl and event.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Event Service was received from the DUT. On-board ECC key pair generation is supported by the DUT as indicated by the ECCKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional ECC key pair. Device supports Pull-Point Notification feature.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

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

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 elliptic curve *ellipticCurve* in the EllipticCurves capability contained in *cap*.KeystoreCapabilities repeat the following steps:

   6.1. ONVIF Client invokes **CreateECCKeyPair** with parameter

- EllipticCurve := *ellipticCurve*

6.2. The DUT responds with **CreateECCKeyPairResponse** message with parameters

- KeyID =: *keyID*
- EstimatedCreationTime =: *duration*

6.3. Until *operationDelay* + *duration* timeout expires repeat the following steps:

6.3.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 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 ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.2 to restore DUT configuration and go to the step 7.

6.4. If *operationDelay* + *duration* 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 ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.2 to restore DUT configuration and go to the step 7.

6.5. ONVIF Client deletes the ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.2 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 passes all assertions.

FAIL –

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **CreateECCKeyPairResponse** message(s).
- The DUT did not send **PullMessagesResponse** message(s).
- The DUT did not send the **UnsubscribeResponse** message.

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

5.1.8 Check private Key status for an ECC private key

Test Case ID: ADVANCED_SECURITY-1-1-8

Specification Coverage: Key Management (ONVIF Security Configuration Service Specification)

Feature under test: GetAllKeys

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. On-board ECC key pair generation is supported by the DUT as indicated by the ECCKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional ECC key pair.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.1.
4. ONVIF Client creates an ECC key pair by following the procedure mentioned in Annex A.3 with the following input and output parameters

- in *cap* - service capabilities
- in ECC - key pair algorithm
- out *keyID* - key pair ID

5. ONVIF Client invokes **GetAllKeys**.
6. The DUT responds with a **GetAllKeysResponse** message with parameters
   - KeyAttribute list =: *keyAttributeList*
7. If *keyAttributeList*[KeyID = *keyID*].hasPrivateKey is not equal to true, FAIL the test and go to the next step.
8. ONVIF Client deletes the ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.2 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.1.9 Get all keys (ECC)

Test Case ID: ADVANCED_SECURITY-1-1-9

Specification Coverage: Key Management (ONVIF Security Configuration Service Specification)

Feature under test: GetAllKeys

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. On-board ECC key pair generation is supported by the DUT as indicated by the ECCKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional ECC key pair.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.1.
4. ONVIF Client invokes **GetAllKeys**.
5. The DUT responds with a **GetAllKeysResponse** message with parameters
    - KeyAttribute list =: *initialKeyList*
6. ONVIF Client creates an ECC key pair by following the procedure mentioned in Annex A.3 with the following input and output parameters
    - in *cap* - service capabilities
    - in ECC - key pair algorithm
    - out *keyID* - key pair ID
7. ONVIF Client invokes **GetAllKeys**.
8. The DUT responds with a **GetAllKeysResponse** message with parameters
    - KeyAttribute list =: *updatedKeyList*
9. If *updatedKeyList* does not contain *keyID* and all keys from *initialKeyList*, FAIL the test, and go to the step 11.
10. If *updatedCertificateList* contains keys other than *keyID* or keys from *initialCertificateList*, FAIL the test, and go to the step 11.
11. ONVIF Client deletes the ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.2 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Note: The DUT may return an empty list at step 5.

5.1.10 Delete Key (ECC)

Test Case ID: ADVANCED_SECURITY-1-1-10

Specification Coverage: Key Management (ONVIF Security Configuration Service Specification)

Feature under test: DeleteKey

WSDL Reference: security.wsdl

Test Purpose: To test deletion of ECC key pairs

Pre-Requisite: Security Configuration Service is received from the DUT. On-board ECC key pair generation is supported by the DUT as indicated by the ECCKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional ECC key pair.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.1.
4. ONVIF Client invokes **GetAllKeys**.
5. The DUT responds with a **GetAllKeysResponse** message with parameters
   - KeyAttribute list =: *initialKeyList*
6. ONVIF Client creates an ECC key pair by following the procedure mentioned in Annex A.3 with the following input and output parameters
   - in *cap* - service capabilities
   - in ECC - key pair algorithm
   - out *keyID* - key pair ID
7. ONVIF Client invokes **GetAllKeys**.
8. The DUT responds with a **GetAllKeysResponse** message with parameters
   - KeyAttribute list =: *updatedKeyList*

9. If *updatedKeyList* does not contain *keyID* and all keys from *initialKeyList*, FAIL the test, delete the ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.2 to restore DUT configuration, and skip other steps.

10. If *updatedKeyList* contains keys other than *keyID* or keys from *initialKeyList*, FAIL the test, and delete the ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.2 to restore DUT configuration, and skip other steps.

11. ONVIF Client invokes **DeleteKey** with parameters

    - KeyID =: *keyID*

12. The DUT responds with a **DeleteKeyResponse** message.

13. ONVIF Client invokes **GetAllKeys**.

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

    - KeyAttribute list =: *finalKeyList*

15. If *finalKeyList* is not equal *initialKeyList*, FAIL the test.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

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

5.2 Certificate Management

5.2.1 Create PKCS#10 certification requests

Test Case ID: ADVANCED_SECURITY-2-1-1

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: CreatePKCS10CSR

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates an RSA key pair by following the procedure mentioned in Annex A.3 with the following input and output parameters
   - in *cap* - service capabilities
   - in RSA - key pair algorithm
   - out *keyID* - key pair ID
4. ONVIF Client selects signature algorithm by following the procedure mentioned in Annex A.5 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in RSA - key pair algorithm
   - out *signatureAlgorithm* - signature algorithm
5. ONVIF Client invokes **CreatePKCS10CSR** with parameters
   - Subject := *subject* (see Annex A.6)
   - KeyID := *keyID*
   - CSRAttribute skipped
   - SignatureAlgorithm.algorithm := *signatureAlgorithm*
6. The DUT responds with a **CreatePKCS10CSRResponse** message with parameters
   - PKCS10CSR =: *PKCS10request*

7. ONVIF Client validates that *PKCS10request* is correctly DER encoded (see 11).

8. If *PKCS10request* is incorrectly DER encoded, FAIL the test and go to the step11.

9. ONVIF Client validates that *PKCS10request* contains the correct subject equals to *subject*.

10. If *PKCS10request* contains a wrong subject, 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.2 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.2 Create self-signed certificate

Test Case ID: ADVANCED_SECURITY-2-1-2

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: CreateSelfSignedCertificate

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or 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 Procedure:

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client creates an RSA key pair by following the procedure mentioned in Annex A.3 with the following input and output parameters
   - in *cap* - service capabilities
   - in RSA - key pair algorithm
   - out *keyID* - key pair ID
4. ONVIF Client selects signature algorithm by following the procedure mentioned in Annex A.5 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in RSA - key pair algorithm
   - out *signatureAlgorithm* - signature algorithm
5. ONVIF Client invokes **CreateSelfSignedCertificate** with parameters
   - X509Version skipped
   - KeyID := *keyID*
   - Subject := *subject* (see Annex A.6)
   - Alias skipped
   - notValidBefore skipped
   - notValidAfter skipped
   - SignatureAlgorithm.algorithm := *signatureAlgorithm*
   - SignatureAlgorithm.parameters skipped
   - SignatureAlgorithm.anyParameters skipped
   - Extension skipped
6. The DUT responds with a **CreateSelfSignedCertificateResponse** message with parameters
   - CertificateID =: *certID*
7. ONVIF Client deletes the self-signed certificate (in *certID*) and related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.3 Upload certificate – Keystore contains private key

Test Case ID: ADVANCED_SECURITY-2-1-3

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadCertificate

WSDL Reference: security.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: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.8 with the following input and output parameters:

- in *cap* - DUT capabilities
- in RSA - key pair algorithm
- out *CAcert* - CA certificate
- out *CAkeyPair* - key pair with private and public keys

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

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.4 Upload certificate – Keystore contains private key (negative test)

Test Case ID: ADVANCED_SECURITY-2-1-4

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadCertificate

WSDL Reference: security.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: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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 Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.8 with the following input and output parameters:

   - in *cap* - DUT capabilities

   - in RSA - key pair algorithm

- out *CAcert* - CA certificate
- out *CAkeyPair* - key pair with public and private keys

5. ONVIF Client invokes UploadCertificate with parameters
   - Certificate := *CAcert*
   - Alias := "ONVIF_Test"
   - PrivateKeyRequired := true
6. The DUT returns env:Receiver/ter:Action/ter:NoMatchingPrivateKey SOAP 1.2 fault.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.5 Upload certificate – Keystore does not contain private key

Test Case ID: ADVANCED_SECURITY-2-1-5

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadCertificate

WSDL Reference: security.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: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.8 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in RSA - key pair algorithm
   - out *CAcert* - CA certificate
   - out *CAkeyPair* - key pair with public and private keys
5. ONVIF Client invokes **UploadCertificate** with parameters
   - Certificate := *CAcert*
   - Alias := "ONVIF_Test"
   - PrivateKeyRequired : = false
6. The DUT responds with a **UploadCertificateResponse** message with parameters
   - CertificateID =: *certID*
   - KeyID =: *keyID*
7. ONVIF Client deletes the CA certificate (in *certID*) and related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.6 Get certificate – self-signed

Test Case ID: ADVANCED_SECURITY-2-1-6

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: GetCertificate

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or 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 Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client creates a self-signed certificate and related RSA key pair by following the procedure mentioned in Annex A.12 with the following input and output parameters:

   - in *cap* - service capabilities
   - in RSA - key pair algorithm
   - out *keyID* - key pair ID
   - out *certID* - certificate ID

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.13).
8. If *X509Cert*.CertificateContent is incorrectly DER encoded, FAIL the test and go to the step 10.
9. ONVIF Client validates that *X509Cert*.CertificateContent contains the correct subject equals to subject defined in Annex A.6.
10. If *X509Cert*.CertificateContent contains wrong subject, FAIL the test and go to the step 10.
11. 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.7 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.7 Get certificate – CA

Test Case ID: ADVANCED_SECURITY-2-1-7

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: GetCertificate

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
    - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.8 with the following input and output parameters:
    - in *cap* - DUT capabilities
    - in RSA - key pair algorithm
    - out *CAcert* - CA certificate
    - out *CAkeyPair* - key pair with public and private keys
5. 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.14.
6. ONVIF Client invokes **GetCertificate** message with parameters
    - CertificateID := *certID*
7. The DUT responds with a **GetCertificateResponse** message with parameters
    - Certificate =: *X509Cert*
8. ONVIF Client validates that *X509Cert*.CertificateContent is correctly DER encoded (see Annex A.13).
9. If *X509Cert*.CertificateContent is incorrectly DER encoded, FAIL the test and go to the step 10.
10. If *X509Cert*.CertificateContent contains wrong subject, 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.7 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.8 Get all certificates – self signed

Test Case ID: ADVANCED_SECURITY-2-1-8

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: GetAllCertificates

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client invokes **GetAllCertificates**.
5. The DUT responds with a **GetAllCertificatesResponse** message with parameters
   - CertificateID list =: *initialCertificateList*

6. ONVIF Client creates a self-signed certificate and related RSA key pair by following the procedure mentioned in Annex A.12 with the following input and output parameters:
   - in *cap* - service capabilities
   - in RSA - key pair algorithm
   - out *keyID* - key pair ID
   - out *certID* - certificate ID
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 self-signed certificate (in *certID*) and related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Note: The DUT may return an empty list at step 5.

5.2.9 Get All Certificate – CA

Test Case ID: ADVANCED_SECURITY-2-1-9

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: GetAllCertificates

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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 Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client invokes **GetAllCertificates**.

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

   - CertificateID list =: *initialCertificateList*

6. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.8 with the following input and output parameters:

   - in *cap* - DUT capabilities

   - in RSA - key pair algorithm

   - out *CAcert* - CA certificate

   - out *CAkeyPair* - key pair with private and public keys

7. 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.14.

8. ONVIF Client invokes **GetAllCertificates**.

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

- CertificateID list =: *updatedCertificateList*

10. If *updatedCertificateList* does not contain *certID* and all certificates from *initialCertificateList*, FAIL the test, and go to the step 10.

11. If *updatedCertificateList* contains certificates other than *certID* or certificates from *initialCertificateList*, FAIL the test, and go to the step 10.

12. ONVIF Client deletes the CA certificate (in *certID*) and related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

5.2.10 Delete Certificate – self signed

Test Case ID: ADVANCED_SECURITY-2-1-10

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: DeleteCertificate

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or 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 Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client invokes **GetAllCertificates**.

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

   - CertificateID list =: *initialCertificateList*

6. ONVIF Client creates a self-signed certificate and related RSA key pair by following the procedure mentioned in Annex A.12 with the following input and output parameters:

   - in *cap* - service capabilities

   - in RSA - key pair algorithm

   - out *keyID* - key pair ID

   - out *certID* - certificate ID

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 self-signed certificate (in *certID*) and related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.7 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 self-signed certificate (in *certID*) and related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.7 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.2 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 –

- DUT passes all assertions.

FAIL –

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

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

Note: The DUT may return an empty list at step 5.

5.2.11 Delete Certificate – CA

Test Case ID: ADVANCED_SECURITY-2-1-11

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: DeleteCertificate

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client invokes **GetAllCertificates**.
5. The DUT responds with a **GetAllCertificatesResponse** message with parameters
   - CertificateID list =: *initialCertificateList*
6. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.8 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in RSA - key pair algorithm
   - out *CAcert* - CA certificate
   - out *CAkeyPair* - key pair with public and private keys
7. 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.14.
8. ONVIF Client invokes **GetAllCertificates**.
9. The DUT responds with a **GetAllCertificatesResponse** message with parameters
   - CertificateID list =: *updatedCertificateList*
10. If *updatedCertificateList* does not contain *certID* and all certificates from *initialCertificateList*, FAIL the test, delete the CA certificate (in *certID*) and related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.7 to restore DUT configuration, and skip other steps.
11. If *updatedCertificateList* contains certificates other than *certID* or certificates from *initialCertificateList*, FAIL the test, delete the CA certificate (in *certID*) and related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.7 to restore DUT configuration, and skip other steps.

12. ONVIF Client invokes **DeleteCertificate** with parameters
    - CertificateID =: *certID*
13. The DUT responds with a **DeleteCertificateResponse** message.
14. ONVIF Client deletes the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.2 to restore DUT configuration.
15. ONVIF Client invokes **GetAllCertificates**.
16. The DUT responds with a **GetAllCertificatesResponse** message with parameters
    - CertificateID list =: *finalCertificateList*
17. If *finalCertificateList* is not equal *initialCertificateList*, FAIL the test.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **DeleteCertificateResponse** message.
- DUT did not send **GetAllCertificatesResponse** message(s).

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

5.2.12 Create Certification Path – self-signed

Test Case ID: ADVANCED_SECURITY-2-1-12

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: CreateCertificationPath

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or

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 certificate. The DUT shall have enough free storage capacity for one additional certification path.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
    - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client creates a self-signed certificate and related RSA key pair by following the procedure mentioned in Annex A.12 with the following input and output parameters:
    - in *cap* - service capabilities
    - in RSA - key pair algorithm
    - out *keyID* - key pair ID
    - out *certID* - certificate ID
5. ONVIF Client invokes **CreateCertificationPath** with parameters
    - CertficateIDs.CertificateID[0] := *certID*
    - Alias := "ONVIF_Test"
6. The DUT responds with a **CreateCertificationPathResponse** message with parameters
    - CertificationPathID =: *certPathID*
7. 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.15 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.13 Create Certification Path – CA

Test Case ID: ADVANCED_SECURITY-2-1-13

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: CreateCertificationPath

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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 Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.8 with the following input and output parameters:

   - in *cap* - DUT capabilities

- in RSA - key pair algorithm
- out *CAcert* - CA certificate
- out *CAkeyPair* - key pair with public and private keys

5. ONVIF Client creates and uploads a CA-signed certificate for RSA key pair and associated CA certificate and a corresponding by following the procedure described in Annex A.16 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *CAcert* - CA certificate
   - in *CAkeyPair*.privateKey - CA certificate private key
   - out *certID1* - CA-signed certificate
   - out *keyID1* - RSA key pair
6. 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.14.
7. ONVIF Client invokes **CreateCertificationPath** with parameters
   - CertficateIDs.CertificateID[0] =: *certID1*
   - CertficateIDs.CertificateID[1] =: *certID2*
   - Alias := "ONVIF_Test2"
8. The DUT responds with a **CreateCertificationPathResponse** message with parameters
   - CertificationPathID =: *certPathID*
9. 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 –

- DUT passes all assertions.

FAIL –

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

5.2.14 Get Certification Path – self-signed

Test Case ID: ADVANCED_SECURITY-2-1-14

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: GetCertificationPath

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client creates a certification path based on self-signed certificate and related RSA key pair by following the procedure mentioned in Annex A.18 with the following input and output parameters:
   - in *cap* - service capabilities
   - in RSA - key pair algorithm
   - out *keyID* - key pair ID
   - out *certID1* - certificate ID

- out *certPathID* - certification path ID

5. ONVIF Client invokes **GetCertificationPath** with parameters
   - CertificationPathID =: *certPathID*
6. The DUT responds with a **GetCertificationPathResponse** message with parameters
   - CertificationPath.CertificateID[0] =: *certID2*
   - CertificationPath.Alias
7. If *certID1* is not equal to *certID2*, FAIL the test and go to the step 8.
8. ONVIF Client deletes the certification path (in *certPathID1*), related the self-signed certificate (in *certID1*) and the RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.15 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.15 Get Certification Path – CA

Test Case ID: ADVANCED_SECURITY-2-1-15

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: GetCertificationPath

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client creates a certification path based on CA-signed certificate and related RSA key pair and a corresponding CA certificate and related RSA key pair by following the procedure mentioned in Annex A.19 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in RSA - CA key pair algorithm
   - in RSA - cert key pair algorithm
   - out *certID2* - CA certificate
   - out *keyID2* - CA certificate RSA key pair
   - out *keyID1* - RSA key pair
   - out *certID1* - CA-signed certificate
   - out *certPathID* - certification path
5. ONVIF Client invokes **GetCertificationPath** message with parameters
   - CertificationPathID =: *certPathID*
6. The DUT responds with a **GetCertificationPathResponse** message with parameters
   - CertificationPath.CertificateID[0] =: *certID3*
   - CertificationPath.CertificateID[1] =: *certID4*

- CertificationPath.Alias

7. If *certID1* is not equal to *certID3*, FAIL the test and go to the step 8.

8. If *certID2* is not equal to *certID4*, FAIL the test and go to the step 8.

9. 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 –

- DUT passes all assertions.

FAIL –

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

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

5.2.16 Get All Certification Paths – self-signed

Test Case ID: ADVANCED_SECURITY-2-1-16

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: GetAllCertificationPaths

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client invokes **GetAllCertificationPaths**.
5. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters
   - CertificationPathID list =: *initialCertificationPathList*
6. ONVIF Client creates a certification path based on self-signed certificate and related RSA key pair by following the procedure mentioned in Annex A.18 with the following input and output parameters:
   - in *cap* - service capabilities
   - in RSA - key pair algorithm
   - out *keyID* - key pair ID
   - out *certID1* - certificate ID
   - out *certPathID* - certification path ID
7. ONVIF Client invokes **GetAllCertificationPaths**.
8. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters
   - CertificationPathID list =: *updatedCertificationPathList*
9. If *updatedCertificationPathList* does not contain *certPathID* and all certification paths from *initialCertificationPathList*, FAIL the test, and go to the step 11.
10. If *updatedCertificationPathList* contains certification paths other than *certPathID* or certification paths from *initialCertificationPathList*, FAIL the test, and go to the step 11.
11. ONVIF Client deletes the certification path (in *certPathID*), related the self-signed certificate (in *certID*) and the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.15 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetAllCertificationPathsResponse** message(s).

Note: The DUT may return an empty list at step 5.

5.2.17 Get All Certification Paths – CA

Test Case ID: ADVANCED_SECURITY-2-1-17

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: GetAllCertificationPaths

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
    - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client invokes **GetAllCertificationPaths**.

5. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters
    - CertificationPathID list =: *initialCertificationPathList*
6. ONVIF Client creates a certification path based on CA-signed certificate and related RSA key pair and a corresponding CA certificate and related RSA key pair by following the procedure mentioned in Annex A.19 with the following input and output parameters:
    - in *cap* - DUT capabilities
    - in RSA - CA key pair algorithm
    - in RSA - cert key pair algorithm
    - out *certID2* - CA certificate
    - out *keyID2* - CA certificate RSA key pair
    - out *keyID1* - RSA key pair
    - out *certID1* - CA-signed certificate
    - out *certPathID* - certification path
7. ONVIF Client invokes **GetAllCertificationPaths**.
8. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters
    - CertificationPathID list =: *updatedCertificationPathList*
9. If *updatedCertificationPathList* does not contain *certPathID* and all certification paths from *initialCertificationPathList*, FAIL the test, and go to the step 9.
10. If *updatedCertificationPathList* contains certification paths other than *certPathID* or certification paths from *initialCertificationPathList*, FAIL the test, and go to the step 9.
11. 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 –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetAllCertificationPathsResponse** message(s).

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

5.2.18 Delete Certification Path – self-signed

Test Case ID: ADVANCED_SECURITY-2-1-18

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: DeleteCertificationPath

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client invokes **GetAllCertificationPaths**.
5. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters
   - CertificationPathID list =: *initialCertificationPathList*
6. ONVIF Client creates a certification path based on self-signed certificate and related RSA key pair by following the procedure mentioned in Annex A.18 with the following input and output parameters:

- in *cap* - service capabilities
- in RSA - key pair algorithm
- out *keyID* - key pair ID
- out *certID1* - certificate ID
- out *certPathID* - certification path ID

7. ONVIF Client invokes **GetAllCertificationPaths**.
8. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters
   - CertificationPathID list =: *updatedCertificationPathList*
9. 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 related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.15 to restore DUT configuration, and skip other steps.
10. 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 related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.15 to restore DUT configuration, and skip other steps.
11. ONVIF Client invokes **DeleteCertificationPath** with parameters
    - CertificationPathID =: *certPathID*
12. The DUT responds with a **DeleteCertificationPathResponse** message.
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 –

- DUT passes all assertions.

FAIL –

- DUT did not send **DeleteCertificationResponse** message.
- DUT did not send **GetAllCertificationPathsResponse** message(s).

Note: The DUT may return an empty list at step 5.

5.2.19 Delete Certification Path - CA

Test Case ID: ADVANCED_SECURITY-2-1-19

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: DeleteCertification

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
    - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client invokes **GetAllCertificationPaths**.
5. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters

- CertificationPathID list =: *initialCertificationPathList*

6. ONVIF Client creates a certification path based on CA-signed certificate and related RSA key pair and a corresponding CA certificate and related RSA key pair by following the procedure mentioned in Annex A.19 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in RSA - CA key pair algorithm
   - in RSA - cert key pair algorithm
   - out *certID2* - CA certificate
   - out *keyID2* - CA certificate RSA key pair
   - out *keyID1* - RSA key pair
   - out *certID1* - CA-signed certificate
   - out *certPathID* - certification path
7. ONVIF Client invokes **GetAllCertificationPaths**.
8. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters
   - CertificationPathID list =: *updatedCertificationPathList*
9. 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.
10. 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.
11. ONVIF Client invokes **DeleteCertificationPath** with parameters
    - CertificationPathID =: *certPathID*
12. The DUT responds with a **DeleteCertificationPathResponse** message.
13. ONVIF Client deletes the CA certificate (*certID2*) and related RSA key pair (*keyID2*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.
14. ONVIF Client deletes the CA certificate (*certID1*) and related RSA key pair (*keyID1*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.

15. ONVIF Client invokes **GetAllCertificationPaths**.
16. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters
    - CertificationPathID list =: *finalCertificationPathList*
17. If *finalCertificationPathList* is not equal *initialCertificationPathList*, FAIL the test.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetAllCertificationPathsResponse** message(s).
- DUT did not send **DeleteCertificationPathResponse** message.

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

5.2.20 CreatePKCS10CSR – negative test

Test Case ID: ADVANCED_SECURITY-2-1-20

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: CreatePKCS10CSR

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PKCS#10 supported by the DUT as indicated by the PKCS10 or 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 Procedure:

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client invokes **GetServiceCapabilities**.
4. The DUT responds with **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** message with parameters
   - KeyLength =: *keyLength*
8. The DUT responds with a **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.2.
   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 selects signature algorithm by following the procedure mentioned in Annex A.5 with the following input and output parameters:
    - in *cap* - DUT capabilities
    - in RSA - key pair algorithm
    - out *signatureAlgorithm* - signature algorithm
11. ONVIF Client invokes **CreatePKCS10CSR** with parameters
    - Subject =: *subject* (see Annex A.6)
    - KeyID =: *keyID*

- CSRAttribute skipped
- SignatureAlgorithm.algorithm := *signatureAlgorithm*

12. If the DUT returns env:Sender\ter:InvalidArgVal\ter:InvalidKeyStatus SOAP 1.2 fault, go to step 14.

13. If the DUT returns normal **CreatePKCS10CSRResponse** message.:

    13.1. ONVIF Client deletes RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.2.

    13.2. Set *keyLength* := (the smallest supported RSA key length that is larger than the current *keyLength*)

    13.3. If no such key length exists, log WARNING message, and PASS the test.

    13.4. Go to step 7.

14. ONVIF Client deletes the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.2 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.21 Delete Certificate – CA – Preserve Public Key

Test Case ID: ADVANCED_SECURITY-2-1-21

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: DeleteCertificate

WSDL Reference: security.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: Security Configuration Service is received from the DUT. Create PKCS#10 supported by the DUT as indicated by the PKCS10 or 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.8 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in RSA - key pair algorithm
   - out *CAcert* - CA certificate
   - out *CAkeyPair* - key pair with public and private keys
5. 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.14.
6. ONVIF Client invokes **DeleteCertificate** with parameters
   - CertificateID =: *certID*
7. The DUT responds with a **DeleteCertificateResponse** message.
8. ONVIF Client invokes **GetAllKeys**.
9. The DUT responds with a **GetAllKeysResponse** message where KeyAttribute list contains *keyID*.
10. ONVIF Client deletes the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.2 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The **GetAllKeysResponse** message at Step 8 does not contain *keyID* in KeyAttribute list.
- DUT did not send **DeleteCertificateResponse** message.
- DUT did not send **GetAllKeysResponse** message.

5.2.22 Upload certificate – delete linked key (negative test)

Test Case ID: ADVANCED_SECURITY-2-1-22

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadCertificate

WSDL Reference: security.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: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

- out *cap* - Security Configuration Service Capabilities

4. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.8 with the following input and output parameters:

    - in *cap* - DUT capabilities
    - in RSA - key pair algorithm
    - out *CAcert* - CA certificate
    - out *CAkeyPair* - key pair with public and private keys

5. ONVIF Client creates a certificate from the PKCS#10 request with RSA key pair and associated CA certificate and a corresponding private key by following the procedure described in Annex A.10 with the following input and output parameters:

    - in *cap* - DUT capabilities
    - in *CAcert* - CA certificate
    - in *CAkeyPair*.privateKey - private key
    - in RSA - CSR key pair algorithm
    - out *cert* - certificate
    - out *keyID1* - RSA key pair

6. ONVIF Client invokes **UploadCertificate** with parameters

    - Certificate := *cert*
    - Alias := "ONVIF_Test"
    - PrivateKeyRequired : = true

7. The DUT responds with a **UploadCertificateResponse** message with parameters

    - CertificateID =: *certID*
    - KeyID =: *keyID1*

8. ONVIF Client invokes **DeleteKey** with parameters

    - KeyID =: *keyID1*

9. The DUT returns env:Sender\ter:InvalidArgVal\ter:ReferenceExists SOAP 1.2 fault.

10. ONVIF Client deletes the CA certificate (in *certID*) and related RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send env:Sender\ter:InvalidArgVal\ter:ReferenceExists SOAP 1.2 fault.
- DUT did not send **UploadCertificateResponse** message.
- DUT did not send **DeleteKeyResponse** message.

5.2.23 Upload certificate – Upload malformed certificate (negative test)

Test Case ID: ADVANCED_SECURITY-2-1-23

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadCertificate

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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 Procedure:

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

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.8 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in RSA - key pair algorithm
   - out *CAcert* - CA certificate
   - out *CAkeyPair* - key pair with public and private keys
5. ONVIF Client corrupts *CAcert*.
6. ONVIF Client invokes **UploadCertificate** with parameters
   - Certificate := *CAcert* (malformed)
   - Alias := "ONVIF_Test"
   - PrivateKeyRequired : = false
7. The DUT returns env:Sender\ter:InvalidArgVal\ter:BadCertificate SOAP 1.2 fault.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.24 Upload certificate – Upload expired certificate

Test Case ID: ADVANCED_SECURITY-2-1-24

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadCertificate

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.20 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in RSA - key pair algorithm
   - out *CAcert* - CA certificate
   - out *CAkeyPair* - key pair with public and private keys
5. ONVIF Client invokes **UploadCertificate** with parameters
   - Certificate := *CAcert*
   - Alias := "ONVIF_Test"
   - PrivateKeyRequired : = false
6. The DUT responds with a **UploadCertificateResponse** message with parameters
   - CertificateID =: *certID*

- KeyID =: *keyID*

7. Check that the DUT does not return env:Sender\ter:InvalidArgVal\ter:BadCertificate SOAP 1.2 fault in **UploadCertificateResponse**.

8. ONVIF Client deletes the CA certificate (in *certID*) and related RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

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

5.2.25 Create PKCS#10 certification requests (ECC)

Test Case ID: ADVANCED_SECURITY-2-1-29

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: CreatePKCS10CSR

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 capability. ECC key pair generation supported by the DUT as indicated by the ECCKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional ECC key pair.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client creates an ECC key pair by following the procedure mentioned in Annex A.3 with the following input and output parameters
   - in *cap* - service capabilities
   - in ECC - key pair algorithm
   - out *keyID* - key pair ID
4. ONVIF Client selects signature algorithm by following the procedure mentioned in Annex A.5 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in ECC - key pair algorithm
   - out *signatureAlgorithm* - signature algorithm
5. ONVIF Client invokes **CreatePKCS10CSR** with parameters
   - Subject := *subject* (see Annex A.6)
   - KeyID := *keyID*
   - CSRAttribute skipped
   - SignatureAlgorithm.algorithm := *signatureAlgorithm*
6. The DUT responds with a **CreatePKCS10CSRResponse** message with parameters
   - PKCS10CSR =: *PKCS10request*
7. ONVIF Client validates that *PKCS10request* is correctly DER encoded (see 11).
8. If *PKCS10request* is incorrectly DER encoded, FAIL the test and go to the step11.
9. ONVIF Client validates that *PKCS10request* contains the correct subject equals to *subject*.
10. If *PKCS10request* contains a wrong subject, FAIL the test and go to the step 11.
11. ONVIF Client deletes the ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.2 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.26 Create self-signed certificate (ECC)

Test Case ID: ADVANCED_SECURITY-2-1-30

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: CreateSelfSignedCertificate

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation capability. ECC key pair generation supported by the DUT as indicated by the ECCKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional ECC key pair. The DUT shall have enough free storage capacity for one additional certificate.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates an ECC key pair by following the procedure mentioned in Annex A.3 with the following input and output parameters
   - in *cap* - service capabilities
   - in ECC - key pair algorithm
   - out *keyID* - key pair ID
4. ONVIF Client selects signature algorithm by following the procedure mentioned in Annex A.5 with the following input and output parameters:
   - in *cap* - DUT capabilities

- in ECC - key pair algorithm
- out *signatureAlgorithm* - signature algorithm

5. ONVIF Client invokes **CreateSelfSignedCertificate** with parameters
   - X509Version skipped
   - KeyID := *keyID*
   - Subject := *subject* (see Annex A.6)
   - Alias skipped
   - notValidBefore skipped
   - notValidAfter skipped
   - SignatureAlgorithm.algorithm := *signatureAlgorithm*
   - SignatureAlgorithm.parameters skipped
   - SignatureAlgorithm.anyParameters skipped
   - Extension skipped
6. The DUT responds with a **CreateSelfSignedCertificateResponse** message with parameters
   - CertificateID =: *certID*
7. ONVIF Client deletes the self-signed certificate (in *certID*) and related ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.27 Upload certificate – Keystore contains private key (ECC)

Test Case ID: ADVANCED_SECURITY-2-1-31

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadCertificate

WSDL Reference: security.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: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 capability. ECC key pair generation supported by the DUT as indicated by the ECCKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional ECC 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 Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.8 with the following input and output parameters:

   - in *cap* - DUT capabilities

   - in ECC - key pair algorithm

   - out *CAcert* - CA certificate

   - out *CAkeyPair* - key pair with private and public keys

5. ONVIF Client creates a certificate from the PKCS#10 request with ECC key pair and associated CA certificate and a corresponding private key by following the procedure described in Annex A.10 with the following input and output parameters:

   - in *cap* - DUT capabilities

   - in *CAcert* - CA certificate

- in *CAkeyPair*.privateKey - private key
- in ECC - CSR key pair algorithm
- out *cert* - certificate
- out *keyID1* - ECC key pair

6. ONVIF Client invokes **UploadCertificate** with parameters
   - Certificate := *cert*
   - Alias := "ONVIF_Test"
   - PrivateKeyRequired : = true
7. The DUT responds with a **UploadCertificateResponse** message with parameters
   - CertificateID =: *certID*
   - KeyID =: *keyID2*
8. ONVIF Client validates that *keyID2* equal to *keyID1*.
9. If *keyID2* is not equal to *keyID1*, FAIL the test and go to the step 9.
10. ONVIF Client deletes the CA certificate (in *certID*) and related ECC key pair (in *keyID1*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.28 Upload certificate – Keystore contains private key (ECC, negative test)

Test Case ID: ADVANCED_SECURITY-2-1-32

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadCertificate

WSDL Reference: security.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: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.8 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in ECC - key pair algorithm
   - out *CAcert* - CA certificate
   - out *CAkeyPair* - key pair with public and private keys
5. ONVIF Client invokes UploadCertificate with parameters
   - Certificate := *CAcert*
   - Alias := "ONVIF_Test"
   - PrivateKeyRequired := true
6. The DUT returns env:Receiver/ter:Action/ter:NoMatchingPrivateKey SOAP 1.2 fault.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.29 Upload certificate – Keystore does not contain private key (ECC)

Test Case ID: ADVANCED_SECURITY-2-1-33

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadCertificate

WSDL Reference: security.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: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 capability. The DUT shall have enough free storage capacity for one additional ECC 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 Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

    - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.8 with the following input and output parameters:

    - in *cap* - DUT capabilities

    - in ECC - key pair algorithm

   - out *CAcert* - CA certificate
   - out *CAkeyPair* - key pair with public and private keys

5. ONVIF Client invokes **UploadCertificate** with parameters
   - Certificate := *CAcert*
   - Alias := "ONVIF_Test"
   - PrivateKeyRequired : = false

6. The DUT responds with a **UploadCertificateResponse** message with parameters
   - CertificateID =: *certID*
   - KeyID =: *keyID*

7. ONVIF Client deletes the CA certificate (in *certID*) and related ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.30 Get certificate – self-signed (ECC)

Test Case ID: ADVANCED_SECURITY-2-1-34

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: GetCertificate

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation capability. ECC key pair generation supported by the DUT as indicated by the ECCKeyPairGeneration capability.

The DUT shall have enough free storage capacity for one additional ECC key pair. The DUT shall have enough free storage capacity for one additional certificate.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client creates a self-signed certificate and related ECC key pair by following the procedure mentioned in Annex A.12 with the following input and output parameters:
   - in *cap* - service capabilities
   - in ECC - key pair algorithm
   - out *keyID* - key pair ID
   - out *certID* - certificate ID
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.13).
8. If *X509Cert*.CertificateContent is incorrectly DER encoded, FAIL the test and go to the step 10.
9. ONVIF Client validates that *X509Cert*.CertificateContent contains the correct subject equals to subject defined in Annex A.6.
10. If *X509Cert*.CertificateContent contains wrong subject, FAIL the test and go to the step 10.
11. ONVIF Client deletes the self-signed certificate (in *certID*) and related the ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.31 Get certificate – CA (ECC)

Test Case ID: ADVANCED_SECURITY-2-1-35

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: GetCertificate

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 capability. The DUT shall have enough free storage capacity for one additional ECC 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 Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.8 with the following input and output parameters:

   - in *cap* - DUT capabilities

   - in ECC - key pair algorithm

- out *CAcert* - CA certificate
- out *CAkeyPair* - key pair with public and private keys

5. ONVIF Client uploads a CA certificate (out *certID*, in *CAcert*) and new ECC key pair with the public key from the CA certificate (out *keyID*) by following the procedure described in Annex A.14.
6. ONVIF Client invokes **GetCertificate** message with parameters
   - CertificateID := *certID*
7. The DUT responds with a **GetCertificateResponse** message with parameters
   - Certificate =: *X509Cert*
8. ONVIF Client validates that *X509Cert*.CertificateContent is correctly DER encoded (see Annex A.13).
9. If *X509Cert*.CertificateContent is incorrectly DER encoded, FAIL the test and go to the step 10.
10. If *X509Cert*.CertificateContent contains wrong subject, FAIL the test and go to the step 10.
11. ONVIF Client deletes the CA certificate (in *certID*) and related ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.32 CreatePKCS10CSR – negative test (ECC)

Test Case ID: ADVANCED_SECURITY-2-1-36

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: CreatePKCS10CSR

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PKCS#10 supported by the DUT as indicated by the PKCS10 capability. ECC key pair generation supported by the DUT as indicated by the ECCKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional ECC key pair.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServiceCapabilities**.
4. The DUT responds with **GetServiceCapabilitiesResponse** message with parameters
   - Capabilities =: *cap*
5. Set *ellipticCurve* := (the simplest in the list of supported ECC elliptic curves (cap.KeystoreCapabilities.*EllipticCurves*)).
6. If there is no such elliptic curve, FAIL the test and skip other steps.
7. ONVIF Client invokes **CreateECCKeyPair** message with parameters
   - EllipticCurve =: *ellipticCurve*
8. The DUT responds with a **CreateECCKeyPairResponse** message with parameters
   - KeyID =: *keyID*
   - EstimatedCreationTime =: *duration*
9. If *duration* is less than 2 sec:

   9.1. ONVIF Client deletes ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.2.

   9.2. Set *ellipticCurve* := (the simplest supported ECC elliptic curve that is harder than the current *ellipticCurve*)

   9.3. If no such elliptic curve exists, log WARNING message, and PASS the test.

   9.4. Go to step 7.

10. ONVIF Client selects signature algorithm by following the procedure mentioned in Annex A.5 with the following input and output parameters:

    - in *cap* - DUT capabilities
    - in ECC - key pair algorithm
    - out *signatureAlgorithm* - signature algorithm

11. ONVIF Client invokes **CreatePKCS10CSR** with parameters

    - Subject =: *subject* (see Annex A.6)
    - KeyID =: *keyID*
    - CSRAttribute skipped
    - SignatureAlgorithm.algorithm := *signatureAlgorithm*

12. If the DUT returns env:Sender\ter:InvalidArgVal\ter:InvalidKeyStatus SOAP 1.2 fault, go to step 14.

13. If the DUT returns normal **CreatePKCS10CSRResponse** message.:

    13.1. ONVIF Client deletes ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.2.

    13.2. Set *ellipticCurve* := (the simplest supported ECC elliptic curve that is harder than the current *ellipticCurve*)

    13.3. If no such elliptic curve exists, log WARNING message, and PASS the test.

    13.4. Go to step 7.

14. ONVIF Client deletes the ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.2 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.33 Delete Certificate – CA – Preserve Public Key (ECC)

Test Case ID: ADVANCED_SECURITY-2-1-37

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: DeleteCertificate

WSDL Reference: security.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: Security Configuration Service is received from the DUT. Create PKCS#10 supported by the DUT as indicated by the PKCS10 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 Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.8 with the following input and output parameters:

   - in *cap* - DUT capabilities
   - in ECC - key pair algorithm
   - out *CAcert* - CA certificate
   - out *CAkeyPair* - key pair with public and private keys

5. ONVIF Client uploads a CA certificate (out *certID*, in *CAcert*) and new ECC key pair with the public key from the CA certificate (out *keyID*) by following the procedure described in Annex A.14.

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

    - CertificateID =: *certID*

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

8. ONVIF Client invokes **GetAllKeys**.

9. The DUT responds with a **GetAllKeysResponse** message where KeyAttribute list contains *keyID*.

10. ONVIF Client deletes the ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.2 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The **GetAllKeysResponse** message at Step 8 does not contain *keyID* in KeyAttribute list.
- DUT did not send **DeleteCertificateResponse** message.
- DUT did not send **GetAllKeysResponse** message.

5.2.34 Upload certificate – delete linked key (ECC, negative test)

Test Case ID: ADVANCED_SECURITY-2-1-38

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadCertificate

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 capability. ECC key pair generation supported by the DUT as indicated by the ECCKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional ECC 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.8 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in ECC - key pair algorithm
   - out *CAcert* - CA certificate
   - out *CAkeyPair* - key pair with public and private keys
5. ONVIF Client creates a certificate from the PKCS#10 request with ECC key pair and associated CA certificate and a corresponding private key by following the procedure described in Annex A.10 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *CAcert* - CA certificate
   - in *CAkeyPair*.privateKey - private key
   - in ECC - CSR key pair algorithm
   - out *cert* - certificate
   - out *keyID1* - ECC key pair
6. ONVIF Client invokes **UploadCertificate** with parameters
   - Certificate := *cert*
   - Alias := "ONVIF_Test"

    - PrivateKeyRequired : = true

7. The DUT responds with a **UploadCertificateResponse** message with parameters
    - CertificateID =: *certID*
    - KeyID =: *keyID1*
8. ONVIF Client invokes **DeleteKey** with parameters
    - KeyID =: *keyID1*
9. The DUT returns env:Sender\ter:InvalidArgVal\ter:ReferenceExists SOAP 1.2 fault.
10. ONVIF Client deletes the CA certificate (in *certID*) and related ECC key pair (in *keyID1*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send env:Sender\ter:InvalidArgVal\ter:ReferenceExists SOAP 1.2 fault.
- DUT did not send **UploadCertificateResponse** message.
- DUT did not send **DeleteKeyResponse** message.

5.2.35 Upload certificate – Upload malformed certificate (ECC, negative test)

Test Case ID: ADVANCED_SECURITY-2-1-39

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadCertificate

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 capability. The DUT shall have enough free storage capacity for one additional ECC 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 Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.8 with the following input and output parameters:

   - in *cap* - DUT capabilities

   - in ECC - key pair algorithm

   - out *CAcert* - CA certificate

   - out *CAkeyPair* - key pair with public and private keys

5. ONVIF Client corrupts *CAcert*.

6. ONVIF Client invokes **UploadCertificate** with parameters

   - Certificate := *CAcert* (malformed)

   - Alias := "ONVIF_Test"

   - PrivateKeyRequired : = false

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

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.36 Upload certificate – Upload expired certificate (ECC)

Test Case ID: ADVANCED_SECURITY-2-1-40

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadCertificate

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 capability. The DUT shall have enough free storage capacity for one additional ECC 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.20 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in ECC - key pair algorithm
   - out *CAcert* - CA certificate
   - out *CAkeyPair* - key pair with public and private keys

5. ONVIF Client invokes **UploadCertificate** with parameters
    - Certificate := *CAcert*
    - Alias := "ONVIF_Test"
    - PrivateKeyRequired : = false
6. The DUT responds with a **UploadCertificateResponse** message with parameters
    - CertificateID =: *certID*
    - KeyID =: *keyID*
7. Check that the DUT does not return env:Sender\ter:InvalidArgVal\ter:BadCertificate SOAP 1.2 fault in **UploadCertificateResponse**.
8. ONVIF Client deletes the CA certificate (in *certID*) and related ECC key pair (in *keyID*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

5.3 TLS Server

5.3.1 Certificate Management

5.3.1.1 Add Server Certificate Assignment – self-signed

Test Case ID: ADVANCED_SECURITY-3-1-1

Specification Coverage: TLS Server (ONVIF Security Configuration Service Specification)

Feature under test: AddServerCertificateAssignment

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or 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. Network Configuration is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client disables HTTPS and removes Server Certificate Assignment if required by following the procedure mentioned in Annex A.21 with the following input and output parameters
   - out *initialHTTPSState* - initial HTTPS State
   - out *certPathID* - removed Server Certificate Assignment
5. ONVIF Client creates a certification path based on self-signed certificate and related RSA key pair by following the procedure mentioned in Annex A.18 with the following input and output parameters:
   - in *cap* - service capabilities
   - in RSA - key pair algorithm
   - out *keyID* - key pair ID
   - out *certID1* - certificate ID

- out *certPathID* - certification path ID

6. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
   - CertificationPathID =: *certPathID*
7. The DUT responds with an **AddServerCertificateAssignmentResponse** message.
8. ONVIF Client waits for time *operationDelay*.
9. 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.22 to restore DUT configuration.
10. ONVIF Client restores HTTPS and Server Certificate Assignment if required by following the procedure mentioned in Annex A.23 with the following input and output parameters
    - in *initialHTTPSState* - initial HTTPS State
    - in *certPathID* - removed Server Certificate Assignment

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.3.1.2 Add Server Certificate Assignment – CA

Test Case ID: ADVANCED_SECURITY-3-1-2

Specification Coverage: TLS Server (ONVIF Security Configuration Service Specification)

Feature under test: AddServerCertificateAssignment

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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. Network Configuration is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client disables HTTPS and removes Server Certificate Assignment if required by following the procedure mentioned in Annex A.21 with the following input and output parameters

   - out *initialHTTPSState* - initial HTTPS State

   - out *certPathID* - removed Server Certificate Assignment

5. ONVIF Client creates a certification path based on CA-signed certificate and related RSA key pair and a corresponding CA certificate and related RSA key pair by following the procedure mentioned in Annex A.19 with the following input and output parameters:

   - in *cap* - DUT capabilities

   - in RSA - CA key pair algorithm

   - in RSA - cert key pair algorithm

     out *certID2* - CA certificate

   - out *keyID2* - CA certificate RSA key pair

   - out *keyID1* - RSA key pair

   - out *certID1* - CA-signed certificate

   - out *certPathID* - certification path

6. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
   - CertificationPathID =: *certPathID*
7. The DUT responds with an **AddServerCertificateAssignmentResponse** message.
8. ONVIF Client waits for time *operationDelay*.
9. ONVIF Client removes server certification assignment and deletes related certification path (in *certPathID*), related CA certificate (in *certID2*) and the 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.24 to restore DUT configuration.
10. ONVIF Client restores HTTPS and Server Certificate Assignment if required by following the procedure mentioned in Annex A.23 with the following input and output parameters
    - in *initialHTTPSState* - initial HTTPS State
    - in *certPathID* - removed Server Certificate Assignment

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.3.1.3 Replace Server Certificate Assignment – self-signed

Test Case ID: ADVANCED_SECURITY-3-1-3

Specification Coverage: TLS Server (ONVIF Security Configuration Service Specification)

Feature under test: ReplaceServerCertificateAssignment

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or 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 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 two additional certification paths. The DUT shall have enough free storage capacity for one additional server certificate assignment. Network Configuration is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client disables HTTPS and removes Server Certificate Assignment if required by following the procedure mentioned in Annex A.21 with the following input and output parameters
   - out *initialHTTPSState* - initial HTTPS State
   - out *certPathID* - removed Server Certificate Assignment
5. ONVIF Client adds server certification assignment and creates related certification path by following the procedure mentioned in Annex A.25 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in RSA - key pair algorithm
   - out *certPathID1* - certification path
   - out *certID1* - self-signed certificate
   - out *keyID1* - RSA key pair
6. ONVIF Client creates a certification path based on self-signed certificate and related RSA key pair by following the procedure mentioned in Annex A.18 with the following input and output parameters:
   - in *cap* - service capabilities
   - in RSA - key pair algorithm

- out *keyID2* - key pair ID
- out *certID2* - certificate ID
- out *certPathID2* - certification path ID

7. ONVIF Client invokes **ReplaceServerCertificateAssignment** with parameters
   - OldCertificationPathID =: *certPathID1*
   - NewCertificationPathID =: *certPathID2*
8. The DUT responds with a **ReplaceServerCertificateAssignmentResponse** message.
9. ONVIF Client waits for time *operationDelay*.
10. ONVIF Client removes server certification assignment and deletes related certification path (*certPathID2*), the self-signed certificate (*certID2*) and the RSA key pair (*keyID2*) by following the procedure mentioned in Annex A.22 to restore DUT configuration.
11. 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.15 to restore DUT configuration.
12. ONVIF Client restores HTTPS and Server Certificate Assignment if required by following the procedure mentioned in Annex A.23 with the following input and output parameters
    - in *initialHTTPSState* - initial HTTPS State
    - in *certPathID* - removed Server Certificate Assignment

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.3.1.4 Replace Server Certificate Assignment – CA

Test Case ID: ADVANCED_SECURITY-3-1-4

Specification Coverage: TLS Server (ONVIF Security Configuration Service Specification)

Feature under test: ReplaceServerCertificateAssignment

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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 three 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. Network Configuration is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client disables HTTPS and removes Server Certificate Assignment if required by following the procedure mentioned in Annex A.21 with the following input and output parameters

   - out *initialHTTPSState* - initial HTTPS State

   - out *certPathID* - removed Server Certificate Assignment

5. ONVIF Client creates a CA certificate and a corresponding private key by following the procedure described in Annex A.8 with the following input and output parameters:

   - in *cap* - DUT capabilities

   - in RSA - key pair algorithm

   - out *CAcert* - CA certificate

- out *CAkeyPair* - key pair with private and public keys

6. 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.14.

7. ONVIF Client creates and uploads a CA-signed certificate for RSA key pair and associated CA certificate and a corresponding by following the procedure described in Annex A.16 with the following input and output parameters:

   - in *cap* - DUT capabilities
   - in *CAcert* - CA certificate
   - in *CAkeyPair*.privateKey - CA certificate private key
   - in RSA - cert key pair algorithm
   - out *certID2* - CA-signed certificate
   - out *keyID2* - RSA key pair

8. ONVIF Client invokes **CreateCertificationPath** with parameters

   - CertficateIDs.CertificateID[0] := *certID2*
   - CertficateIDs.CertificateID[1] := *certID1*
   - Alias := "ONVIF_TestPath1"

9. The DUT responds with a **CreateCertificationPathResponse** message with parameter

   - CertificationPathID =: *certPathID1*

10. ONVIF Client invokes **AddServerCertificateAssignment** with parameter

    - CertificationPathID := *certPathID1*

11. The DUT responds with an **AddServerCertificateAssignmentResponse** message.

12. ONVIF Client waits for time *operationDelay*.

13. ONVIF Client creates and uploads a CA-signed certificate for RSA key pair and associated CA certificate and a corresponding by following the procedure described in Annex A.16 with the following input and output parameters:

    - in *cap* - DUT capabilities

- in *CAcert* - CA certificate
- in *privateKey* - CA certificate private key
- out *certID3* - CA-signed certificate
- out *keyID3* - RSA key pair

14. ONVIF Client invokes **CreateCertificationPath** with parameters
    - CertficateIDs.CertificateID[0] := *certID3*
    - CertficateIDs.CertificateID[1] := *certID1*
    - Alias := "ONVIF_TestPath2"
15. The DUT responds with a **CreateCertificationPathResponse** message with parameter
    - CertificationPathID =: *certPathID2*
16. ONVIF Client invokes **ReplaceServerCertificateAssignment** with parameters
    - OldCertificationPathID := *certPathID1*
    - NewCertificationPathID := *certPathID2*
17. The DUT responds with a **ReplaceServerCertificateAssignmentResponse** message.
18. ONVIF Client waits for time *operationDelay*.
19. ONVIF Client deletes the certification path (in *certPathID1*) and related the self-signed certificate (in *certID2*) and the RSA key pair (in *keyID2*) by following the procedure mentioned in Annex A.15 to restore DUT configuration.
20. 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 *keyID3*) by following the procedure mentioned in Annex A.24 to restore DUT configuration.
21. ONVIF Client restores HTTPS and Server Certificate Assignment if required by following the procedure mentioned in Annex A.23 with the following input and output parameters
    - in *initialHTTPSState* - initial HTTPS State
    - in *certPathID* - removed Server Certificate Assignment

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **AddServerCertificateAssignmentResponse** message.
- DUT did not send **CreateCertificationPathResponse** message(s).
- DUT did not send **ReplaceServerCertificateAssignmentResponse** message.

5.3.1.5 Get Assigned Server Certificates – self-signed

Test Case ID: ADVANCED_SECURITY-3-1-5

Specification Coverage: TLS Server (ONVIF Security Configuration Service Specification)

Feature under test: GetAssignedServerCertificates

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or 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. Network Configuration is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
    - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client disables HTTPS and removes Server Certificate Assignment if required by following the procedure mentioned in Annex A.21 with the following input and output parameters
   - out *initialHTTPSState* - initial HTTPS State
   - out *certPathID* - removed Server Certificate Assignment
5. ONVIF Client adds server certification assignment and creates related certification path by following the procedure mentioned in Annex A.25 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in RSA - key pair algorithm
   - out *certPathID* - certification path
   - out *certID* - self-signed certificate
   - out *keyID* - RSA key pair
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*, FAIL the test, and go to the step 9.
9. 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.22 to restore DUT configuration.
10. ONVIF Client restores HTTPS and Server Certificate Assignment if required by following the procedure mentioned in Annex A.23 with the following input and output parameters
    - in *initialHTTPSState* - initial HTTPS State
    - in *certPathID* - removed Server Certificate Assignment

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.3.1.6 Get Assigned Server Certificates – CA

Test Case ID: ADVANCED_SECURITY-3-1-6

Specification Coverage: TLS Server (ONVIF Security Configuration Service Specification)

Feature under test: GetAssignedServerCertificates

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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. Network Configuration is supported by the DUT.

Test Configuration:ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client disables HTTPS and removes Server Certificate Assignment if required by following the procedure mentioned in Annex A.21 with the following input and output parameters

   - out *initialHTTPSState* - initial HTTPS State

   - out *certPathID* - removed Server Certificate Assignment

5. ONVIF Client creates a certification path based on CA-signed certificate and related RSA key pair and a corresponding CA certificate and related RSA key pair by following the procedure mentioned in Annex A.19 with the following input and output parameters:
    - in *cap* - DUT capabilities
    - in RSA - CA key pair algorithm
    - in RSA - cert key pair algorithm
    - out *certID2* - CA certificate
    - out *keyID2* - CA certificate RSA key pair
    - out *keyID1* - RSA key pair
    - out *certID1* - CA-signed certificate
    - out *certPathID* - certification path
6. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
    - CertificationPathID =: *certPathID*
7. The DUT responds with an **AddServerCertificateAssignmentResponse** message.
8. ONVIF Client waits for time *operationDelay*.
9. ONVIF Client invokes **GetAssignedServerCertificates**
10. The DUT responds with a **GetAssignedServerCertificatesResponse** message with parameters
    - CertificationPathID list =: *updatedCertificationPathList*
11. If *updatedCertificationPathList* does not contain *certPathID*, 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.24 to restore DUT configuration
13. ONVIF Client restores HTTPS and Server Certificate Assignment if required by following the procedure mentioned in Annex A.23 with the following input and output parameters
    - in *initialHTTPSState* - initial HTTPS State

- in *certPathID* - removed Server Certificate Assignment

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetAssignedServerCertificatesResponse** message(s).
- DUT did not send **AddServerCertificateAssignmentResponse** message.

5.3.1.7 Remove Server Certificate Assignment – self-signed

Test Case ID: ADVANCED_SECURITY-3-1-7

Specification Coverage: TLS Server (ONVIF Security Configuration Service Specification)

Feature under test: RemoveServerCertificateAssignment

WSDL Reference:security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or 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. Network Configuration is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

- out *cap* - Security Configuration Service Capabilities

4. ONVIF Client disables HTTPS and removes Server Certificate Assignment if required by following the procedure mentioned in Annex A.21 with the following input and output parameters
    - out *initialHTTPSState* - initial HTTPS State
    - out *certPathID* - removed Server Certificate Assignment
5. ONVIF Client adds server certification assignment and creates related certification path by following the procedure mentioned in Annex A.25 with the following input and output parameters:
    - in *cap* - DUT capabilities
    - in RSA - key pair algorithm
    - out *certPathID* - certification path
    - out *certID* - self-signed certificate
    - out *keyID* - RSA key pair
6. ONVIF Client invokes **RemoveServerCertificateAssignment** .
    - CertificationPathID =: *certPathID*
7. The DUT responds with a **RemoveServerCertificateAssignmentResponse** message.
8. ONVIF Client waits for time *operationDelay*.
9. 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.15 to restore DUT configuration.
10. ONVIF Client invokes **GetAssignedServerCertificates**.
11. The DUT responds with a **GetAssignedServerCertificatesResponse** message with parameters
    - CertificationPathID list =: *finalCertificationPathList*
12. If *finalCertificationPathList* contains *certPathID*, FAIL the test.
13. ONVIF Client restores HTTPS and Server Certificate Assignment if required by following the procedure mentioned in Annex A.23 with the following input and output parameters

- in *initialHTTPSState* - initial HTTPS State
- in *certPathID* - removed Server Certificate Assignment

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **RemoveServerCertificateAssignmentResponse** message.
- DUT did not send **GetAssignedServerCertificatesResponse** message(s).

5.3.1.8 Remove Server Certificate Assignment – CA

Test Case ID: ADVANCED_SECURITY-3-1-8

Specification Coverage: TLS Server (ONVIF Security Configuration Service Specification)

Feature under test: RemoveServerCertificateAssignment

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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. Network Configuration is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
    - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client disables HTTPS and removes Server Certificate Assignment if required by following the procedure mentioned in Annex A.21 with the following input and output parameters
    - out *initialHTTPSState* - initial HTTPS State
    - out *certPathID* - removed Server Certificate Assignment
5. ONVIF Client creates a certification path based on CA-signed certificate and related RSA key pair and a corresponding CA certificate and related RSA key pair by following the procedure mentioned in Annex A.19 with the following input and output parameters:
    - in *cap* - DUT capabilities
    - in RSA - CA key pair algorithm
    - in RSA - cert key pair algorithm
    - out *certID2* - CA certificate
    - out *keyID2* - CA certificate RSA key pair
    - out *keyID1* - RSA key pair
    - out *certID1* - CA-signed certificate
    - out *certPathID* - certification path
6. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
    - CertificationPathID =: *certPathID*
7. The DUT responds with an **AddServerCertificateAssignmentResponse** message.
8. ONVIF Client waits for time *operationDelay*.
9. ONVIF Client invokes **RemoveServerCertificateAssignment** with parameters
    - CertificationPathID =: *certPathID*
10. The DUT responds with a **RemoveServerCertificateAssignmentResponse** message.
11. ONVIF Client waits for time *operationDelay*.

12. 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.

13. ONVIF Client invokes **GetAssignedServerCertificates**.

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

    - CertificationPathID list =: *finalCertificationPathList*

15. If *finalCertificationPathList* contains *certPathID*, FAIL the test.

16. ONVIF Client restores HTTPS and Server Certificate Assignment if required by following the procedure mentioned in Annex A.23 with the following input and output parameters

    - in *initialHTTPSState* - initial HTTPS State

    - in *certPathID* - removed Server Certificate Assignment

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

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

5.3.2 TLS Handshaking

5.3.2.1 Basic TLS Handshake

Test Case ID: ADVANCED_SECURITY-3-2-3

Specification Coverage: TLS Server (ONVIF Security Configuration Service Specification)

Feature under test: execute Basic TLS Handshake

WSDL Reference: security.wsdl

Test Purpose: Check TLS handshake with certificate.

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or 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. Network Configuration is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client disables HTTPS and removes Server Certificate Assignment if required by following the procedure mentioned in Annex A.21 with the following input and output parameters

   - out *initialHTTPSState* - initial HTTPS State

   - out *certPathID* - removed Server Certificate Assignment

5. ONVIF Client invokes **GetNetworkProtocols** to retrieve configured network protocols of the DUT.

6. The DUT responds with a **GetNetworkProtocolsResponse** message with parameters

   - NetworkProtocols list =: *networkProtocolsList*

7. If *networkProtocolsList* does not contain network protocol with NetworkProtocols.Name is equal to "HTTPS", FAIL the test and skip other steps.

8. ONVIF Client creates a certification path based on self-signed certificate and related RSA key pair by following the procedure mentioned in Annex A.18 with the following input and output parameters:

- in *cap* - service capabilities
- in RSA - key pair algorithm
- out *keyID* - key pair ID
- out *certID* - certificate ID
- out *certPathID* - certification path ID

9. ONVIF Client invokes **GetCertificate** message with parameters
    - CertificateID := *certID*
10. The DUT responds with a **GetCertificateResponse** message with parameters
    - Certificate =: *X509Cert*
11. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
    - CertificationPathID := *certPathID*
12. The DUT responds with an **AddServerCertificateAssignmentResponse** message.
13. ONVIF Client waits for time *operationDelay*.
14. If HTTPS protocol with NetworkProtocols.Name is equal to "HTTPS" from *networkProtocolsList* has NetworkProtocols.Enabled equal to true, go to the step 18.
15. ONVIF Client invokes **SetNetworkProtocols** message with parameters
    - NetworkProtocols[0].Name := HTTPS
    - NetworkProtocols[0].Enabled := true
    - NetworkProtocols[0].Port := 443
    - NetworkProtocols[0].Extension skipped
16. The DUT responds with a **SetNetworkProtocolsResponse** message.
17. ONVIF Client waits for time *operationDelay*.
18. ONVIF Client checks that HTTPS protocol Port is open. If HTTPS protocol *portHTTPS* is not open, FAIL the test and go to the step 20.
19. 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.26.

20. If HTTPS protocol with NetworkProtocols.Name is equal to "HTTPS" from *networkProtocolsList* has NetworkProtocols.Enabled equal to true, go to the step 24.

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

    - NetworkProtocols list := *networkProtocolsList*

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

23. ONVIF Client waits for time *operationDelay*.

24. 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.22 to restore DUT configuration.

25. ONVIF Client restores HTTPS and Server Certificate Assignment if required by following the procedure mentioned in Annex A.23 with the following input and output parameters

    - in *initialHTTPSState* - initial HTTPS State

    - in *certPathID* - removed Server Certificate Assignment

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetNetworkProtocolsResponse** message(s).

- DUT did not send **GetCertificateResponse** message(s).

- DUT did not send **AddServerCertificateAssignmentResponse** message(s).

- The DUT did not provide Basic TLS handshake at step 19.

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.26 execution, log WARNING message.

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

5.3.2.2 Basic TLS Handshake after Replace Server Certificate Assignment

Test Case ID: ADVANCED_SECURITY-3-2-4

Specification Coverage: TLS Server (ONVIF Security Configuration Service Specification)

Feature under test: ReplaceServerCertificateAssignment

WSDL Reference: security.wsdl

Test Purpose: Check TLS handshake with replaced certificate.

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or 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 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 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. Network Configuration is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client disables HTTPS and removes Server Certificate Assignment if required by following the procedure mentioned in Annex A.21 with the following input and output parameters

   - out *initialHTTPSState* - initial HTTPS State

   - out *certPathID* - removed Server Certificate Assignment

5. ONVIF Client invokes **GetNetworkProtocols** to retrieve configured network protocols of the DUT.

6. The DUT responds with a **GetNetworkProtocolsResponse** message with parameters
    - NetworkProtocols list =: *networkProtocolsList*
7. If *networkProtocolsList* does not contain network protocol with NetworkProtocols.Name is equal to "HTTPS", FAIL the test, restore the DUT state, and skip other steps.
8. ONVIF Client adds server certification assignment and creates related certification path by following the procedure mentioned in Annex A.25 with the following input and output parameters:
    - in *cap* - DUT capabilities
    - in RSA - key pair algorithm
    - out *certPathID1* - certification path
    - out *certID1* - self-signed certificate
    - out *keyID1* - RSA key pair
9. ONVIF Client invokes **GetCertificate** message with parameters
    - CertificateID := *certID1*
10. The DUT responds with a **GetCertificateResponse** message with parameters
    - Certificate =: *X509Cert1*
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 *operationDelay*.
15. ONVIF Client checks that HTTPS protocol Port is open. If HTTPS protocol *portHTTPS* is not open, FAIL the test, restore the DUT state, and skip other steps.

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.26.

17. ONVIF Client creates a certification path based on self-signed certificate and related RSA key pair by following the procedure mentioned in Annex A.18 with the following input and output parameters:

    - in *cap* - service capabilities
    - in RSA - key pair algorithm
    - out *keyID2* - key pair ID
    - out *certID2* - certificate ID
    - out *certPathID2* - certification path ID

18. ONVIF Client invokes **ReplaceServerCertificateAssignment** with parameters

    - OldCertificationPathID =: *certPathID1*
    - NewCertificationPathID =: *certPathID2*

19. The DUT responds with a **ReplaceServerCertificateAssignmentResponse** message.

20. ONVIF Client waits for time *operationDelay*.

21. ONVIF Client invokes **GetCertificate** message with parameters

    - CertificateID := *certID2*

22. The DUT responds with a **GetCertificateResponse** message with parameters

    - Certificate =: *X509Cert2*

23. 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.26

24. ONVIF Client selects network protocol with Name ="HTTPS" in *networkProtocolsList* (received in step 6) and saves this protocol in *HTTPProtocol* variable.

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

    - NetworkProtocols[0] := *HTTPProtocol*

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

27. ONVIF Client removes server certification assignment and deletes related certification path (*certPathID2*), the self-signed certificate (*certID2*) and the RSA key pair (*keyID2*) by following the procedure mentioned in Annex A.22 to restore DUT configuration.

28. 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.15 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetNetworkProtocolsResponse** message.
- DUT did not send **GetCertificateResponse** message.
- DUT did not send **ReplaceServerCertificateAssignmentResponse** message.
- The DUT did not provide Basic TLS handshake at step 16.
- The DUT did not provide Basic TLS handshake at step 23.

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.26 execution, log WARNING message.

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

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

Test Case ID: ADVANCED_SECURITY-3-2-5

Specification Coverage: TLS Server (ONVIF Security Configuration Service Specification)

Feature under test: UploadCertificateWithPrivateKeyInPKCS12, AddServerCertificateAssignment, ReplaceServerCertificateAssignment, GetAssignedServerCertificates

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration 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 PKCS12 or 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. Network Configuration is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client disables HTTPS and removes Server Certificate Assignment if required by following the procedure mentioned in Annex A.21 with the following input and output parameters
   - out *initialHTTPSState* - initial HTTPS State
   - out *certPathID* - removed Server Certificate Assignment
5. ONVIF Client invokes **GetNetworkProtocols** to retrieve configured network protocols of the DUT.
6. The DUT responds with a **GetNetworkProtocolsResponse** message with parameters
   - NetworkProtocols list =: *networkProtocolsList*
7. If *networkProtocolsList* does not contain network protocol with NetworkProtocols.Name is equal to "HTTPS", FAIL the test and skip other steps.

8. Set
   - *portHTTPS* =: *networkProtocolsList*.NetworkProtocols[Name = "HTTPS"].Port
9. ONVIF Client generates an encryption passphrase *passphrase1* (see Annex A.27).
10. ONVIF Client creates a CA certificate and a corresponding public key in the certificate along with the corresponding private key in the form of a PKCS#12 file with given passphrase by following the procedure described in Annex A.28 with the following input and output parameters:
    - in *cap* - DUT capabilities
    - in RSA - key pair algorithm
    - in *passphrase1* - given passphrase
    - out *CAcert* - CA certificate
    - out *CAkeyPair* - key pair with public and private keys
    - out *PKCS12data* - PKCS#12 file
11. ONVIF Client invokes **UploadCertificateWithPrivateKeyInPKCS12** message with parameters
    - CertWithPrivateKey := *PKCS12data1*
    - CertificationPathAlias := "ONVIF_CertificationPath_Test"
    - KeyAlias := "ONVIF_Key_Test"
    - IgnoreAdditionalCertificates skiped
    - IntegrityPassphraseID skipped
    - EncryptionPassphraseID skipped
    - Passphrase := *passphrase1*
12. The DUT responds with an **UploadCertificateWithPrivateKeyInPKCS12Response** message with parameters
    - CertificationPathID =: *certPathID1*
    - KeyID =: *keyID1*
13. ONVIF Client invokes **AddServerCertificateAssignment** with parameters

- CertificationPathID := *certPathID1*

14. The DUT responds with an **AddServerCertificateAssignmentResponse** message.

15. ONVIF Client waits for time *operationDelay*.

16. If HTTPS protocol with NetworkProtocols.Name is equal to "HTTPS" from *networkProtocolsList* has NetworkProtocols.Enabled equal to true, go to the step 21.

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

    - NetworkProtocols[0].Name := HTTPS
    - NetworkProtocols[0].Enabled := true
    - NetworkProtocols[0].Port := 443
    - NetworkProtocols[0].Extension skipped

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

19. ONVIF Client waits for time *operationDelay*.

20. Set

    - *portHTTPS* =: 443

21. ONVIF Client checks that HTTPS protocol Port is open. If HTTPS protocol *portHTTPS* is not open, FAIL the test and go to the step 41.

22. 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.26.

23. If the DUT presents Certificate which is not equal to *PKCS12data1* during the Annex A.26 execution, FAIL the test and go to the step 41.

24. ONVIF Client creates a CA certificate with subject and a corresponding public key in the certificate along with the corresponding private key in the form of a PKCS#12 file with given passphrase by following the procedure described in Annex A.28 with the following input and output parameters:

    - in *cap* - DUT capabilities
    - in RSA - key pair algorithm
    - in *passphrase1* - given passphrase

- in *"CN=ONVIF TT2,C=US"* - subject
- out *CAcert2* - CA certificate
- out *CAkeyPair2* - key pair with public and private keys
- out *PKCS12data2* - PKCS#12 file

25. ONVIF Client invokes **UploadCertificateWithPrivateKeyInPKCS12** message with parameters
    - CertWithPrivateKey := *PKCS12data2*
    - CertificationPathAlias := "ONVIF_CertificationPath_Test2"
    - KeyAlias := "ONVIF_Key_Test2"
    - IgnoreAdditionalCertificates skipped
    - IntegrityPassphraseID skipped
    - EncryptionPassphraseID skipped
    - Passphrase := *passphrase1*
26. The DUT responds with an **UploadCertificateWithPrivateKeyInPKCS12Response** message with parameters
    - CertificationPathID =: *certPathID2*
    - KeyID =: *keyID2*
27. ONVIF Client invokes **ReplaceServerCertificateAssignment** with parameters
    - OldCertificationPathID =: *certPathID1*
    - NewCertificationPathID =: *certPathID2*
28. The DUT responds with a **ReplaceServerCertificateAssignmentResponse** message.
29. ONVIF Client waits for time *operationDelay*.
30. ONVIF Client invokes **GetAssignedServerCertificates**.
31. The DUT responds with a **GetAssignedServerCertificatesResponse** message with parameters
    - CertificationPathID list =: *certificationPathIDList*

32. If *certificationPathIDList* contains certification path with CertificationPathID equal to *certPathID1*, FAIL the test and go to the step 40.

33. If *certificationPathIDList* does not contain certification path with CertificationPathID equal to *certPathID2*, FAIL the test and go to the step 40.

34. ONVIF Client checks that HTTPS protocol Port is open. If HTTPS protocol *portHTTPS* is not open, FAIL the test and go to the step 37.

35. 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.26.

36. If the DUT presents Certificate which is not equal to *X509Cert* during the Annex A.26 execution, FAIL the test and go to the step 37.

37. ONVIF Client invokes **RemoveServerCertificateAssignment** with parameters

    - CertificationPathID := *certPathID2*

38. The DUT responds with a **RemoveServerCertificateAssignmentResponse** message.

39. ONVIF Client waits for time *operationDelay*.

40. ONVIF Client deletes the certification path (in *certPathID2*) and RSA key pair (in *keyID2*) by following the procedure mentioned in Annex A.30 to restore DUT configuration.

41. ONVIF Client deletes the certification path (in *certPathID1*) and RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.30 to restore DUT configuration.

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

    - NetworkProtocols := *networkProtocolsList*

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

44. ONVIF Client waits for time *operationDelay*.

45. ONVIF Client restores HTTPS and Server Certificate Assignment if required by following the procedure mentioned in Annex A.23 with the following input and output parameters

    - in *initialHTTPSState* - initial HTTPS State

    - in *certPathID* - removed Server Certificate Assignment

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetNetworkProtocolsResponse** message.
- DUT did not send **SetNetworkProtocolsResponse** message.
- DUT did not send **UploadCertificateWithPrivateKeyInPKCS12Response** message.s.
- DUT did not send **AddServerCertificateAssignmentResponse** message.s.
- DUT did not send **ReplaceServerCertificateAssignmentResponse** message.
- DUT did not send **GetAssignedServerCertificatesResponse** message.
- DUT did not send **RemoveServerCertificateAssignmentResponse** message.
- The DUT did not provide Basic TLS handshake at Step 22.
- The DUT did not provide Basic TLS handshake at Step 35.

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

5.3.3 TLS client authentication

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

Test Case ID: ADVANCED_SECURITY-3-3-1

Specification Coverage: TLS Server (ONVIF Security Configuration Service Specification)

Feature under test: GetClientAuthenticationRequired, SetClientAuthenticationRequired, AddCertPathValidationPolicyAssignment

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or 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 PKCS10 or 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. Network Configuration is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client disables HTTPS and removes Server Certificate Assignment if required by following the procedure mentioned in Annex A.21 with the following input and output parameters
   - out *initialHTTPSState* - initial HTTPS State
   - out *certPathID* - removed Server Certificate Assignment
5. ONVIF Client invokes **GetNetworkProtocols** to retrieve configured network protocols of the DUT.
6. The DUT responds with a **GetNetworkProtocolsResponse** message with parameters
   - NetworkProtocols list =: *networkProtocolsList*
7. If *networkProtocolsList* does not contain network protocol with NetworkProtocols.Name is equal to "HTTPS", FAIL the test and skip other steps.
8. Set

- *portHTTPS* =: *networkProtocolsList*.NetworkProtocols[Name = "HTTPS"].Port

9. ONVIF Client invokes **GetClientAuthenticationRequired**.

10. The DUT responds with a **GetClientAuthenticationRequiredResponse** message with parameters

    - clientAuthenticationRequired =: *clientAuthenticationRequired*

11. ONVIF Client adds server certification assignment and creates related certification path by following the procedure mentioned in Annex A.25 with the following input and output parameters:

    - in *cap* - DUT capabilities
    - in RSA - key pair algorithm
    - out *certPathID0* - certification path
    - out *certID0* - self-signed certificate
    - out *keyID0* - RSA key pair

12. ONVIF Client creates an CA certificate by following the procedure mentioned in Annex A.8 with the following input and output parameters

    - in *cap* - DUT capabilities
    - in RSA - key pair algorithm
    - in "CN=ONVIF1 TT,C=US" - CA certificate subject
    - out *CAcert1* - CA certificate
    - out *CAkeyPair1* - key pair with private and public keys for the CA certificate

13. 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.14.

14. ONVIF Client creates a certificate signed by private key of the CA-certificate with subject and a corresponding public key in the certificate along with the corresponding private key by following the procedure described in Annex A.31 with the following input and output parameters:

    - in *cap* - DUT capabilities

- in RSA - key pair algorithm
- in "CN=ONVIF2 TT,C=US" - certificate subject
- in *CAcert1* - CA-certificate
- in *CAkeyPair1*.privateKey - private key of CA-certificate for certificate signature
- out *cert2* - cerificate
- out *keyPair2* - key pair with private and public keys of the certificate

15. ONVIF Client creates an CA certificate by following the procedure mentioned in Annex A.8 with the following input and output parameters

    - in *cap* - DUT capabilities
    - in RSA - key pair algorithm
    - in "CN=ONVIF3 TT,C=US" - CA certificate subject
    - out *CAcert3* - CA certificate
    - out *CAkeyPair3* - key pair with private and public keys of the certificate

16. ONVIF Client creates a certificate signed by private key of the CA-certificate with subject and a corresponding public key in the certificate along with the corresponding private key by following the procedure described in Annex A.31 with the following input and output parameters:

    - in *cap* - DUT capabilities
    - in RSA - key pair algorithm
    - in "CN=ONVIF4 TT,C=US" - certificate subject
    - in *CAcert3* - CA-certificate
    - in *CAkeyPair3*.privateKey - private key of CA-certificate for certificate signature
    - out *cert4* - cerificate
    - out *keyPair4* - key pair with private and public keys of the certificate

17. 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.32.

18. ONVIF Client invokes **AddCertPathValidationPolicyAssignment** with parameters

    - CertPathValidationPolicyID := *certPathValidationPolicyID*

19. The DUT responds with an **AddCertPathValidationPolicyAssignmentResponse** message.

20. If HTTPS protocol with NetworkProtocols.Name is equal to "HTTPS" from *networkProtocolsList* has NetworkProtocols.Enabled equal to true, go to the step 25.

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

    - NetworkProtocols[0].Name := HTTPS
    - NetworkProtocols[0].Enabled := true
    - NetworkProtocols[0].Port := 443
    - NetworkProtocols[0].Extension skipped

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

23. ONVIF Client waits for time *operationDelay*.

24. Set

    - *portHTTPS* := 443.

25. If *clientAuthenticationRequired* is equal to false:

    25.1. ONVIF Client invokes **SetClientAuthenticationRequired** with parameters

    - clientAuthenticationRequired := true

    25.2. The DUT responds with a **SetClientAuthenticationRequiredResponse** message.

26. ONVIF Client invokes **GetClientAuthenticationRequired** through HTTPS using the client certificate *cert2* and port *portHTTPS*.

27. The DUT responds with a **GetClientAuthenticationRequiredResponse** message with parameters

    - clientAuthenticationRequired =: *clientAuthenticationRequired1*

28. ONVIF Client invokes **GetClientAuthenticationRequired** through HTTPS using the client certificate *cert4* and port *portHTTPS*.

29. The DUT does not establish a TLS connection.

30. If *clientAuthenticationRequired* is equal to false:

    30.1. ONVIF Client invokes **SetClientAuthenticationRequired** with parameters

    - clientAuthenticationRequired := false

    30.2. The DUT responds with a **SetClientAuthenticationRequiredResponse** message.

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

    - NetworkProtocols := *networkProtocolsList*

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

33. ONVIF Client waits for time *operationDelay*.

34. ONVIF Client invokes **RemoveCertPathValidationPolicyAssignment** with parameters

    - CertPathValidationPolicyID := *certPathValidationPolicyID*

35. The DUT responds with a **RemoveCertPathValidationPolicyAssignmentResponse** message.

36. ONVIF Client deletes the certification path validation policy (in *certPathValidationPolicyID*) by following the procedure mentioned in Annex A.33 to restore DUT configuration.

37. ONVIF Client deletes the CA certificate (in *certID1*) and related RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.7 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.22 to restore DUT configuration.

39. ONVIF Client restores HTTPS and Server Certificate Assignment if required by following the procedure mentioned in Annex A.23 with the following input and output parameters

    - in *initialHTTPSState* - initial HTTPS State

    - in *certPathID* - removed Server Certificate Assignment

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **RemoveCertPathValidationPolicyAssignmentResponse** message.
- DUT did not send **SetNetworkProtocolsResponse** message.
- DUT did not send **SetClientAuthenticationRequiredResponse** message.
- The DUT establishes a TLS connection for step 29.
- DUT did not send **GetClientAuthenticationRequiredResponse** message.
- DUT did not send **AddCertPathValidationPolicyAssignmentResponse** message.
- DUT did not send **GetNetworkProtocolsResponse** message.

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

5.3.3.2 CRL processing with on-device RSA key pair

Test Case ID: ADVANCED_SECURITY-3-3-2

Specification Coverage: TLS Server (ONVIF Security Configuration Service Specification)

Feature under test: GetClientAuthentication, SetClientAuthenticationRequired

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or 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 PKCS10 or 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. Network Configuration is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client disables HTTPS and removes Server Certificate Assignment if required by following the procedure mentioned in Annex A.21 with the following input and output parameters
   - out *initialHTTPSState* - initial HTTPS State
   - out *certPathID* - removed Server Certificate Assignment
5. ONVIF Client invokes **GetNetworkProtocols** to retrieve configured network protocols of the DUT.
6. The DUT responds with a **GetNetworkProtocolsResponse** message with parameters
   - NetworkProtocols list =: *networkProtocolsList*
7. If *networkProtocolsList* does not contain network protocol with NetworkProtocols.Name is equal to "HTTPS", FAIL the test and skip other steps.
8. Set
   - *portHTTPS* =: *networkProtocolsList*.NetworkProtocols[Name = "HTTPS"].Port
9. ONVIF Client invokes **GetClientAuthenticationRequired**.
10. The DUT responds with a **GetClientAuthenticationRequiredResponse** message with parameters
    - clientAuthenticationRequired =: *clientAuthenticationRequired*
11. ONVIF Client adds server certification assignment and creates related certification path by following the procedure mentioned in Annex A.25 with the following input and output parameters:
    - in *cap* - DUT capabilities
    - in RSA - key pair algorithm
    - out *certPathID0* - certification path

- out *certID0* - self-signed certificate
- out *keyID0* - RSA key pair

12. ONVIF Client creates an CA certificate by following the procedure mentioned in Annex A.8 with the following input and output parameters

- in *cap* - DUT capabilities
- in RSA - key pair algorithm
- in "CN=ONVIF1 TT,C=US" - CA certificate subject
- out *CAcert1* - CA certificate
- out *CAkeyPair1* - key pair with private and public keys of the certificate

13. 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.14.

14. ONVIF Client creates a certificate signed by private key of the CA-certificate with subject and a corresponding public key in the certificate along with the corresponding private key by following the procedure described in Annex A.31 with the following input and output parameters:

- in *cap* - DUT capabilities
- in RSA - key pair algorithm
- in "CN=ONVIF2 TT,C=US" - certificate subject
- in *CAcert1* - CA-certificate
- in *CAkeyPair1*.privateKey - private key of CA-certificate for certificate signature
- out *cert2* - cerificate
- out *keyPair2* - key pair with private and public keys of the certificate

15. ONVIF Client creates a certificate signed by private key of the CA-certificate with subject and a corresponding public key in the certificate along with the corresponding private key by following the procedure described in Annex A.31 with the following input and output parameters:

- in *cap* - DUT capabilities

- in RSA - key pair algorithm
- in "CN=ONVIF3 TT,C=US" - certificate subject
- in *CAcert1* - CA-certificate
- in *CAkeyPair1*.privateKey - private key of CA-certificate for certificate signature
- out *cert3* - cerificate
- out *keyPair3* - key pair with private and public keys of the certificate

16. ONVIF Client creates a CRL for certificate revocation signed by private key by following the procedure mentioned in Annex A.34 with the following input and output parameters:
    - in *cap* - DUT capabilities
    - in *cert3* - certificate with revocation
    - in *CAkeyPair1*.privateKey - private key to be used to signe certificate with revocation
    - out *crl* - CRL
17. ONVIF Client uploads a CRL (in *crl*) with alias (in "ONVIF_CRL_Test") identifier (out *crlID*) by following the procedure described in Annex A.35.
18. 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.32.
19. ONVIF Client invokes **AddCertPathValidationPolicyAssignment** with parameters
    - CertPathValidationPolicyID := *certPathValidationPolicyID*
20. The DUT responds with an **AddCertPathValidationPolicyAssignmentResponse** message.
21. If HTTPS protocol with NetworkProtocols.Name is equal to "HTTPS" from *networkProtocolsList* has NetworkProtocols.Enabled equal to true, go to the step 26.
22. ONVIF Client invokes **SetNetworkProtocols** message with parameters
    - NetworkProtocols[0].Name := HTTPS
    - NetworkProtocols[0].Enabled := true
    - NetworkProtocols[0].Port := 443

- NetworkProtocols[0].Extension skipped

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

24. ONVIF Client waits for time *operationDelay*.

25. Set

- *portHTTPS* := 443.

26. If *clientAuthenticationRequired* is equal to false:

    26.1. ONVIF Client invokes **SetClientAuthenticationRequired** with parameters

    - clientAuthenticationRequired := true

    26.2. The DUT responds with a **SetClientAuthenticationRequiredResponse** message.

27. ONVIF Client invokes **GetClientAuthenticationRequired** through HTTPS using the client certificate *cert2* and port *portHTTPS*.

28. The DUT responds with a **GetClientAuthenticationRequiredResponse** message with parameters

- clientAuthenticationRequired =: *clientAuthenticationRequired1*

29. ONVIF Client invokes **GetClientAuthenticationRequired** through HTTPS using the client certificate *cert3* and port *portHTTPS*.

30. The DUT does not establish a TLS connection.

31. If *clientAuthenticationRequired* is equal to false:

    31.1. ONVIF Client invokes **SetClientAuthenticationRequired** with parameters

    - clientAuthenticationRequired := false

    31.2. The DUT responds with a **SetClientAuthenticationRequiredResponse** message.

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

- NetworkProtocols := *networkProtocolsList*

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

34. ONVIF Client waits for time *operationDelay*.

35. ONVIF Client invokes **RemoveCertPathValidationPolicyAssignment** with parameters

- CertPathValidationPolicyID := *certPathValidationPolicyID*

36. The DUT responds with a **RemoveCertPathValidationPolicyAssignmentResponse** message.

37. ONVIF Client deletes the certification path validation policy (in *certPathValidationPolicyID*) by following the procedure mentioned in Annex A.33 to restore DUT configuration.

38. ONVIF Client deletes the CRL (in *crlID*) by following the procedure mentioned in Annex A.36 to restore DUT configuration.

39. ONVIF Client deletes the CA certificate (in *certID1*) and related RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.

40. 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.22 to restore DUT configuration.

41. ONVIF Client restores HTTPS and Server Certificate Assignment if required by following the procedure mentioned in Annex A.23 with the following input and output parameters

    - in *initialHTTPSState* - initial HTTPS State

    - in *certPathID* - removed Server Certificate Assignment

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **RemoveCertPathValidationPolicyAssignmentResponse** message.
- DUT did not send **SetNetworkProtocolsResponse** message.
- DUT did not send **SetClientAuthenticationRequiredResponse** message.
- The DUT establishes a TLS connection for step 30.
- DUT did not send **GetClientAuthenticationRequiredResponse** message.
- DUT did not send **AddCertPathValidationPolicyAssignmentResponse** message.
- DUT did not send **GetNetworkProtocolsResponse** message.

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

5.3.3.3 Replace certification path validation policy assignment

Test Case ID: ADVANCED_SECURITY-3-3-3

Specification Coverage: Certification Path Validation Policy Management (ONVIF Security Configuration Service Specification)

Feature under test: ReplaceCertPathValidationPolicyAssignment, GetAssignedCertPathValidationPolicies

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration 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 TLSClientAuthSupported 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 Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

    • out *cap* - Security Configuration Service Capabilities

4. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.37 with the following input and output parameters

    • in *cap* - DUT capabilities

- in RSA - key pair algorithm
- in "CN=ONVIF TT1,C=US" - CA certificate subject
- in "Test CertPathValidationPolicy1 Alias" - certification path validation policy alias
- out *certPathValidationPolicyID1* - certification path validation policy identifier
- out *certificationPathID1* - certification path identifier (if any)
- out *certID1* - certificate identifier
- out *keyID1* - RSA key pair identifier

5. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.37 with the following input and output parameters
    - in *cap* - DUT capabilities
    - in RSA - key pair algorithm
    - in "CN=ONVIF TT2,C=US" - CA certificate subject
    - in "Test CertPathValidationPolicy2 Alias" - certification path validation policy alias
    - out *certPathValidationPolicyID2* - certification path validation policy identifier
    - out *certificationPathID2* - certification path identifier (if any)
    - out *certID2* - certificate identifier
    - out *keyID2* - RSA key pair identifier
6. ONVIF Client invokes **AddCertPathValidationPolicyAssignment** with parameters
    - CertPathValidationPolicyID := *certPathValidationPolicyID1*
7. The DUT responds with an **AddCertPathValidationPolicyAssignmentResponse** message.
8. ONVIF Client invokes **ReplaceCertPathValidationPolicyAssignment** with parameters
    - OldCertPathValidationPolicyID := *certPathValidationPolicyID1*
    - NewCertPathValidationPolicyID := *certPathValidationPolicyID2*
9. The DUT responds with a **ReplaceCertPathValidationPolicyAssignmentResponse** message.

10. ONVIF Client invokes **GetAssignedCertPathValidationPolicies**.

11. The DUT responds with a **GetAssignedCertPathValidationPoliciesResponse** message with parameters

    • CertPathValidationPolicyID list =: *certPathValidationPolicyIDList*

12. If *certPathValidationPolicyIDList* contains CertPathValidationPolicyID equal to *certPathValidationPolicyID1*, FAIL the test and go to the step 14.

13. If *certPathValidationPolicyIDList* does not contain CertPathValidationPolicyID equal to *certPathValidationPolicyID2*, FAIL the test and go to the step 19.

14. Go to the step 17.

15. ONVIF Client invokes **RemoveCertPathValidationPolicyAssignment** with parameters

    • CertPathValidationPolicyID := *certPathValidationPolicyID1*

16. The DUT responds with a **RemoveCertPathValidationPolicyAssignmentResponse** message.

17. Go to the step 19.

18. ONVIF Client invokes **RemoveCertPathValidationPolicyAssignment** with parameters

    • CertPathValidationPolicyID := *certPathValidationPolicyID2*

19. The DUT responds with a **RemoveCertPathValidationPolicyAssignmentResponse** message.

20. ONVIF Client invokes **DeleteCertPathValidationPolicy** with parameters

    • CertPathValidationPolicyID := *certPathValidationPolicyID2*

21. DUT responds with a **DeleteCertPathValidationPolicyResponse** message.

22. ONVIF Client deletes the certification path validation policy (in *certPathValidationPolicyID1*) by following the procedure mentioned in Annex A.33 to restore DUT configuration.

23. If *certificationPathID1* is null:

    23.1. ONVIF Client deletes the self-signed certificate (in *certID1*) and related the RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.7.

    23.2. Skip other steps.

24. If *certificationPathID1* is not null:

24.1. ONVIF Client deletes the certification path (in *certificationPathID1*) and RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.30 to restore DUT configuration.

25. If *certificationPathID2* is null:

25.1. ONVIF Client deletes the self-signed certificate (in *certID2*) and related the RSA key pair (in *keyID2*) by following the procedure mentioned in Annex A.7.

25.2. Skip other steps.

26. If *certificationPathID2* is not null:

26.1. ONVIF Client deletes the certification path (in *certificationPathID2*) and RSA key pair (in *keyID2*) by following the procedure mentioned in Annex A.30 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **AddCertPathValidationPolicyAssignmentResponse** message.
- DUT did not send **ReplaceCertPathValidationPolicyAssignmentResponse** message.
- DUT did not send **GetAssignedCertPathValidationPoliciesResponse** message.
- DUT did not send **RemoveCertPathValidationPolicyAssignmentResponse** message.
- DUT did not send **DeleteCertPathValidationPolicyResponse** message.

5.4 Referential Integrity

5.4.1 TLS Server Certificate - self-signed

Test Case ID: ADVANCED_SECURITY-4-1-1

Specification Coverage: Key Management (ONVIF Security Configuration Service Specification), Certificate Management (ONVIF Security Configuration Service Specification), TLS Server (ONVIF Security Configuration Service Specification)

Feature under test: DeleteKey, DeleteCertificate. DeleteCertificationPath

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or 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. Network Configuration is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client disables HTTPS and removes Server Certificate Assignment if required by following the procedure mentioned in Annex A.21 with the following input and output parameters

   - out *initialHTTPSState* - initial HTTPS State

   - out *certPathID* - removed Server Certificate Assignment

5. ONVIF Client adds server certification assignment and creates related certification path by following the procedure mentioned in Annex A.25 with the following input and output parameters:

   - in *cap* - DUT capabilities

   - in RSA - key pair algorithm

   - out *certPathID* - certification path

- out *certID* - self-signed certificate
- out *keyID* - RSA key pair

6. ONVIF Client invokes **DeleteKey** with parameters
   - KeyID =: *keyID*
7. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
8. ONVIF Client invokes **DeleteCertificate** with parameters
   - CertificateID =: *certID*
9. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
10. ONVIF Client invokes **DeleteCertificationPath** with parameters
    - CertificationPathID =: *certPathID*
11. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
12. ONVIF Client invokes **RemoveServerCertificateAssignment** .
    - CertificationPathID =: *certPathID*
13. The DUT responds with a **RemoveServerCertificateAssignmentResponse** message.
14. ONVIF Client waits for time *operationDelay*.
15. ONVIF Client invokes **DeleteKey** with parameters
    - KeyID =: *keyID*
16. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
17. ONVIF Client invokes **DeleteCertificate** with parameters
    - CertificateID =: *certID*
18. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.
19. ONVIF Client invokes **DeleteCertificationPath** with parameters
    - CertificationPathID =: *certPathID*
20. The DUT responds with a **DeleteCertificationPathResponse** message.
21. ONVIF Client invokes **DeleteKey** with parameters

- KeyID =: *keyID*

22. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.

23. ONVIF Client deletes the self-signed certificate (in *certID*) and related RSA key pair (in *keyID*) by following procedure mentioned in Annex A.7 to restore DUT configuration.

24. ONVIF Client restores HTTPS and Server Certificate Assignment if required by following the procedure mentioned in Annex A.23 with the following input and output parameters

    - in *initialHTTPSState* - initial HTTPS State

    - in *certPathID* - removed Server Certificate Assignment

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

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

5.4.2 TLS Server Certificate – CA

Test Case ID: ADVANCED_SECURITY-4-1-2

Specification Coverage: Key Management (ONVIF Security Configuration Service Specification), Certificate Management (ONVIF Security Configuration Service Specification), TLS Server (ONVIF Security Configuration Service Specification)

Feature under test: DeleteKey, DeleteCertificate. DeleteCertificationPath

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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. Network Configuration is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client disables HTTPS and removes Server Certificate Assignment if required by following the procedure mentioned in Annex A.21 with the following input and output parameters
   - out *initialHTTPSState* - initial HTTPS State
   - out *certPathID* - removed Server Certificate Assignment
5. ONVIF Client creates a certification path based on CA-signed certificate and related RSA key pair and a corresponding CA certificate and related RSA key pair by following the procedure mentioned in Annex A.19 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in RSA - CA key pair algorithm
   - in RSA - cert key pair algorithm
   - out *certID2* - CA certificate
   - out *keyID2* - CA certificate RSA key pair
   - out *keyID1* - RSA key pair
   - out *certID1* - CA-signed certificate
   - out *certPathID* - certification path

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

23. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.

24. ONVIF Client invokes **DeleteKey** with parameters

    - KeyID =: *keyID2*

25. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.

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

    - CertificateID =: *certID1*

27. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.

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

    - CertificateID =: *certID2*

29. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.

30. ONVIF Client invokes **DeleteCertificationPath** with parameters

    - CertificationPathID =: *certPathID*

31. The DUT responds with a **DeleteCertificationPathResponse** message.

32. ONVIF Client invokes **DeleteKey** with parameters

    - KeyID =: *keyID1*

33. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.

34. ONVIF Client invokes **DeleteKey** with parameters

    - KeyID =: *keyID2*

35. The DUT returns env:Sender/ter:InvalidArgVal/ter:ReferenceExists SOAP 1.2 fault.

36. ONVIF Client deletes the self-signed certificate (in *certID1*) and related RSA key pair (in *keyID1*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.

37. ONVIF Client deletes the self-signed certificate (in *certID2*) and related RSA key pair (in *keyID2*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.

38. ONVIF Client restores HTTPS and Server Certificate Assignment if required by following the procedure mentioned in Annex A.23 with the following input and output parameters

    - in *initialHTTPSState* - initial HTTPS State

    - in *certPathID* - removed Server Certificate Assignment

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.5 Capabilities

5.5.1 Security Configuration Service Capabilities

Test Case ID: ADVANCED_SECURITY-5-1-1

Specification Coverage: Capabilities (ONVIF Security Configuration Service Specification)

Feature under test: GetServiceCapabilities (for Security Configuration Service)

WSDL Reference: security.wsdl

Test Purpose: To verify DUT Security Configuration Service Capabilities.

Pre-Requisite: Security Configuration Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

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.ECCKeyPairGeneration = true:
   - 8.1. If *cap*.KeystoreCapabilities.EllipticCurves is empty or skipped, FAIL the test.
   - 8.2. If *cap*.KeystoreCapabilities.MaximumNumberOfKeys <= 0 or skipped, FAIL the test.
9. If *cap*.KeystoreCapabilities.PKCS8RSAKeyPairUpload = true:
   - 9.1. If *cap*.KeystoreCapabilities.MaximumNumberOfKeys < 1 or skipped, FAIL the test.
   - 9.2. If *cap*.KeystoreCapabilities.RSAKeyLenghts is empty or skipped, FAIL the test.
   - 9.3. If *cap*.KeystoreCapabilities.PasswordBasedEncryptionAlgorithms is empty or skipped, FAIL the test.
   - 9.4. If *cap*.KeystoreCapabilities.PasswordBasedEncryptionAlgorithms does not contain "pbeWithSHAAnd3-KeyTripleDES-CBC" item, FAIL the test.
10. If *cap*.KeystoreCapabilities.PKCS8 = true:
    - 10.1. If *cap*.KeystoreCapabilities.MaximumNumberOfKeys < 1 or skipped, FAIL the test.
    - 10.2. If (*cap*.KeystoreCapabilities.RSAKeyLenghts is empty or skipped) or (*cap*.KeystoreCapabilities.EllipticCurves is empty or skipped), FAIL the test.
    - 10.3. If *cap*.KeystoreCapabilities.PasswordBasedEncryptionAlgorithms is empty or skipped, FAIL the test.
    - 10.4. If *cap*.KeystoreCapabilities.PasswordBasedEncryptionAlgorithms does not contain "pbeWithSHAAnd3-KeyTripleDES-CBC" item, FAIL the test.

11. If *cap*.KeystoreCapabilities.PKCS12CertificateWithRSAPrivateKeyUpload = true:

    11.1. If *cap*.KeystoreCapabilities.MaximumNumberOfKeys < 2 or skipped, FAIL the test.

    11.2. If *cap*.KeystoreCapabilities.MaximumNumberOfCertificates < 2 or skipped, FAIL the test.

    11.3. If *cap*.KeystoreCapabilities.MaximumNumberOfCertificationPaths <= 0 or skipped, FAIL the test.

    11.4. If *cap*.KeystoreCapabilities.SignatureAlgorithms list is empty, FAIL the test.

    11.5. If *cap*.KeystoreCapabilities.RSAKeyLenghts is empty or skipped, FAIL the test.

    11.6. If *cap*.KeystoreCapabilities.PasswordBasedEncryptionAlgorithms is empty or skipped, FAIL the test.

    11.7. If *cap*.KeystoreCapabilities.PasswordBasedEncryptionAlgorithms does not contain "pbeWithSHAAnd3-KeyTripleDES-CBC" item, FAIL the test.

    11.8. If *cap*.KeystoreCapabilities.PasswordBasedMACAlgorithms is empty or skipped, FAIL the test.

    11.9. If *cap*.KeystoreCapabilities.PasswordBasedMACAlgorithms does not contain "hmacWithSHA256" item, FAIL the test.

    11.10. If *cap*.KeystoreCapabilities.X509Versions is empty or skipped, FAIL the test.

    11.11. If *cap*.KeystoreCapabilities.X509Versions does not contain "3" item, FAIL the test.

12. If *cap*.KeystoreCapabilities.PKCS12 = true:

    12.1. If *cap*.KeystoreCapabilities.MaximumNumberOfKeys < 2 or skipped, FAIL the test.

    12.2. If *cap*.KeystoreCapabilities.MaximumNumberOfCertificates < 2 or skipped, FAIL the test.

    12.3. If *cap*.KeystoreCapabilities.MaximumNumberOfCertificationPaths <= 0 or skipped, FAIL the test.

    12.4. If *cap*.KeystoreCapabilities.SignatureAlgorithms list is empty, FAIL the test.

    12.5. If (*cap*.KeystoreCapabilities.RSAKeyLenghts is empty or skipped) or (*cap*.KeystoreCapabilities.EllipticCurves is empty or skipped), FAIL the test.

    12.6. If *cap*.KeystoreCapabilities.PasswordBasedEncryptionAlgorithms is empty or skipped, FAIL the test.

12.7. If *cap*.KeystoreCapabilities.PasswordBasedEncryptionAlgorithms does not contain "pbeWithSHAAnd3-KeyTripleDES-CBC" item, FAIL the test.

12.8. If *cap*.KeystoreCapabilities.PasswordBasedMACAlgorithms is empty or skipped, FAIL the test.

12.9. If *cap*.KeystoreCapabilities.PasswordBasedMACAlgorithms does not contain "hmacWithSHA256" item, FAIL the test.

12.10. If *cap*.KeystoreCapabilities.X509Versions is empty or skipped, FAIL the test.

12.11. If *cap*.KeystoreCapabilities.X509Versions does not contain "3" item, FAIL the test.

13. If *cap*.KeystoreCapabilities.PKCS10ExternalCertificationWithRSA = true:

13.1. If (*cap*.KeystoreCapabilities.RSAKeyPairGeneration = false or skipped) and (*cap*.KeystoreCapabilities.PKCS8RSAKeyPairUpload = false or skipped) and (*cap*.KeystoreCapabilities.PKCS12CertificateWithRSAPrivateKeyUpload = false or skipped), FAIL the test.

13.2. If *cap*.KeystoreCapabilities.SignatureAlgorithms list is empty, FAIL the test.

13.3. If *cap*.KeystoreCapabilities.MaximumNumberOfKeys < 2 or skipped, FAIL the test.

13.4. If *cap*.KeystoreCapabilities.MaximumNumberOfCertificates < 2 or skipped, FAIL the test.

13.5. If *cap*.KeystoreCapabilities.MaximumNumberOfCertificationPaths <= 0 or skipped, FAIL the test.

14. If *cap*.KeystoreCapabilities.PKCS10 = true:

14.1. If (*cap*.KeystoreCapabilities.RSAKeyPairGeneration = false or skipped) and (*cap*.KeystoreCapabilities.ECCKeyPairGeneration = false or skipped) and (*cap*.KeystoreCapabilities.PKCS8 = false or skipped) and (*cap*.KeystoreCapabilities.PKCS12 = false or skipped), FAIL the test.

14.2. If *cap*.KeystoreCapabilities.SignatureAlgorithms list is empty, FAIL the test.

14.3. If *cap*.KeystoreCapabilities.MaximumNumberOfKeys < 2 or skipped, FAIL the test.

14.4. If *cap*.KeystoreCapabilities.MaximumNumberOfCertificates < 2 or skipped, FAIL the test.

14.5. If *cap*.KeystoreCapabilities.MaximumNumberOfCertificationPaths <= 0 or skipped, FAIL the test.

15. If *cap*.KeystoreCapabilities.SelfSignedCertificateCreationWithRSA = true:

    15.1. If (*cap*.KeystoreCapabilities.RSAKeyPairGeneration = false or skipped) and (*cap*.KeystoreCapabilities.PKCS8RSAKeyPairUpload = false or skipped) and (*cap*.KeystoreCapabilities.PKCS12CertificateWithRSAPrivateKeyUpload = false or skipped), FAIL the test.

    15.2. If *cap*.KeystoreCapabilities.MaximumNumberOfCertificates <= 0 or skipped, FAIL the test.

    15.3. If *cap*.KeystoreCapabilities.SignatureAlgorithms list is empty, FAIL the test.

16. If *cap*.KeystoreCapabilities.SelfSignedCertificateCreation = true:

    16.1. If (*cap*.KeystoreCapabilities.RSAKeyPairGeneration = false or skipped) and (*cap*.KeystoreCapabilities.ECCKeyPairGeneration = false or skipped) and (*cap*.KeystoreCapabilities.PKCS8 = false or skipped) and (*cap*.KeystoreCapabilities.PKCS12 = false or skipped), FAIL the test.

    16.2. If *cap*.KeystoreCapabilities.MaximumNumberOfCertificates <= 0 or skipped, FAIL the test.

    16.3. If *cap*.KeystoreCapabilities.SignatureAlgorithms list is empty, FAIL the test.

17. If *cap*.KeystoreCapabilities.MaximumNumberOfCertificationPathValidationPolicies > 0:

    17.1. If (*cap*.KeystoreCapabilities.SelfSignedCertificateCreationWithRSA = false or skipped) and (*cap*.KeystoreCapabilities.PKCS10ExternalCertificationWithRSA = false or skipped) and (*cap*.KeystoreCapabilities.PKCS12CertificateWithRSAPrivateKeyUpload = false or skipped), FAIL the test.

18. If *cap*.TLSServerCapabilities.TLSServerSupported is not empty:

    18.1. If *cap*.TLSServerCapabilities.TLSServerSupported does not contain at least one value, FAIL the test.

    18.2. If *cap*.KeystoreCapabilities.MaximumNumberOfCertificationPaths < 2 or skipped, FAIL the test.

    18.3. If *cap*.TLSServerCapabilities.MaximumNumberOfTLSCertificationPaths <= 0 or skipped, FAIL the test.

    18.4. If (*cap*.KeystoreCapabilities.PKCS10ExternalCertificationWithRSA = false or skipped) and (*cap*.KeystoreCapabilities.SelfSignedCertificateCreationWithRSA = false or skipped), FAIL the test.

19. If *cap*.TLSServerCapabilities.TLSClientAuthSupported = true:

    19.1. If *cap*.TLSServerCapabilities.TLSServerSupported is empty, FAIL the test.

    19.2. If *cap*.KeystoreCapabilities.MaximumNumberOfCertificationPathValidationPolicies < 2 or skipped, FAIL the test.

    19.3. If *cap*.TLSServerCapabilities.MaximumNumberOfTLSCertificationPathValidationPolicies <= 0 or skipped, FAIL the test.

20. If *cap*.TLSServerCapabilities.MaximumNumberOfTLSCertificationPathValidationPolicies > 0:

    20.1. If *cap*.KeystoreCapabilities.MaximumNumberOfCertificationPathValidationPolicies <= 0 or skipped, FAIL the test.

21. If *cap*.TLSServerCapabilities.TLSServerSupported is not empty and *cap*.KeystoreCapabilities.PKCS10ExternalCertificationWithRSA = true:

    21.1. If *cap*.KeystoreCapabilities.MaximumNumberOfCertificates < 3 or skipped, FAIL the test.

22. If *cap*.TLSServerCapabilities.MaximumNumberOfTLSCertificationPaths > 0:

    22.1. If *cap*.KeystoreCapabilities.MaximumNumberOfCertificationPaths <= 0 or skipped, FAIL the test.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.5.2 Get Services and Get Security Configuration Service Capabilities Consistency

Test Case ID: ADVANCED_SECURITY-5-1-2

Specification Coverage: Capability exchange (ONVIF Core Specification), Capabilities (ONVIF Security Configuration Service Specification)

Feature under test: GetServices, GetServiceCapabilities (for Security Configuration Service)

WSDL Reference: devicemgmt.wsdl, security.wsdl

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

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices**.
   - IncludeCapability =: true
4. The DUT responds with a **GetServicesResponse** message with parameters
   - Services list =: *servicesList*
5. ONVIF Client selects Service with Service.Namespace = "http://www.onvif.org/ver10/advancedsecurity/wsdl":
   - Services list [Namespace = "http://www.onvif.org/ver10/advancedsecurity/wsdl"] =: *securityConfigurationService*
6. ONVIF Client invokes **GetServiceCapabilities**.
7. The DUT responds with a **GetServiceCapabilitiesResponse** message with parameters
   - Capabilities =: *cap*
8. If cap differs from *securityConfigurationService*.Capabilities.Capabilities, FAIL the test.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Note: The following fields are compared at step 8:

- KeystoreCapabilities:
  - SignatureAlgorithms
    - algorithm
    - parameters
  - MaximumNumberOfKeys
  - MaximumNumberOfCertificates
  - MaximumNumberOfCertificationPaths
  - RSAKeyPairGeneration
  - ECCKeyPairGeneration
  - RSAKeyLengths
  - EllipticCurves
  - PKCS10ExternalCertificationWithRSA
  - PKCS10
  - SelfSignedCertificateCreationWithRSA
  - SelfSignedCertificateCreation
  - X509Versions
  - MaximumNumberOfPassphrases
  - PKCS8RSAKeyPairUpload
  - PKCS8
  - PKCS12CertificateWithRSAPrivateKeyUpload
  - PKCS12
  - PasswordBasedEncryptionAlgorithms
  - PasswordBasedMACAlgorithms
  - MaximumNumberOfCRLs

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

5.6 Off-Device Key Generation Operations

5.6.1 Passphrase Management

5.6.1.1 Upload Passphrase

Test Case ID: ADVANCED_SECURITY-6-1-1

Specification Coverage: Passphrase Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadPassphrase (for Security Configuration Service)

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client generates an encryption passphrase *passphrase1* (see Annex A.27).

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 **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.40 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

5.6.1.2 Delete Passphrase

Test Case ID: ADVANCED_SECURITY-6-1-2

Specification Coverage: Passphrase Management (ONVIF Security Configuration Service Specification)

Feature under test: DeletePassphrase

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client generates an encryption passphrase *passphrase1* (see Annex A.27).
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 –

- DUT passes all assertions.

FAIL –

- DUT did not send **UploadPassphraseResponse** message.
- DUT did not send **GetAllPassphrasesResponse** message.
- DUT did not send **DeletePassphraseResponse** message.

5.6.2 Key Management

5.6.2.1 Upload PKCS8 – no key pair exists

Test Case ID: ADVANCED_SECURITY-6-2-1

Specification Coverage: Key Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadKeyPairInPKCS8

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration 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 PKCS8 or PKCS8RSAKeyPairUpload capability. The DUT shall have enough free storage capacity for one additional RSA key pair.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client generates a PKCS#8 data structure with new RSA key pair by following the procedure mentioned in Annex A.41 with the following input and output parameters:
   - in *cap* - Security Configuration Service Capabilities
   - in RSA - key pair algorithm
   - out *keyPairInPKCS8* - PKCS#8 data structure

   - out *keyPair* - key pair with public and private keys

5. ONVIF Client invokes **UploadKeyPairInPKCS8** with parameters
   - KeyPair := *keyPairInPKCS8*
   - Alias := "ONVIF_Test"
   - EncryptionPassphraseID skipped
6. The DUT responds with a **UploadKeyPairInPKCS8Response** message with parameters
   - KeyID =: *keyID*
7. ONVIF Client invokes **GetAllKeys**.
8. The DUT responds with a **GetAllKeysResponse** message with parameters
   - KeyAttribute list =: *keyList*
9. If *keyList* does not contain KeyAttribute.KeyID =: *keyID*, FAIL the test, and go to the step 11.
10. 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.
11. If KeyAttribute from *keyList* with KeyAttribute.KeyID =: *keyID* has KeyAttribute.KeyStatus value other than "ok", FAIL the test, and go to the step 11.
12. ONVIF Client deletes the RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.2 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **UploadKeyPairInPKCS8Response** message.
- DUT did not send **GetAllKeysResponse** message.

5.6.2.2 Upload Encrypted PKCS8

Test Case ID: ADVANCED_SECURITY-6-2-3

Specification Coverage: Key Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadKeyPairInPKCS8

WSDL Reference: security.wsdl

Test Purpose: To verify that PKCS#8 data structure can be uploaded correctly if passphrase specified. To verify that a DecryptionFailed fault is produced when wrong decryption passphrase is used.

Pre-Requisite: Security Configuration 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 PKCS8 or PKCS8RSAKeyPairUpload capability. The DUT shall have enough free storage capacity for one additional RSA key pair.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client generates an encryption passphrases *passphrase1* and *passphrase2* (see Annex A.27)
5. ONVIF Client generates a PKCS#8 data structure with new RSA key pair with encryption passphraseby following the procedure mentioned in Annex A.43 with the following input and output parameters:
   - in *cap* - Security Configuration Service Capabilities
   - in RSA - key pair algorithm
   - in *passphrase1* - encryption passphrase
   - out *keyPairInPKCS8* - PKCS#8 data structure
   - out *keyPair* - key pair with public and private keys
6. ONVIF Client invokes **UploadKeyPairInPKCS8** with parameters

- KeyPair := *keyPairInPKCS8*
- Alias := "ONVIF_Test"
- EncryptionPassphrase := *passphrase2*

7. The DUT returns **env:Sender/ter:InvalidArgVal/ter:DecryptionFailed** SOAP 1.2 fault.
8. ONVIF Client invokes **UploadKeyPairInPKCS8** with parameters
   - KeyPair := *keyPairInPKCS8*
   - Alias := "ONVIF_Test"
   - EncryptionPassphrase := *passphrase1*
9. The DUT responds with a **UploadKeyPairInPKCS8Response** message with parameters
   - KeyID
10. ONVIF Client restores DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send the **env:Sender/ter:InvalidArgVal/ter:DecryptionFailed** SOAP 1.2 fault message at step 7.
- DUT did not send **UploadKeyPairInPKCS8Response** message at step 9.

5.6.2.3 Upload Encrypted PKCS8 – Using Passphrase Management

Test Case ID: ADVANCED_SECURITY-6-2-4

Specification Coverage: Key Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadKeyPairInPKCS8, UploadPassphrase

WSDL Reference: security.wsdl

Test Purpose: To verify that PKCS#8 data structure can be uploaded correctly if passphrase specified. To verify that a DecryptionFailed fault is produced when wrong decryption passphrase is used. To verify work of UploadKeyPairInPKCS8 with passphrase management.

Pre-Requisite: Security Configuration 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 PKCS8 or PKCS8RSAKeyPairUpload capability. Passphrase Management is supported by the DUT as indicated by the MaximumNumberOfPassphrases 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 passphrases.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client generates an encryption passphrases *passphrase1* and *passphrase2* (see Annex A.27)
5. ONVIF Client generates a PKCS#8 data structure with new RSA key pair with encryption passphrase by following the procedure mentioned in Annex A.43 with the following input and output parameters:
   - in *cap* - Security Configuration Service Capabilities
   - in RSA - key pair algorithm
   - in *passphrase1* - encryption passphrase
   - out *keyPairInPKCS8* - PKCS#8 data structure
   - out *keyPair* - key pair with public and private keys
6. ONVIF Client invokes **UploadPassphrase** with parameters
   - Passphrase := *passphrase1*
   - KeyAlias := "ONVIF_Passphrase_Test1"
7. The DUT responds with a **UploadPassphraseResponse** message with parameters

- PassphraseID =: *passphraseID1*

8. ONVIF Client invokes **UploadPassphrase** with parameters

    - Passphrase := *passphrase2*

    - KeyAlias := "ONVIF_Passphrase_Test2"

9. The DUT responds with a **UploadPassphraseResponse** message with parameters

    - PassphraseID =: *passphraseID2*

10. ONVIF Client invokes **UploadKeyPairInPKCS8** with parameters

    - KeyPair := *keyPairInPKCS8*

    - Alias := "ONVIF_Test"

    - EncryptionPassphraseID := *passphraseID2*

11. The DUT returns **env:Sender/ter:InvalidArgVal/ter:DecryptionFailed** SOAP 1.2 fault.

12. ONVIF Client invokes **UploadKeyPairInPKCS8** with parameters

    - KeyPair := *keyPairInPKCS8*

    - Alias := "ONVIF_Test"

    - EncryptionPassphraseID := *passphraseID1*

13. The DUT responds with a **UploadKeyPairInPKCS8Response** message with parameters

    - KeyID

14. ONVIF Client restores DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

- The DUT did not send the **env:Sender/ter:InvalidArgVal/ter:DecryptionFailed** SOAP 1.2 fault message at step 11.

- DUT did not send **UploadKeyPairInPKCS8Response** message at step 13.

5.6.3 Certificate Management

5.6.3.1 Upload PKCS12 – no key pair exists

Test Case ID: ADVANCED_SECURITY-6-3-1

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadCertificateWithPrivateKeyInPKCS12

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration 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 PKCS12 or 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client generates an encryption passphrase *passphrase1* (see Annex A.27).
5. ONVIF Client creates a CA certificate and a corresponding public key in the certificate along with the corresponding private key in the form of a PKCS#12 file with given passphrase by following the procedure described in Annex A.28 with the following input and output parameters:
   - in *cap* - DUT capabilities

- in RSA - key pair algorithm
- in *passphrase1* - given passphrase
- out *CAcert* - CA certificate
- out *keyPair* - key pair with public and private keys
- out *PKCS12data* - PKCS#12 file

6. ONVIF Client invokes **UploadCertificateWithPrivateKeyInPKCS12** with parameters
    - CertWithPrivateKey := *PKCS12data*
    - CertificationPathAlias := "ONVIF_Certification_Path_Test"
    - KeyAlias := "ONVIF_Key_Test"
    - IgnoreAdditionalCertificates := true
    - IntegrityPassphraseID skipped
    - EncryptionPassphraseID skipped
    - Passphrase := *passphrase1*
7. The DUT responds with a **UploadCertificateWithPrivateKeyInPKCS12Response** message with parameters
    - CertificationPathID =: *certPathID*
    - KeyID =: *keyID*
8. ONVIF Client invokes **GetAllKeys**.
9. The DUT responds with a **GetAllKeysResponse** message with parameters
    - KeyAttribute list =: *keyList*
10. If *keyList* does not contain key with KeyID equal to *keyID*, FAIL the test and restore configuration.
11. If key with KeyID equal to *keyID* from *keyList* has hasPrivateKey equal to false or has skipped hasPrivateKey, FAIL the test and restore configuration.
12. If key with KeyID equal to *keyID* from *keyList* has Alias skipped or other than "ONVIF_Key_Test", FAIL the test and restore configuration.

13. If key with KeyID equal to *keyID* from *keyList* has KeyStatus other than "ok", FAIL the test and restore configuration.

14. ONVIF Client invokes **GetAllCertificationPaths**.

15. The DUT responds with a **GetAllCertificationPathsResponse** message with parameters

    - CertificationPathID list =: *certPathList*

16. If *certPathList* does not contain certification path with CertificationPathID equal to *certPathID*, FAIL the test and restore configuration.

17. ONVIF Client invokes **GetCertificationPath** message with parameters

    - CertificationPathID =: *certPathID*

18. The DUT responds with a **GetCertificationPathResponse** message with parameters

    - CertificationPath.CertificateID[0] =: *certID*
    - CertificationPath.Alias =: *CertPathAlias*

19. If *CertPathAlias* Alias skipped or other than "ONVIF_CertificationPath_Test", FAIL the test and restore configuration.

20. If received CertificationPath contains more than one CertificateID item, FAIL the test and restore configuration.

21. ONVIF Client deletes the certification path (in *certPathID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.30 to restore DUT configuration and finish the test.

22. 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.15 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **UploadCertificateWithPrivateKeyInPKCS12Response** message.
- DUT did not send **GetAllKeysResponse** message.

- DUT did not send **GetAllCertificationPathsResponse** message.
- DUT did not send **GetCertificationPathResponse** message.

5.6.3.2 Upload PKCS12 - verify key and certificate

Test Case ID: ADVANCED_SECURITY-6-3-4

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadCertificateWithPrivateKeyInPKCS12, GetKeyStatus, GetCertificate, GetAllCertificates

WSDL Reference: security.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: Security Configuration 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 PKCS12 or 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client generates an encryption passphrase *passphrase1* (see Annex A.27).
5. ONVIF Client creates a CA certificate and a corresponding public key in the certificate along with the corresponding private key in the form of a PKCS#12 file with given passphrase by following the procedure described in Annex A.28 with the following input and output parameters:
   - in *cap* - DUT capabilities

- in RSA - key pair algorithm
- in *passphrase1* - given passphrase
- out *CAcert* - CA certificate
- out *keyPair* - key pair with public and private keys
- out *PKCS12data* - PKCS#12 file

6. ONVIF Client invokes **UploadCertificateWithPrivateKeyInPKCS12** with parameters
   - CertWithPrivateKey := *PKCS12data*
   - CertificationPathAlias := "ONVIF_Certification_Path_Test"
   - KeyAlias := "ONVIF_Key_Test"
   - IgnoreAdditionalCertificates skipped
   - IntegrityPassphraseID skipped
   - EncryptionPassphraseID skipped
   - Passphrase := *passphrase1*
7. The DUT responds with a **UploadCertificateWithPrivateKeyInPKCS12Response** message with parameters
   - CertificationPathID =: *certPathID*
   - KeyID =: *keyID*
8. ONVIF Client invokes **GetKeyStatus** with parameters
   - KeyID := *keyID*
9. The DUT responds with **GetKeyStatusResponse** message with parameters
   - KeyStatus =: *keyStatus*
10. If *keyStatus* is not equal to "ok", FAIL the test, and go to the step 23.
11. ONVIF Client invokes **GetCertificationPath** message with parameters
    - CertificationPathID =: *certPathID*
12. The DUT responds with a **GetCertificationPathResponse** message with parameters

- CertificationPath.CertificateID list =: *certIDList*
- CertificationPath.Alias

13. If *certIDList* contains more item than one, FAIL the test and go to the step 23.
14. ONVIF Client invokes **GetAllCertificates**.
15. The DUT responds with a **GetAllCertificatesResponse** message with parameters
    - CertificateID list =: *certificateList*
16. If *certificateList* does not contain certificate with Certificate.CertificateID equal to *certIDList*[0], FAIL the test and go to the step 23.
17. Set:
    - *certificateList*.Certificate[CertificateID = *certIDList*[0] =: *X509Cert*
18. If *X509Cert*.KeyID is not equal to *keyID*, FAIL the test and go to the step 23.
19. If *X509Cert*.CertificateContent is not equal to *CAcert*, FAIL the test and go to the step 23.
20. ONVIF Client invokes **GetCertificate** message with parameters
    - CertificateID := *certIDList*[0]
21. The DUT responds with a **GetCertificateResponse** message with parameters
    - Certificate =: *X509Cert*
22. If *X509Cert*.CertificateID is not equal to *certIDList*[0], FAIL the test and go to the step 23.
23. If *X509Cert*.KeyID is not equal to *keyID*, FAIL the test and go to the step 23.
24. If *X509Cert*.CertificateContent is not equal to *CAcert*, FAIL the test and go to the step 23.
25. ONVIF Client deletes the certification path (in *certPathID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.30 to restore DUT configuration and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **UploadCertificateWithPrivateKeyInPKCS12Response** message.
- DUT did not send **GetKeyStatusResponse** message.
- DUT did not send **GetCertificationPathResponse** message.
- DUT did not send **GetAllCertificatesResponse** message.
- DUT did not send **GetCertificateResponse** message.

5.6.3.3 Upload PKCS12 – Decryption Failed

Test Case ID: ADVANCED_SECURITY-6-3-5

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadCertificateWithPrivateKeyInPKCS12

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration 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 PKCS12 or 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
    - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client generates an encryption passphrases *passphrase1* and *passphrase2* (see Annex A.27).
5. ONVIF Client creates a CA certificate and a corresponding public key in the certificate along with the corresponding private key in the form of a PKCS#12 file with given passphrase

by following the procedure described in Annex A.28 with the following input and output parameters:

- in *cap* - DUT capabilities
- in RSA - key pair algorithm
- in *passphrase1* - given passphrase
- out *CAcert* - CA certificate
- out *keyPair* - key pair with public and private keys
- out *PKCS12data* - PKCS#12 file

6. ONVIF Client invokes **UploadCertificateWithPrivateKeyInPKCS12** message with parameters
   - CertWithPrivateKey := *PKCS12data*
   - CertificationPathAlias := "ONVIF_CertificationPath_Test"
   - KeyAlias := "ONVIF_Key_Test"
   - IgnoreAdditionalCertificates := false
   - IntegrityPassphraseID skipped
   - EncryptionPassphraseID skipped
   - Passphrase := *passphrase2*
7. The DUT returns **env:Sender\ter:InvalidArgVal\ter:DecryptionFailed** SOAP 1.2 fault.
8. ONVIF Client restores DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **UploadPassphrase** message.
- DUT did not send **env:Sender\ter:InvalidArgVal\ter:DecryptionFailed** SOAP 1.2 fault at step 7.

5.6.3.4 Upload PKCS12 – Using Passphrase Storage

Test Case ID: ADVANCED_SECURITY-6-3-6

Specification Coverage: Certificate Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadCertificateWithPrivateKeyInPKCS12, UploadPassphrase

WSDL Reference: security.wsdl

Test Purpose: To verify that a PKCS#12 data structure can be uploaded correctly if passphrase specified. To verify that a DecryptionFailed fault is produced when wrong decryption passphrase is used. To verify work of UploadCertificateWithPrivateKeyInPKCS12 with passphrase management.

Pre-Requisite: Security Configuration 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 PKCS12 or PKCS12CertificateWithRSAPrivateKeyUpload capability. Passphrase Management is supported by the DUT as indicated by the MaximumNumberOfPassphrases 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 two additional passphrases.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client generates an encryption passphrases *passphrase1* and *passphrase2* (see Annex A.27).

5. ONVIF Client creates a CA certificate and a corresponding public key in the certificate along with the corresponding private key in the form of a PKCS#12 file with given passphrase by following the procedure described in Annex A.28 with the following input and output parameters:

   - in *cap* - DUT capabilities

- in RSA - key pair algorithm
- in *passphrase1* - given passphrase
- out *CAcert* - CA certificate
- out *keyPair* - key pair with public and private keys
- out *PKCS12data* - PKCS#12 file

6. ONVIF Client invokes **UploadPassphrase** with parameters
    - Passphrase =: *passphrase1*
    - KeyAlias := "ONVIF_Passphrase_Test1"
7. The DUT responds with a **UploadPassphraseResponse** message with parameters
    - PassphraseID =: *passphraseID1*
8. ONVIF Client invokes **UploadPassphrase** with parameters
    - Passphrase =: *passphrase2*
    - KeyAlias := "ONVIF_Passphrase_Test2"
9. The DUT responds with a **UploadPassphraseResponse** message with parameters
    - PassphraseID =: *passphraseID2*
10. ONVIF Client invokes **UploadCertificateWithPrivateKeyInPKCS12** message with parameters
    - CertWithPrivateKey := *PKCS12data*
    - CertificationPathAlias := "ONVIF_CertificationPath_Test"
    - KeyAlias := "ONVIF_Key_Test"
    - IgnoreAdditionalCertificates := false
    - IntegrityPassphraseID skipped
    - EncryptionPassphraseID =: *passphraseID2*
11. The DUT returns **env:Sender\ter:InvalidArgVal\ter:DecryptionFailed** SOAP 1.2 fault.
12. ONVIF Client invokes **UploadCertificateWithPrivateKeyInPKCS12** message with parameters

- CertWithPrivateKey := *PKCS12data*
- CertificationPathAlias := "ONVIF_CertificationPath_Test"
- KeyAlias := "ONVIF_Key_Test"
- IgnoreAdditionalCertificates := false
- IntegrityPassphraseID skipped
- EncryptionPassphraseID =: *passphraseID1*

13. The DUT responds with a **UploadCertificateWithPrivateKeyInPKCS12Response** message with parameters
    - CertificationPathID
    - KeyID
14. ONVIF Client restores DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **UploadPassphrase** message.
- DUT did not send **UploadCertificateWithPrivateKeyInPKCS12** message at step 13.
- DUT did not send **env:Sender\ter:InvalidArgVal\ter:DecryptionFailed** SOAP 1.2 fault at step 11.

5.7 Certificate-based Client Authentication

5.7.1 Upload CRL

Test Case ID: ADVANCED_SECURITY-8-1-1

Specification Coverage: CRL Management (ONVIF Security Configuration Service Specification)

Feature under test: UploadCRL, GetAllCRLs

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client creates a CRL by following the procedure mentioned in Annex A.45 with the following input and output parameters:
   - in *cap* - Security Configuration Service Capabilities
   - in RSA - key pair algorithm
   - out *crl* - CRL
5. ONVIF Client invokes **UploadCRL** with parameter
   - Crl =: *crl*
   - Alias := "ONVIF_CRL_Test"
   - anyParameters skipped
6. The DUT responds with a **UploadCRLResponse** message with parameters
   - CrlID =: *crlID*
7. ONVIF Client invokes **GetAllCRLs**
8. The DUT responds with a **GetAllCRLsResponse** message with parameters
   - CrlID list =: *crlList*
9. If *crlList* does not contain *crlID*, FAIL the test and restore configuration.

10. If *crlList*[CRLID = *crlID*].Alias is not equal to "ONVIF_CRL_Test", FAIL the test and restore configuration.

11. If *crlList*[CRLID = *crlID*].CRLContent is not equal to *crl*, FAIL the test and restore configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **UploadCRLResponse** message.
- DUT did not send **GetAllCRLsResponse** message.

5.7.2 Delete CRL

Test Case ID: ADVANCED_SECURITY-8-1-2

Specification Coverage: CRL Management (ONVIF Security Configuration Service Specification)

Feature under test: DeleteCRL, GetAllCRLs

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration 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 Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

    - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client creates a CRL by following the procedure mentioned in Annex A.45 with the following input and output parameters:
   - in *cap* - Security Configuration Service Capabilities
   - in RSA - key pair algorithm
   - out *crl* - CRL
5. ONVIF Client uploads a CRL (in *crl*) with alias (in "ONVIF_CRL_Test") identifier (out *crlID*) by following the procedure described in Annex A.35.
6. ONVIF Client invokes **DeleteCRL** with parameters
   - CrlID =: *crlID*
7. The DUT responds with a **DeleteCRLResponse** message.
8. ONVIF Client invokes **GetAllCRLs**
9. The DUT responds with a **GetAllCRLsResponse** message with parameters
   - CrlID list =: *crlList*
10. If *crlList* contains *crlID*, FAIL the test and restore configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **DeleteCRLResponse** message.
- DUT did not send **GetAllCRLsResponse** message.

5.7.3 Get CRL

Test Case ID: ADVANCED_SECURITY-8-1-3

Specification Coverage: CRL Management (ONVIF Security Configuration Service Specification)

Feature under test: GetCRL

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration 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 Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client creates a CRL by following the procedure mentioned in Annex A.45 with the following input and output parameters:

   - in *cap* - Security Configuration Service Capabilities

   - in RSA - key pair algorithm

   - out *crl* - CRL

5. ONVIF Client uploads a CRL (in *crl*) with alias (in "ONVIF_CRL_Test") identifier (out *crlID*) by following the procedure described in Annex A.35.

6. ONVIF Client invokes **GetCRL** with parameters

   - CrlID =: *crlID*

7. The DUT responds with a **GetCRLResponse** message with parameters

   - Crl =: *crl*

8. If *crl*.CRLID is not equal to *crlID*, FAIL the test and restore configuration.

9. If *crl*.Alias is not equal to "ONVIF_CRL_Test", FAIL the test and restore configuration.

10. If *crl*.CRLContent is not equal to *crl*, FAIL the test and restore configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.7.4 Create certification path validation policy

Test Case ID: ADVANCED_SECURITY-8-1-4

Specification Coverage: Certification Path Validation Policy Management (ONVIF Security Configuration Service Specification)

Feature under test: CreateCertPathValidationPolicy, GetAllCertPathValidationPolicies

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration 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 Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client prepares certificate related RSA key pair and certification path if any by following the procedure mentioned in Annex A.38 with the following input and output parameters:

   - in *cap* - service capabilities

   - in RSA - key pair algorithm

- out *keyID* - key pair ID
- out *certID* - certificate ID
- out *certificationPathID* - certification path ID

5. 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
6. The DUT responds with **CreateCertPathValidationPolicyResponse** message with parameters
   - CertPathValidationPolicyID =: *certPathValidationPolicyID*
7. ONVIF Client invokes **GetAllCertPathValidationPolicies**.
8. The DUT responds with a **GetAllCertPathValidationPoliciesResponse** message with parameters
   - CertPathValidationPolicy list =: *certPathValidationPolicyList*
9. If *certPathValidationPolicyList* does not contain *certPathValidationPolicyID*, FAIL the test, and go to the step 14.
10. If *certPathValidationPolicyList*[CertPathValidationPolicyID = *certPathValidationPolicyID*].Alias is not equal to "ONVIF_CertPathValidationPolicy_Test", FAIL the test, and go to the step 14.
11. If *certPathValidationPolicyList*[CertPathValidationPolicyID = *certPathValidationPolicyID*].Parameters.RequireTLSWWWClientAuthExtendedKeyUsage is equal to true, FAIL the test, and go to the step 14.
12. If *certPathValidationPolicyList*[CertPathValidationPolicyID = *certPathValidationPolicyID*].Parameters.UseDeltaCRLs is not equal to true, FAIL the test, and go to the step 14.

13. If *certPathValidationPolicyList*[CertPathValidationPolicyID = *certPathValidationPolicyID*]. TrustAnchor does not contain one and only one element with CertificateID equal to *certID*, FAIL the test, and go to the step 14.

14. ONVIF Client deletes the certification path validation policy (in *certPathValidationPolicyID*) by following the procedure mentioned in Annex A.33 to restore DUT configuration.

15. If *certificationPathID* is null:

    15.1. 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.7.

    15.2. Skip other steps.

16. If *certificationPathID* is not null:

    16.1. ONVIF Client deletes the certification path (in *certificationPathID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.30 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

5.7.5 Get certification path validation policy

Test Case ID: ADVANCED_SECURITY-8-1-5

Specification Coverage: Certification Path Validation Policy Management (ONVIF Security Configuration Service Specification)

Feature under test: CreateCertPathValidationPolicy, GetCertPathValidationPolicy

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration 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 Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.37 with the following input and output parameters

   - in *cap* - DUT capabilities

   - in RSA - key pair algorithm

   - in "Test CertPathValidationPolicy Alias" - certification path validation policy alias

   - out *certPathValidationPolicyID1* - certification path validation policy identifier

   - out *certificationPathID* - certification path identifier (if any)

   - out *certID* - certificate identifier

   - out *keyID* - RSA key pair identifier

5. ONVIF Client invokes **GetCertPathValidationPolicy** with parameters

   - CertPathValidationPolicyID =: *certPathValidationPolicyID*

6. The DUT responds with a **GetCertPathValidationPolicyResponse** message with parameters

   - CertPathValidationPolicy =: *certPathValidationPolicy*

7. If *certPathValidationPolicy*.CertPathValidationPolicyID is not equal to *certPathValidationPolicyID*, FAIL the test, and go to the step 12.

8. If *certPathValidationPolicy*.Alias is not equal to "Test CertPathValidationPolicy Alias", FAIL the test, and go to the step 12.

9. If *certPathValidationPolicy*.Parameters.RequireTLSWWWClientAuthExtendedKeyUsage is equal to true, FAIL the test, and go to the step 12.

10. If *certPathValidationPolicy*.Parameters.UseDeltaCRLs is not equal to true, FAIL the test, and go to the step 12.

11. If *certPathValidationPolicy*.TrustAnchor does not contain one and only one element with CertificateID equal to *certID*, FAIL the test, and go to the step 12.

12. ONVIF Client deletes the certification path validation policy (in *certPathValidationPolicyID*) by following the procedure mentioned in Annex A.33 to restore DUT configuration.

13. If *certificationPathID* is null:

    13.1. 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.7.

    13.2. Skip other steps.

14. If *certificationPathID* is not null:

    14.1. ONVIF Client deletes the certification path (in *certificationPathID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.30 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.7.6 Delete certification path validation policy

Test Case ID: ADVANCED_SECURITY-8-1-6

Specification Coverage: Certification Path Validation Policy Management (ONVIF Security Configuration Service Specification)

Feature under test: DeleteCertPathValidationPolicy, GetAllCertPathValidationPolicies

WSDL Reference: security.wsdl

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

Pre-Requisite: Security Configuration 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 Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.37 with the following input and output parameters
   - in *cap* - DUT capabilities
   - in RSA - key pair algorithm
   - in "Test CertPathValidationPolicy Alias" - certification path validation policy alias
   - out *certPathValidationPolicyID1* - certification path validation policy identifier
   - out *certificationPathID* - certification path identifier (if any)
   - out *certID* - certificate identifier
   - out *keyID* - RSA key pair identifier
5. ONVIF Client invokes **DeleteCertPathValidationPolicy** with parameters
   - CertPathValidationPolicyID =: *certPathValidationPolicyID*
6. The DUT responds with a **DeleteCertPathValidationPolicyResponse** message.

7. ONVIF Client invokes **GetAllCertPathValidationPolicies**.

8. The DUT responds with a **GetAllCertPathValidationPoliciesResponse** message with parameters

   - CertPathValidationPolicy list =: *certPathValidationPolicyList*

9. If *certPathValidationPolicyList* contains *certPathValidationPolicyID*, FAIL the test, and go to the step 10.

10. If *certificationPathID* is null:

    10.1. 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.7.

    10.2. Skip other steps.

11. If *certificationPathID* is not null:

    11.1. ONVIF Client deletes the certification path (in *certificationPathID*) and RSA key pair (in *keyID*) by following the procedure mentioned in Annex A.30 to restore DUT configuration.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

- DUT did not send **GetAllCertPathValidationPoliciesResponse** .

5.8 TLS Versions

5.8.1 TLS Version Management

Test Case ID: ADVANCED_SECURITY-9-1-1

Specification Coverage: SetEnabledTLSVersions (ONVIF Security Configuration Service Specification), GetEnabledTLSVersions (ONVIF Security Configuration Service Specification)

Feature under test: SetEnabledTLSVersions, GetServiceCapabilities (Security Configuration), GetEnabledTLSVersions

WSDL Reference: security.wsdl

Test Purpose: Verify that TLS Versions settings could be changes by SetEnabledTLSVersions request and recieved by GetEnabledTLSVersions request. Verify that TLS Versions settings are consistant with capabilities.

Pre-Requisite: Security Configuration Service is received from the DUT. Enabling and disabling specific TLS versions supported by the DUT as indicated by the EnabledVersionsSupported capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves Security Configuration Service Capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client invokes **GetEnabledTLSVersions** request.
5. The DUT responds with **GetEnabledTLSVersionsResponse** message with parameters
   - Versions list =: *initialVersionList*
6. If *initialVersionList* does not contains any items, FAIL the test, restore the DUT state, and skip other steps.
7. If *cap*.TLSServerCapabilities.TLSServerSupported does not contains all items from *initialVersionList* list, FAIL the test, restore the DUT state, and skip other steps.
8. ONVIF Client invokes **SetEnabledTLSVersions** request with parameters
   - Versions list := *cap*.TLSServerCapabilities.TLSServerSupported
9. The DUT responds with **SetEnabledTLSVersionsResponse** message.
10. ONVIF Client invokes **GetEnabledTLSVersions** request.
11. The DUT responds with **GetEnabledTLSVersionsResponse** message with parameters
    - Versions list =: *updatedVersionList1*

12. If *updatedVersionList1* is not equal to *cap*.TLSServerCapabilities.TLSServerSupported list (the order of the items shall be ignored during comparition), FAIL the test, restore the DUT state, and skip other steps.

13. ONVIF Client invokes **SetEnabledTLSVersions** request with parameters

    - Versions list := empty list

14. The DUT returns **env:Sender/ter:InvalidArgVal/ter:EmptyList** SOAP 1.2 fault.

15. ONVIF Client invokes **GetEnabledTLSVersions** request.

16. The DUT responds with **GetEnabledTLSVersionsResponse** message with parameters

    - Versions list =: *updatedVersionList2*

17. If *updatedVersionList2* is not equal to *cap*.TLSServerCapabilities.TLSServerSupported list (the order of the items shall be ignored during comparition), FAIL the test, restore the DUT state, and skip other steps.

18. If *cap*.TLSServerCapabilities.TLSServerSupported list contains more than one item:

    18.1. ONVIF Client invokes **SetEnabledTLSVersions** request with parameters

    - Versions list := first item from *cap*.TLSServerCapabilities.TLSServerSupported list

    18.2. The DUT responds with **SetEnabledTLSVersionsResponse** message.

    18.3. ONVIF Client invokes **GetEnabledTLSVersions** request.

    18.4. The DUT responds with **GetEnabledTLSVersionsResponse** message with parameters

    - Versions list =: *updatedVersionList3*

    18.5. If *updatedVersionList3* contains more than one item, FAIL the test, restore the DUT state, and skip other steps.

    18.6. If *updatedVersionList3* does not contain first item from *cap*.TLSServerCapabilities.TLSServerSupported list, FAIL the test, restore the DUT state, and skip other steps.

    18.7. ONVIF Client invokes **SetEnabledTLSVersions** request with parameters

    - Versions list := last item from *cap*.TLSServerCapabilities.TLSServerSupported list

    18.8. The DUT responds with **SetEnabledTLSVersionsResponse** message.

18.9. ONVIF Client invokes **GetEnabledTLSVersions** request.

18.10. The DUT responds with **GetEnabledTLSVersionsResponse** message with parameters

- Versions list =: *updatedVersionList4*

18.11. If *updatedVersionList4* contains more than one item, FAIL the test, restore the DUT state, and skip other steps.

18.12. If *updatedVersionList4* does not contain last item from *cap*.TLSServerCapabilities.TLSServerSupported list, FAIL the test, restore the DUT state, and skip other steps.

19. Restore the DUT state.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetEnabledTLSVersionsResponse** message at steps 9, 17.2, 17.8.
- DUT did not send **SetEnabledTLSVersionsResponse** message.
- DUT did not send the **env:Sender/ter:InvalidArgVal/ter:EmptyList** SOAP 1.2 fault message at step 14.

5.8.2 Disable TLS Version

Test Case ID: ADVANCED_SECURITY-9-1-2

Specification Coverage: SetEnabledTLSVersions (ONVIF Security Configuration Service Specification), GetEnabledTLSVersions (ONVIF Security Configuration Service Specification)

Feature under test: SetEnabledTLSVersions, GetEnabledTLSVersions

WSDL Reference: security.wsdl

Test Purpose: Verify that TLS Versions settings were applied.

Pre-Requisite: Security Configuration Service is received from the DUT. CRLs supported by the DUT as indicated by the MaximumNumberOfCRLs capability. Network Configuration is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves Security Configuration Service Capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters
   - out *cap* - Security Configuration Service Capabilities
4. If *cap*.TLSServerCapabilities.TLSServerSupported contains only one item, skip other steps.
5. ONVIF Client configures HTTPS if required by following the procedure mentioned in Annex A.46 with the following input parameters:
   - in *cap* - DUT capabilities
6. ONVIF Client invokes **SetEnabledTLSVersions** request with parameters
   - Versions list := all items from *cap*.TLSServerCapabilities.TLSServerSupported list except lowest version
7. The DUT responds with **SetEnabledTLSVersionsResponse** message.
8. ONVIF Client verifies basic TLS handshake by following the procedure mentioned in Annex A.49 with the following input and output parameters.
   - in lowest version from *cap*.TLSServerCapabilities.TLSServerSupported list - disabled TLS version
9. ONVIF Client verifies basic TLS handshake by following the procedure mentioned in Annex A.26.
10. Restore the DUT state.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetEnabledTLSVersionsResponse** message.

5.9 Authorization Server Configuration

5.9.1 Authorization Server Configuration Creation

Test Case ID: ADVANCED_SECURITY-10-1-1

Specification Coverage: GetAuthorizationServerConfigurations (ONVIF Security Configuration Service Specification), CreateAuthorizationServerConfiguration (ONVIF Security Configuration Service Specification), DeleteAuthorizationServerConfiguration (ONVIF Security Configuration Service Specification)

Feature under test: GetAuthorizationServerConfigurations, CreateAuthorizationServerConfiguration, DeleteAuthorizationServerConfiguration

WSDL Reference: security.wsdl

Test Purpose: Verify that Authorization Server Configuration could be created by CreateAuthorizationServerConfiguration request, received by GetAuthorizationServerConfigurations request, and deleted by DeleteAuthorizationServerConfiguration request.

Pre-Requisite: Security Configuration Service is received from the DUT. Authorization Server Configuration is supported by the DUT as indicated by the AuthorizationServer.MaxConfigurations capability. The DUT shall have enough free storage capacity for one additional key pair if private_key_jwt is supported by device as indicated by AuthorizationServer.ClientAuthenticationMethodsSupported capability. The DUT shall have enough free storage capacity for one additional certificate if self_signed_tls_client_auth is supported by device as indicated by AuthorizationServer.ClientAuthenticationMethodsSupported capability. The DUT shall have enough free storage capacity for one additional certification path validation policy.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. If *cap*.AuthorizationServer.ClientAuthenticationMethodsSupported contains "private_key_jwt":

4.1. ONVIF Client configures key pair by following the procedure described in Annex A.3 with the following input and output parameters:

- in *cap* - DUT capabilities
- out *keyID1* - key pair identificator

5. If *cap*.AuthorizationServer.ClientAuthenticationMethodsSupported contains "self_signed_tls_client_auth":

5.1. Set:

- *keyAlgorithm* := **"ECC"** if *cap*.KeystoreCapabilities.ECCKeyPairGeneration = true; **"RSA"** otherwise.

5.2. ONVIF Client prepares certificate by following the procedure mentioned in Annex A.38 with the following input and output parameters

- in *cap* - service capabilities
- in *keyAlgorithm* - key pair algorithm
- in "CN=ONVIF TT1,C=US" - CA certificate subject
- out *certificationPathID1* - certification path identifier (if any)
- out *certID2* - certificate identifier
- out *keyID2* - key pair identifier

6. Configure certification path validation policy:

6.1. Set:

- *keyAlgorithm* := **"ECC"** if *cap*.KeystoreCapabilities.ECCKeyPairGeneration = true; **"RSA"** otherwise.

6.2. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.50 with the following input and output parameters

- in *cap* - DUT capabilities
- in *keyAlgorithm* - DUT capabilities
- in "CN=ONVIF TT2,C=US" - CA certificate subject
- in "Test CertPathValidationPolicy1 Alias" - certification path validation policy alias

- out *certPathValidationPolicyID3* - certification path validation policy identifier
- out *certID3* - certificate identifier
- out *keyID3* - RSA key pair identifier
- out *CAcert* - CA certificate
- out *privateKey* - CA certificate private key

7. ONVIF Client deletes one authorization server configuration is maximum is reached by following the procedure described in Annex A.51 with the following input and output parameters:

- in *cap* - DUT capabilities
- out *itemToRestore1* - deleted authorization server configuration if any

8. For each configuration type supported (*configurationType*) in *cap*.AuthorizationServer.ConfigurationTypesSupported:

8.1. For each authentication method supported (*authenticationMethod*) in *cap*.AuthorizationServer.ClientAuthenticationMethodsSupported:

8.1.1. ONVIF Client invokes **CreateAuthorizationServerConfiguration** request with parameters

- Type := *configurationType*
- ClientAuth := *authenticationMethod*
- ServerUri := *anyValidUri1*
- ClientID := *anyClientID1*
- ClientSecret := *anyClientSecret1* if *authenticationMethod* != "private_key_jwt", otherwise is skipped
- Scope := *anyScope1*
- KeyID := *keyID1* if *authenticationMethod* = "private_key_jwt", otherwise is skipped
- CertificateID := *certificateID2* if *authenticationMethod* = "self_signed_tls_client_auth", otherwise is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyID3*

8.1.2. The DUT responds with **CreateAuthorizationServerConfigurationResponse** message with parameters

- Token =: *token1*

8.1.3. ONVIF Client invokes **GetAuthorizationServerConfigurations** request with parameters

- Token =: *token1*

8.1.4. The DUT responds with **GetAuthorizationServerConfigurationsResponse** message with parameters

- Configuration list =: *configurations1*

8.1.5. If *configurations1* does not contain exactly one item with @token equal to *token1*, FAIL the test, restore the DUT state, and skip other steps.

8.1.6. If *configurations1*[0].Data contains parameter ClientSecret, FAIL the test, restore the DUT state, and skip other steps.

8.1.7. If *configurations1*[0].Data does not contain the same parameters as were used at 8.1.1, FAIL the test, restore the DUT state, and skip other steps.

8.1.8. ONVIF Client invokes **GetAuthorizationServerConfigurations** request with parameters

- Token is skipped

8.1.9. The DUT responds with **GetAuthorizationServerConfigurationsResponse** message with parameters

- Configuration list =: *configurations2*

8.1.10. If *configurations2* does not contain item with @token equal to *token1*, FAIL the test, restore the DUT state, and skip other steps.

8.1.11. If *configurations2*[0].Data contains parameter ClientSecret, FAIL the test, restore the DUT state, and skip other steps.

8.1.12. If *configurations2*[0].Data does not contain the same parameters as were used at 8.1.1, FAIL the test, restore the DUT state, and skip other steps.

8.1.13. ONVIF Client invokes **DeleteAuthorizationServerConfiguration** with parameter

- Token := *token1*

8.1.14. The DUT responds with **DeleteAuthorizationServerConfigurationResponse** message.

8.1.15. ONVIF Client invokes **GetAuthorizationServerConfigurations** request with parameters

- Token =: *token1*

8.1.16. The DUT returns **env:Sender/ter:InvalidArgVal/ter:NoConfig** SOAP 1.2 fault.

8.1.17. ONVIF Client invokes **GetAuthorizationServerConfigurations** request with parameters

- Token is skipped

8.1.18. The DUT responds with **GetAuthorizationServerConfigurationsResponse** message with parameters

- Configuration list =: *configurations4*

8.1.19. If *configurations4* contains item with @token equal to *token1*, FAIL the test, restore the DUT state, and skip other steps.

9. Restore the DUT state.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **CreateAuthorizationServerConfigurationResponse** message.
- DUT did not send **GetAuthorizationServerConfigurationsResponse** message at steps 8.1.4, 8.1.8, 8.1.16.
- DUT did not send **DeleteAuthorizationServerConfigurationResponse** message.
- DUT did not send the **env:Sender/ter:InvalidArgVal/ter:NoConfig** SOAP 1.2 fault message at step 8.1.14.

Note: The following fields are compared at steps 8.1.7 and 8.1.12:

- Type
- ClientAuth
- ServerUri
- ClientID
- Scope
- KeyID
- CertificateID
- CertPathValidationPolicyID

5.9.2 Authorization Server Configuration Modification

Test Case ID: ADVANCED_SECURITY-10-1-2

Specification Coverage: GetAuthorizationServerConfigurations (ONVIF Security Configuration Service Specification), SetAuthorizationServerConfiguration (ONVIF Security Configuration Service Specification)

Feature under test: GetAuthorizationServerConfigurations, SetAuthorizationServerConfiguration

WSDL Reference: security.wsdl

Test Purpose: Verify that Authorization Server Configuration could be modified by SetAuthorizationServerConfiguration request and received by GetAuthorizationServerConfigurations request.

Pre-Requisite: Security Configuration Service is received from the DUT. Authorization Server Configuration is supported by the DUT as indicated by the AuthorizationServer.MaxConfigurations capability. The DUT shall have enough free storage capacity for two additional key pair if private_key_jwt is supported by device as indicated by AuthorizationServer.ClientAuthenticationMethodsSupported capability. The DUT shall have enough free storage capacity for two additional certificate if self_signed_tls_client_auth is supported by device as indicated by AuthorizationServer.ClientAuthenticationMethodsSupported capability. The DUT shall have enough free storage capacity for two additional certification path validation policies.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

   - out *cap* - Security Configuration Service Capabilities

4. If *cap*.AuthorizationServer.ClientAuthenticationMethodsSupported contains "private_key_jwt":

   4.1. ONVIF Client configures key pair by following the procedure described in Annex A.3 with the following input and output parameters:

   - in *cap* - DUT capabilities

   - out *keyID1* - key pair identificator

   4.2. ONVIF Client configures key pair by following the procedure described in Annex A.3 with the following input and output parameters:

   - in *cap* - DUT capabilities

   - out *keyID2* - key pair identificator

5. If *cap*.AuthorizationServer.ClientAuthenticationMethodsSupported contains "self_signed_tls_client_auth":

   5.1. Set:

   - *keyAlgorithm* := **"ECC"** if *cap*.KeystoreCapabilities.ECCKeyPairGeneration = true; **"RSA"** otherwise.

   5.2. ONVIF Client prepares certificate by following the procedure mentioned in Annex A.38 with the following input and output parameters

   - in *cap* - service capabilities

   - in *keyAlgorithm* - key pair algorithm

   - in "CN=ONVIF TT3,C=US" - CA certificate subject

   - out *certificationPathID3* - certification path identifier (if any)

   - out *certID3* - certificate identifier

   - out *keyID3* - key pair identifier

5.3. ONVIF Client prepares certificate by following the procedure mentioned in Annex A.38 with the following input and output parameters

- in *cap* - service capabilities
- in *keyAlgorithm* - key pair algorithm
- in "CN=ONVIF TT4,C=US" - CA certificate subject
- out *certificationPathID4* - certification path identifier (if any)
- out *certID4* - certificate identifier
- out *keyID4* - key pair identifier

6. Configure certification path validation policy:

6.1. Set:

- *keyAlgorithm* := **"ECC"** if *cap*.KeystoreCapabilities.ECCKeyPairGeneration = true; **"RSA"** otherwise.

6.2. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.50 with the following input and output parameters

- in *cap* - DUT capabilities
- in *keyAlgorithm* - DUT capabilities
- in "CN=ONVIF TT5,C=US" - CA certificate subject
- in "Test CertPathValidationPolicy1 Alias" - certification path validation policy alias
- out *certPathValidationPolicyID5* - certification path validation policy identifier
- out *certID5* - certificate identifier
- out *keyID5* - RSA key pair identifier
- out *CAcert5* - CA certificate
- out *privateKey5* - CA certificate private key

6.3. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.50 with the following input and output parameters

- in *cap* - DUT capabilities

- in *keyAlgorithm* - DUT capabilities
- in "CN=ONVIF TT6,C=US" - CA certificate subject
- in "Test CertPathValidationPolicy2 Alias" - certification path validation policy alias
- out *certPathValidationPolicyID6* - certification path validation policy identifier
- out *certID6* - certificate identifier
- out *keyID6* - RSA key pair identifier
- out *CAcert6* - CA certificate
- out *privateKey6* - CA certificate private key

7. ONVIF Client deletes one authorization server configuration is maximum is reached by following the procedure described in Annex A.51 with the following input and output parameters:

   - in *cap* - DUT capabilities
   - out *itemToRestore1* - deleted authorization server configuration if any

8. ONVIF Client invokes **CreateAuthorizationServerConfiguration** request with parameters

   - Type := *cap*.AuthorizationServer.ConfigurationTypesSupported[0]
   - ClientAuth := *cap*.AuthorizationServer.ClientAuthenticationMethodsSupported[0]
   - ServerUri := *anyValidUri1*
   - ClientID := *anyClientID1*
   - ClientSecret := *anyClientSecret1* if *cap*.AuthorizationServer.ClientAuthenticationMethodsSupported[0] != "private_key_jwt", othervise is skipped
   - Scope := *anyScope1*
   - KeyID := *keyID1* if *cap*.AuthorizationServer.ClientAuthenticationMethodsSupported[0] = "private_key_jwt", othervise is skipped
   - CertificateID := *certificateID3* if *cap*.AuthorizationServer.ClientAuthenticationMethodsSupported[0] = "self_signed_tls_client_auth", othervise is skipped

- CertPathValidationPolicyID := *certPathValidationPolicyID6*

9. The DUT responds with **CreateAuthorizationServerConfigurationResponse** message with parameters

    - Token =: *token1*

10. For each configuration type supported (*configurationType*) in *cap*.AuthorizationServer.ConfigurationTypesSupported:

    10.1. For each authentication method supported (*authenticationMethod*) in *cap*.AuthorizationServer.ClientAuthenticationMethodsSupported:

    10.1.1. ONVIF Client invokes **SetAuthorizationServerConfiguration** request with parameters

    - @token := *token1*
    - Data.Type := *configurationType*
    - Data.ClientAuth := *authenticationMethod*
    - Data.ServerUri := any valid Uri other then used before in this test
    - Data.ClientID := any client id other then used before in this test
    - Data.ClientSecret := any client secret other then used before in this test if *authenticationMethod* != "private_key_jwt", otherwise is skipped
    - Data.Scope := any scope other then used before in this test
    - Data.KeyID := *keyID2* if *authenticationMethod* = "private_key_jwt", otherwise is skipped
    - Data.CertificateID := *certificateID4* if *authenticationMethod* = "self_signed_tls_client_auth", otherwise is skipped
    - Data.CertPathValidationPolicyID := *certPathValidationPolicyID5*

    10.1.2. The DUT responds with **SetAuthorizationServerConfigurationResponse** message.

    10.1.3. The DUT responds with **GetAuthorizationServerConfigurationsResponse** message with parameters

    - Configuration list =: *configurations1*

10.1.4. If *configurations1*[0].Data contains parameter ClientSecret, FAIL the test, restore the DUT state, and skip other steps.

10.1.5. If *configurations1*[0].Data does not contain the same parameters as were used at 10.1.1, FAIL the test, restore the DUT state, and skip other steps.

11. Restore the DUT state.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **CreateAuthorizationServerConfigurationResponse** message.
- DUT did not send **GetAuthorizationServerConfigurationsResponse** message.
- DUT did not send **SetAuthorizationServerConfigurationResponse** message.

Note: The following fields are compared at step 10.1.5:

- Type
- ClientAuth
- ServerUri
- ClientID
- Scope
- KeyID
- CertificateID
- CertPathValidationPolicyID

Annex A Helper Procedures and Additional Notes

A.1 Get service capabilities for Advanced Security service

Name: HelperGetServiceCapabilities_AdvancedSecurity

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification
- ONVIF Media2 Configuration Device Test Specification

Procedure Purpose: Helper procedure to get service capabilities.

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

Input: None

Returns: The service capabilities (*cap*).

Procedure:

1. ONVIF Client invokes **GetServiceCapabilities**
2. The DUT responds with **GetServiceCapabilitiesResponse** message with parameters
   - Capabilities =: *cap*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.2 Delete a key pair

Name: HelperDeleteKeyPair

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification

Procedure Purpose: Helper procedure to delete a key pair.

Pre-requisite: Security Configuration Service is received from the DUT. On-board RSA or ECC key pair generation is supported by the DUT as indicated by the RSAKeyPairGeneration or ECCKeyPairGeneration 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 –

- DUT passes all assertions.

FAIL –

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

A.3 Create a key pair

Name: HelperCreateKeyPair

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification

Procedure Purpose: Helper procedure to create ECC or RSA key pair

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

Input: The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*).

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

Procedure:

1. ONVIF Client determines the key pair generation params by following the procedure mentioned in Annex A.4 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *keyGenParams* - key pair generation params
2. If *keyGenParams*.algorithm = RSA:
   a. ONVIF Client invokes **CreateRSAKeyPair** with parameter
      - KeyLength := *keyGenParams*.keyLength

   b. The DUT responds with **CreateRSAKeyPairResponse** message with parameters
      - KeyID =: *keyID*
      - EstimatedCreationTime =: *duration*
3. If *keyGenParams*.algorithm = ECC:
   a. ONVIF Client invokes **CreateECCKeyPair** with parameter
      - EllipticCurve := *keyGenParams*.ellipticCurve

   b. The DUT responds with **CreateECCKeyPairResponse** message with parameters
      - KeyID =: *keyID*
      - EstimatedCreationTime =: *duration*
4. Until *operationDelay* + *duration* expires repeat the following steps:

   4.1. ONVIF Client waits for 5 seconds.

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 key pair (*keyID*) by following the procedure mentioned in Annex A.2.

5. If *operationDelay* + *duration* expires for step 4 and the last *keyStatus* is other than "ok", FAIL the procedure and deletes the key pair (*keyID*) by following the procedure mentioned in Annex A.2.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **CreateRSAKeyPairResponse** or **CreateECCKeyPairResponse** message.
- DUT did not send **GetKeyStatusResponse** message(s).

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

A.4 Determine key pair generation params

Name: HelperDetermineKeyPairGenerationParams

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification

Procedure Purpose: Helper procedure to determine the key pair generation params to use during testing.

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

Input: The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*).

Returns: The key pair generation params (*keyGenParams*).

Procedure:

1. If *keyAlgorithm* = RSA:
   a. ONVIF Client loops through the supported Key length list (*cap*.KeystoreCapabilities .RSAKeyLengths) and selects the smallest supported key length (*keyLength*).
   b. ONVIF Client creates the RSA key generation params (*keyGenParams*) with the key length (*keyLength*).
2. If *keyAlgorithm* = ECC:
   a. ONVIF Client loops through the supported elliptic curves list (*cap*.KeystoreCapabilities.EllipticCurves) and selects the simplest elliptic curve (*ellipticCurve*).
   b. ONVIF Client creates the ECC key generation params (*keyGenParams*) with the elliptic curve (*ellipticCurve*).
3. If previous steps is done with empty key pair generation params (*keyGenParams*): FAIL the procedure.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- No supported RSA key length was found at step 1.1 or no supported ECC elliptic curves was found at step 2.1.

A.5 Signature Algorithm Selection

Name: HelperSignatureAlgorithmSelection

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification

Procedure Purpose: Helper procedure to select signature algorithm wich will be used for tests based on Device capabilities.

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

Input: The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*).

Returns: The signature algorithm (*signatureAlgorithm*).

Procedure:

1. If *keyAlgorithm* = RSA: ONVIF Client selects signature algorithm (*signatureAlgorithm*) that will be used for the test from the list provided by DUT at *cap*.KeystoreCapabilities.SignatureAlgorithms. Selection is done among the following list of signature algorithms supported by the Client by priority from first to last:
   - 1.2.840.113549.1.1.13 (OID of SHA-512 with RSA Encryption algorithm)
   - 1.2.840.113549.1.1.12 (OID of SHA-384 with RSA Encryption algorithm)
   - 1.2.840.113549.1.1.11 (OID of SHA-256 with RSA Encryption algorithm)
   - 1.2.840.113549.1.1.14 (OID of SHA-224 with RSA Encryption algorithm)
   - 1.2.840.113549.1.1.5 (OID of SHA-1 with RSA Encryption algorithm)
2. If *keyAlgorithm* = ECC: ONVIF Client selects signature algorithm (*signatureAlgorithm*) that will be used for the test from the list provided by DUT at *cap*.KeystoreCapabilities.SignatureAlgorithms. Selection is done among the following list of signature algorithms supported by the Client by priority from first to last:
   - 1.2.840.10045.4.3.4 (OID of SHA-512 with ECC Encryption algorithm)
   - 1.2.840.10045.4.3.3 (OID of SHA-384 with ECC Encryption algorithm)
   - 1.2.840.10045.4.3.2 (OID of SHA-256 with ECC Encryption algorithm)
   - 1.2.840.10045.4.3.1 (OID of SHA-224 with ECC Encryption algorithm)
   - 1.2.840.10045.4.1 (OID of SHA-1 with ECC Encryption algorithm)

3. If the previous steps is done with empty signature algorithm (*signatureAlgorithm*): FAIL the procedure.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not return any of signature algorithms listed at step 1 or 2.

A.6 Subject for a server certificate

Name: HelperSubjectForServerCertificate

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification

Use the following subject for test cases:

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

A.7 Delete a certificate with corresponding key pair

Name: HelperDeleteCertWithKey

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification

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

Pre-requisite: Security Configuration Service is received from the DUT. Create self-signed certificate by the DUT as indicated by the SelfSignedCertificateCreation or

SelfSignedCertificateCreationWithRSA, or PCKS#10 supported by the DUT as indicated by the PKCS10 or PKCS10ExternalCertificationWithRSA capability.

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

Returns: None

Procedure:

1. ONVIF Client invokes **DeleteCertificate** with parameters
   - CertificateID := *certID*
2. The DUT responds with **DeleteCertificateResponse** message.
3. ONVIF Client deletes the key pair (in *keyID*) by following the procedure mentioned in Annex A.2.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.8 Provide CA certificate

Name: HelperCreateCACertificate

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification

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). The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*).

Returns: An X.509 CA certificate (*CAcert*) that is compliant to [RFC 5280] and a corresponding key pair (*keyPair*) with private key and public key.

Procedure:

1. ONVIF Client selects signature algorithm by following the procedure mentioned in Annex A.5 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *signatureAlgorithm* - signature algorithm
2. ONVIF Client generates a key pair by following the procedure mentioned in Annex A.9 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *keyPair* - key pair
3. If *subject* is skipped set:
   - *subject* := "CN=ONVIF TT,C=US"
4. ONVIF Client creates an X.509 self-signed CA certificate that is compliant to [RFC 5280] and has the following properties:
   - version := v3
   - signature := *signatureAlgorithm*
   - validity := not before 19700101000000Z and not after 99991231235959Z
   - subject := *subject*
   - public key := *keyPair* .publicKey
   - private key to be used := *keyPair*.privateKey

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

A.9 Generate a key pair

Name: HelperGenerateKeyPair

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification

Procedure Purpose: Helper procedure to generate a key pair.

Pre-requisite: None.

Input: The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*).

Returns: A [RFC 3447] compliant RSA or [RFC 5480, RFC 5915] compliant ECC key pair (*keyPair*) with new public key and private key.

Procedure:

1. ONVIF Client determines the key pair generation params by following the procedure mentioned in Annex A.4 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *keyGenParams* - key pair generation params
2. If *keyGenParams*.algorithm = RSA:
   a. Create an [RFC 3447] compliant RSA key pair (out *keyPair*) with new public key and private key with the following properties:
      - KeyLength := *keyGenParams*.keyLength
3. If *keyGenParams*.algorithm = ECC:
   a. Create an [RFC 5480, RFC 5915] compliant ECC key pair (out *keyPair*) with new public key and private key with the following properties:
      - EllipticCurve := *keyGenParams*.ellipticCurve

A.10 Create a CA-signed certificate for the key pair

Name: HelperCreateCASignedCertificate

Procedure Purpose: Helper procedure to create a CA-signed certificate for key pair.

Pre-requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or PKCS10ExternalCertificationWithRSA capability. On-board ECC or RSA key pair generation is supported by the DUT as indicated by the ECCKeyPairGeneration or RSAKeyPairGeneration capability. The DUT shall have enough free storage capacity for one additional key pair. Current time of the DUT shall be at least Jan 01, 1970.

Input: CA certificate (*CAcert*) and a corresponding private key (*caPrivateKey*). The service capabilities (*cap*). The CSR key pair algorithm (*csrKeyAlgorithm*).

Returns: The identifier of the new key pair (*keyID*), a CA-signed certificate (*cert*).

Procedure:

1. ONVIF Client creates a key pair by following the procedure mentioned in Annex A.3 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *csrKeyAlgorithm* - key pair algorithm
   - out *csrKeyID* - key pair ID
2. ONVIF Client selects signature algorithm by following the procedure mentioned in Annex A.5 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in csrKeyAlgorithm - key pair algorithm
   - out *caSignatureAlgorithm* - signature algorithm
3. ONVIF Client invokes **CreatePKCS10CSR** with parameters
   - Subject := *subject* (see Annex A.6)
   - KeyID := *csrKeyID*
   - CSRAttribute skipped
   - SignatureAlgorithm.algorithm := *signatureAlgorithm*
4. The DUT responds with **CreatePKCS10CSRResponse** message with parameters
   - PKCS10CSR =: *PKCS10request*
5. ONVIF Client creates a certificate from the PKCS#10 request and an associated CA certificate with related private key by following the procedure described in Annex A.11 with the following input and output parameters:
   - in *cap* - DUT capabilities

- in *PKCS10request* - PKCS#10 request
- in *CAcert* - CA certificate
- out *privateKey* - private key
- out *cert* - certificate

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.11 Creating a certificate from a PCKS#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*). The service capabilities (*cap*).

Returns: An [RFC 5280] compliant X.509 certificate (*certResult*) from the PKCS#10 request signed with the private key of the CA certificate.

Procedure:

1. ONVIF Client selects signature algorithm by following the procedure mentioned in Annex A.5 with the following input and output parameters:
    - in *cap* - DUT capabilities
    - in *privateKey*.algorithm - key pair algorithm
    - out *signatureAlgorithm* - signature algorithm
2. Create an [RFC 5280] compliant X.509 certificate (*certResult*) from the PKCS#10 request (*pkcs10*) with the following properties:

- version:= v3
- signature := *signatureAlgorithm*
- 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*) of the CA certificate (*CAcert*)
- certificate extensions := the X.509v3 extensions from the PKCS#10 request (*pkcs10*)

A.12 Create a self-signed certificate

Name: HelperCreateSelfSignedCertificate

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification

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

Pre-requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or SelfSignedCertificateCreationWithRSA capability. On-board ECC or RSA key pair generation is supported by the DUT as indicated by the ECCKeyPairGeneration or RSAKeyPairGeneration 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.

Input: The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*).

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

Procedure:

1. ONVIF Client creates a key pair by following the procedure mentioned in Annex A.3 with the following input and output parameters:

- in *cap* - DUT capabilities
- in *keyAlgorithm* - key pair algorithm
- out *keyID* - key pair ID

2. ONVIF Client selects signature algorithm by following the procedure mentioned in Annex A.5 with the following input and output parameters:
    - in *cap* - DUT capabilities
    - in *keyAlgorithm* - Key Pair Algorithm
    - out *signatureAlgorithm* - signature algorithm
3. ONVIF Client invokes **CreateSelfSignedCertificate** with parameters
    - X509Version skipped
    - KeyID := *keyID*
    - Subject := subject (see Annex A.6)
    - Alias skipped
    - notValidBefore skipped
    - notValidAfter skipped
    - SignatureAlgorithm := *signatureAlgorithm*
    - SignatureAlgorithm.parameters skipped
    - SignatureAlgorithm.anyParameters skipped
    - Extension skipped
4. The DUT responds with **CreateSelfSignedCertificateResponse** message with parameters
    - CertificateID =: *certID*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.13 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 –

- DUT passes all assertions.

A.14 Upload a certificate without Private Key Assignment

Name: HelperUploadCertificate

Procedure Purpose: Helper procedure to upload a certificate without private key assignment.

Pre-requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or 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.

Input: Certificate (*cert*).

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

Procedure:

1. ONVIF Client invokes **UploadCertificate** with parameters

- Certificate := *cert*
- Alias := "ONVIF_Test"
- PrivateKeyRequired : = false

2. DUT responds with a **UploadCertificateResponse** message.

   - Certificate := *certID*
   - KeyID := *keyID*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.15 Delete a certification path with corresponding certificate and key pair

Name: HelperDeleteCertificationPath

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

Pre-requisite: Security Configuration Service is received from the DUT. Create self-signed certificate by the DUT as indicated by the SelfSignedCertificateCreation or SelfSignedCertificateCreationWithRSA, or PCKS#10 supported by the DUT as indicated by the PKCS10 or PKCS10ExternalCertificationWithRSA capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability.

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

Returns: None

Procedure:

1. ONVIF Client invokes **DeleteCertificationPath** request with parameters

   - CertificationPathID := *certPathID*

2. The DUT responds with a **DeleteCertificationPathResponse** message.

3. ONVIF Client deletes the self-signed certificate (in *certID*) and the related key pair (in *keyID*) by following the procedure mentioned in Annex A.7.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **DeleteCertificationPathResponse** 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: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or PKCS10ExternalCertificationWithRSA capability. On-board ECC or RSA key pair generation is supported by the DUT as indicated by the ECCKeyPairGeneration or RSAKeyPairGeneration 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.

Input: CA certificate (*CAcert*) and a corresponding private key (*caPrivateKey*). The service capabilities (*cap*). The key pair algorithm of the key which will be in the result certificate (*certKeyAlgorithm*).

Returns: The identifier of the new key pair (*keyID*), a certificate identifier (*certID*), a certificate (*cert*).

Procedure:

1. ONVIF Client creates a certificate from the PKCS#10 request with key pair and associated CA certificate and a corresponding private key by following the procedure described in Annex A.10 with the following input and output parameters:

   - in *cap* - DUT capabilities

   - in *CAcert* - CA certificate

   - in *caPrivateKey* - private key

   - in *certKeyAlgorithm* - key pair algorithm

- out *cert* - certificate
- out *keyID1* - key pair ID

2. ONVIF Client invokes **UploadCertificate** with parameters
   - Certificate := *cert*
   - Alias := "ONVIF_Test1"
   - PrivateKeyRequired := true
3. The DUT responds with **UploadCertificateResponse** with parameters
   - CertificateID =: *certID*
   - KeyID =: *keyID*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.17 Delete a certification path with corresponding two certificates and key pairs

Name: HelperDeleteCertificationPath2

Procedure Purpose: Helper procedure to delete certification path and related certificates and key pairs.

Pre-requisite: Security Configuration Service is received from the DUT. Create self-signed certificate by the DUT as indicated by the SelfSignedCertificateCreation or SelfSignedCertificateCreationWithRSA, or PCKS#10 supported by the DUT as indicated by the PKCS10 or PKCS10ExternalCertificationWithRSA capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability.

Input: The identifier of the certification path (*certPathID*), certificate (*certID1*) and key pair (*keyID1*), certificate (*certID2*) and key pair (*keyID2*) to delete.

Returns: None

Procedure:

1. ONVIF Client invokes **DeleteCertificationPath** request with parameters
   - CertificationPathID := *certPathID*
2. The DUT responds with **DeleteCertificationPathResponse** message.
3. ONVIF Client deletes the CA certificate (*certID2*) and related key pair (*keyID2*) by following the procedure mentioned in Annex A.7.
4. ONVIF Client deletes the CA certificate (*certID1*) and related key pair (*keyID1*) by following the procedure mentioned in Annex A.7.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.18 Create a certification path based on self-signed certificate

Name: HelperCreateCertificationPath_SelfSigned

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification

Procedure Purpose: Helper procedure to create a certification path based on self-signed certificate.

Pre-requisite: Security Configuration 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: The service capabilities (*cap*).

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 and related RSA key pair by following the procedure mentioned in Annex A.12 with the following input and output parameters:
    - in *cap* - service capabilities
    - out *keyID* - key pair ID
    - out *certID* - certificate ID
2. ONVIF Client invokes **CreateCertificationPath** request with parameters
    - CertficateIDs.CertificateID[0] := *certID*
    - Alias := "ONVIF_Test"
3. The DUT responds with **CreateCertificationPathResponse** message with parameters
    - CertificationPathID =: *certPathID*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.19 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: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or PKCS10ExternalCertificationWithRSA capability. On-board ECC or RSA key pair generation is supported by the DUT as indicated by the ECCKeyPairGeneration or 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 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: The service capabilities (*cap*). The CA key pair algorithm (*caKeyAlgorithm*). The key pair algorithm of the key which will be in the result certificate (*certKeyAlgorithm*).

Returns: The identifier of the new certification path (*certPathID*) and two related certificates: CA-signed certificate (*certID*) and related key (*keyID*) and associated CA certificate (*CACertID*) and related key (*CAKeyID*).

Procedure:

1. ONVIF Client creates a CA certificate and a corresponding private key by following the procedure described in Annex A.8 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - out *CAcert* - CA certificate
   - out *caKeyPair* - key pair
2. ONVIF Client creates and uploads a CA-signed certificate for the key pair and associated CA certificate and a corresponding by following the procedure described in Annex A.16 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *CAcert* - CA certificate
   - in *caKeyPair*.privateKey - CA certificate private key
   - in *certKeyAlgorithm* - key pair algorithm
   - out *certID* - CA-signed certificate
   - out *keyID* - RSA key pair
3. ONVIF Client uploads a CA certificate (out *CACertID*, in *CAcert*) and new key pair with the public key from the CA certificate (out *CAKeyID*) by following the procedure described in Annex A.14.
4. ONVIF Client invokes **CreateCertificationPath** with parameters
   - CertficateIDs.CertificateID[0] =: *certID*

- CertficateIDs.CertificateID[1] =: *CACertID*
- Alias := "ONVIF_Test2"

5. The DUT responds with **CreateCertificationPathResponse** message with parameters
   - CertificationPathID =: *certPathID*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.20 Provide expired CA certificate

Name: HelperCreateExpiredCACertificate

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

Pre-requisite: None.

Input: The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*).

Returns: An X.509 CA certificate (*CAcert*) that is compliant to [RFC 5280] and a corresponding key pair with public and private keys (*keyPair*).

Procedure:

1. ONVIF Client selects signature algorithm by following the procedure mentioned in Annex A.5 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *signatureAlgorithm* - signature algorithm
2. ONVIF Client generates a key pair by following the procedure mentioned in Annex A.9 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm

- out *keyPair* - key pair

3. ONVIF Client invokes **GetSystemDateAndTimeRequest**.
4. The DUT responds with **GetSystemDateAndTimeResponse** with parameters
   - UTCDateTime =: *DUTCurrentTime*
5. ONVIF Client creates an X.509 self-signed CA certificate that is compliant to [RFC 5280] and has the following properties:
   - version := v3
   - signature := *signatureAlgorithm*
   - validity := not before 19700101000000Z and not after [*DUTCurrentTime* - 1 day]
   - public key := *keyPair*.publicKey
   - private key to be used := *keyPair*.privateKey

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

A.21 Remove Server Certificate Assignment

Name: HelperPreparationRemoveServerCertificateAssignment

Procedure Purpose: Helper to disable HTTPS and remove Server Certificate Assignment if required.

Pre-requisite: Security Configuration Service is received from the DUT. TLS is supported by the DUT as indicated by the TLSServerSupported capability.

Input: None.

Returns: Initial HTTPS State (*initialHTTPSState*). Removed Server Certificate Assignment (*certPathID*).

Procedure:

1. ONVIF Client invokes **GetNetworkProtocols** to retrieve configured network protocols of the DUT.
2. The DUT responds with a **GetNetworkProtocolsResponse** message with parameters
   - NetworkProtocols list =: *networkProtocolsList*

3. If *networkProtocolsList* does not contain network protocol with NetworkProtocols.Name is equal to HTTPS, FAIL the test and skip other steps.

4. If HTTPS protocol with NetworkProtocols.Name is equal to HTTPS from *networkProtocolsList* has NetworkProtocols.Enabled equal to true:

   4.1. Set *initialHTTPSState* := NetworkProtocols value for HTTPS from *networkProtocolsList*.

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

      - NetworkProtocols[0].Name := HTTPS
      - NetworkProtocols[0].Enabled := false
      - NetworkProtocols[0].Port := *initialHTTPSState*.Port
      - NetworkProtocols[0].Extension skipped

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

   4.4. ONVIF Client waits for time *operationDelay*.

   4.5. Set *HTTPSStateChangedFlag* := true.

5. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters

   - out *cap* - Security Configuration Service Capabilities

6. ONVIF Client invokes **GetAssignedServerCertificates**.

7. The DUT responds with a **GetAssignedServerCertificatesResponse** message with parameters

   - CertificationPathID list =: *initialCertificationPathList*

8. If number of items in *initialCertificationPathList* = *cap*.TLSServerCapabilities.MaximumNumberOfTLSCertificationPaths:

   8.1. Set *certPathID* := *initialCertificationPathList*[0].

   8.2. ONVIF Client invokes **RemoveServerCertificateAssignment** .

      - CertificationPathID =: *certPathID*

   8.3. The DUT responds with a **RemoveServerCertificateAssignmentResponse** message.

8.4. ONVIF Client waits for time *operationDelay*.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetNetworkProtocolsResponse** message.
- DUT did not send **SetNetworkProtocolsResponse** message.
- DUT did not send **GetAssignedServerCertificatesResponse** message.
- DUT did not send **RemoveServerCertificateAssignmentResponse** message.

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

A.22 Remove server certificate assignment with corresponding certification path, certificate and key pair

Name: HelperRemoveServerCertificateAssignment

Procedure Purpose: Helper procedure to remove server certificate assignment with corresponding certification path, certificate and key pair.

Pre-requisite: Security Configuration Service is received from the DUT. Create self-signed certificate by the DUT as indicated by the SelfSignedCertificateCreation or SelfSignedCertificateCreationWithRSA, or PCKS#10 supported by the DUT as indicated by the PKCS10 or PKCS10ExternalCertificationWithRSA capability. TLS supported by the DUT as indicated by the TLSServerSupported capability.

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

Returns: None

Procedure:

1. ONVIF Client invokes **RemoveServerCertificateAssignment** with parameters
   - CertificationPathID := *certPathID*
2. The DUT responds with **RemoveServerCertificateAssignmentResponse** message.

3. ONVIF Client waits for time *operationDelay*.

4. ONVIF Client deletes the certification path (in *certPathID*) and related certificate (in *certID*) and the key pair (in *keyID*) by following the procedure mentioned in Annex A.15.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.23 Restore Server Certificate Assignment

Name: HelperRestoreServerCertificateAssignment

Procedure Purpose: Helper to restore HTTPS and Server Certificate Assignment if required.

Pre-requisite: Security Configuration Service is received from the DUT. TLS is supported by the DUT as indicated by the TLSServerSupported capability.

Input: Initial HTTPS State (*initialHTTPSState*). Removed Server Certificate Assignment (*certPathID*).

Returns: None.

Procedure:

1. If *certPathID* specified:

   1.1. ONVIF Client invokes **AddServerCertificateAssignment** with parameters

   - CertificationPathID =: *certPathID*

   1.2. The DUT responds with an **AddServerCertificateAssignmentResponse** message.

   1.3. ONVIF Client waits for time *operationDelay*.

2. If *initialHTTPSState* specified:

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

   - NetworkProtocols[0] := *initialHTTPSState*

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

2.3. ONVIF Client waits for time *operationDelay*.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **AddServerCertificateAssignmentResponse** message.
- DUT did not send **SetNetworkProtocolsResponse** message.

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

A.24 Remove server certificate assignment with corresponding certification path, certificates and key pairs

Name: HelperRemoveServerCertificateAssignment2

Procedure Purpose: Helper procedure to remove server certificate assignment with corresponding certification path, certificates and key pairs.

Pre-requisite: Security Configuration Service is received from the DUT. Create self-signed certificate by the DUT as indicated by the SelfSignedCertificateCreation or SelfSignedCertificateCreationWithRSA, or PCKS#10 supported by the DUT as indicated by the PKCS10 or PKCS10ExternalCertificationWithRSA capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability.

Input: The identifier of certification path (*certPathID*), certificate (*certID1*) and key pair (*keyID1*), certificate (*certID2*) and key pair (*keyID2*) to delete.

Returns: None

Procedure:

1. ONVIF Client invokes **RemoveServerCertificateAssignment** with parameters
   - CertificationPathID =: *certPathID*
2. The DUT responds with **RemoveServerCertificateAssignmentResponse** message.
3. ONVIF Client waits for time *operationDelay*.

4. ONVIF Client deletes the certification path (in *certPathID*), related the CA certificate (in *certID2*) and the key pair (in *keyID2*) and related the CA-signed certificate (in *certID1*) and the key pair (in *keyID1*) by following the procedure mentioned in Annex A.17.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.25 Add server certificate assignment with corresponding certification path, self-signed certificate and key pair

Name: HelperAddServerCertAssign_SSCertificate

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification

Procedure Purpose: Helper procedure to add server certificate assignment with corresponding certification path, self-signed certificate and key pair.

Pre-requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or SelfSignedCertificateCreationWithRSA capability. On-board ECC or RSA key pair generation is supported by the DUT as indicated by the ECCKeyPairGeneration or 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 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: The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*).

Returns: The identifiers of the new certification path (*certPathID*), certificate (*certID*) and key pair (*keyID*).

Procedure:

1. ONVIF Client creates a certification path based on self-signed certificate and related key pair by following the procedure mentioned in Annex A.18 with the following input and output parameters:
   - in *cap* - service capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *keyID* - key pair ID
   - out *certID1* - certificate ID
   - out *certPathID* - certification path ID
2. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
   - CertificationPathID := *certPathID*
3. The DUT responds with **AddServerCertificateAssignmentResponse** message.
4. ONVIF Client waits for time *operationDelay*.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.26 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*).

Returns: None

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 –

- DUT passes all assertions.

FAIL –

- DUT did not send **ServerHello** message.
- DUT did not send **Certificate** message.
- DUT did not send **ServerHelloDone** message.
- DUT did not send **ChangeCipherSpec** message.
- DUT did not send **Finished** message.
- The DUT TLS server sends Alert Message.

A.27 Passphrases for test cases

Use the following passphrases for test cases (20 ASCII characters):

- *passphrase1* := "Passphrase for ONVIF"
- *passphrase2* := "AdditionalPassphrase"

A.28 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. The subject (*subject*) of CA certificate (optional input parameter, could be skipped). The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*).

Returns: A [PKCS#12] compliant PKCS#12 data structure (*PKCS12data*) with CA certificate (*CAcert*) and a corresponding key pair (*keyPair*) with a corresponding public key in the certificate along with the corresponding private key encrypted with passphrase (*passphrase*).

Procedure:

1. ONVIF Client creates an CA certificate by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - in *subject* - CA certificate subject
   - out *keyPair* - key pair for the CA certificate
2. ONVIF Client creates a PKCS#12 file by following the procedure mentioned in Annex A.29 with the following input and output parameters
   - in *CAcert* - CA certificate
   - in *keyPair* - key pair
   - in *passphrase* - passphrase
   - out *PKCS12data* - PKCS#12 file

A.29 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 [RFC 5280] and a corresponding key pair (*keyPair*) with private key and public key, and passphrase (*passphrase*).

Returns: A [PKCS#12] compliant PKCS#12 data structure (*PKCS12data*).

Procedure:

1. Use the current PrivateKeyInfo data:
   - PrivateKeyInfo
     - version := v2
     - privateKeyAlgorithm := *keyPair*.algorithm
     - privateKey := *keyPair* .privateKey
     - publicKey := *keyPair* .publicKey
2. Create an EncryptedPrivateKeyInf 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.30 Delete a certification path with corresponding certificate and key pair when CertificateID is unknown

Name: HelperDeleteCertificationPath3

Procedure Purpose: Helper procedure to delete certification path and related certificate and key pair when CertificateID is unknown.

Pre-requisite: Security Configuration Service is received from the DUT. Create self-signed certificate by the DUT as indicated by the SelfSignedCertificateCreation or SelfSignedCertificateCreationWithRSA, or PCKS#10 supported by the DUT as indicated by the PKCS10 PKCS10ExternalCertificationWithRSA capability. TLS is supported by the DUT as indicated by the TLSServerSupported capability.

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

Returns: None

Procedure:

1. ONVIF Client invokes **GetCertificationPath** request with parameters
   - CertificationPathID =: *certPathID*
2. The DUT responds with **GetCertificationPathResponse** 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 key pair (in *X509Cert*.KeyID) by following the procedure mentioned in Annex A.2 to restore DUT configuration.

8. Skip other steps.

9. ONVIF Client deletes the key pair (in *keyID*) by following the procedure mentioned in Annex A.2 to restore DUT configuration.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetCertificationPathResponse** message.
- DUT did not send **DeleteCertificationPathResponse** message.
- DUT did not send **DeleteCertificateResponse** message.
- DUT did not send **GetCertificateResponse** message.

A.31 Provide certificate signed by private key of other certificate

Name: HelperCreateSignedCertificate

Notes: Annex is used at:

- ONVIF Security Configuration Device Test Specification
- ONVIF Uplink Test Specification

- ONVIF Real Time Streaming using Media2 Device Test Specification

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*). The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*). Certificate SAN (*certSAN*, optional).

Returns: An X.509 certificate (*cert*) signed by input private key that is compliant to [RFC 5280] and a corresponding key pair (*keyPair*) with the corresponding private key and public key.

Procedure:

1. ONVIF Client generates a key pair by following the procedure mentioned in Annex A.9 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *keyPair* - key pair
2. ONVIF Client selects signature algorithm by following the procedure mentioned in Annex A.5 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *inputPrivateKey*.algorithm - key pair algorithm
   - out *signatureAlgorithm* - signature algorithm
3. ONVIF Client creates an X.509 certificate signed by *inputPrivateKey* that is compliant to [RFC 5280] and has the following properties:
   - version:= v3
   - signature := *signatureAlgorithm*
   - publicKey := *keyPair*.publicKey
   - validity := validity from *cert*
   - subject := *subject*
   - SAN := *certSAN*

- 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.32 Create a certification path validation policy with provided certificate identifier

Name: HelperCreateCertPathValidationPolicyWithCertID

Notes: Annex is used at:

- ONVIF Security Configuration Device Test Specification
- ONVIF Uplink Test Specification
- ONVIF Real Time Streaming using Media2 Device Test Specification

Procedure Purpose: Helper procedure to create a certification path validation policy with provided certificate identifier.

Pre-requisite: Security Configuration 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 –

- DUT passes all assertions.

FAIL –

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

A.33 Delete a certification path validation policy

Name: HelperDeleteCertPathValidationPolicy

Procedure Purpose: Helper procedure to delete a certification path validation policy.

Pre-requisite: Security Configuration 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** request with parameters

   - CertPathValidationPolicyID =: *certPathValidationPolicyID*

2. The DUT responds with **DeleteCertPathValidationPolicyResponse** message.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.34 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*). The service capabilities (*cap*).

Returns: A CRL (*crl*) that is compliant to [RFC 5280].

Procedure:

1. ONVIF Client selects signature algorithm by following the procedure mentioned in Annex A.5 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *privateKey*.algorithm - key pair algorithm
   - out *signatureAlgorithm* - signature algorithm
2. ONVIF Client creates a CRL that is compliant to [RFC 5280] signed signed by private key *privateKey* and has the following properties:
   - tbsCertList[0].version := v2
   - tbsCertList[0].signature.algorithm := *signatureAlgorithm*
   - tbsCertList[0].issuer := "ONVIF_DTT"
   - tbsCertList[0].thisUpdate := [current time] - 1 day
   - tbsCertList[0].nextUpdate skipped
   - tbsCertList[0].revokedCertificates[0].userCertificate := *cert*
   - tbsCertList[0].revokedCertificates[0].revocationDate := [current time] - 1 day - 1 hour
   - signatureAlgorithm.algorithm := *signatureAlgorithm*
   - signatureValue := *signatureAlgorithm* signature

Note: ONVIF Client may return the same CRL in subsequent invocations of this procedure.

A.35 Upload a CRL

Name: HelperUploadCRL

Procedure Purpose: Helper procedure to upload a CRL.

Pre-requisite: Security Configuration 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 (*crl*).

Returns: The CRL identifier (*crlID*).

Procedure:

1. ONVIF Client invokes **UploadCRL** with parameters
    - Crl =: *crl*
    - Alias := "Test CRL Alias"
    - anyParameters skipped
2. The DUT responds with **UploadCRLResponse** message with parameters
    - CrlID =: *crlID*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.36 Delete a CRL

Name: HelperDeleteCRL

Procedure Purpose: Helper procedure to delete a CRL.

Pre-requisite: Security Configuration Service is received from the DUT. CRLs supported by the DUT as indicated by the MaximumNumberOfCRLs capability.

Input: The identifier of CRL (*crlID*) to delete.

Returns: None

Procedure:

1. ONVIF Client invokes **DeleteCRL** request with parameters
   - CrlID =: *crlID*
2. The DUT responds with **DeleteCRLResponse** message.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.37 Create a certification path validation policy

Name: HelperCreateCertPathValidationPolicy

Procedure Purpose: Helper procedure to create a certification path validation policy.

Pre-requisite: Security Configuration 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 service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*). The certification path validation policy alias (*alias*). The subject (*subject*) of CA certificate (optional input parameter, could be skipped).

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 by following the procedure mentioned in Annex A.38 with the following input and output parameters

- in *cap* - service capabilities
- in *keyAlgorithm* - key pair algorithm
- in *subject* - CA certificate subject
- out *certificationPathID1* - certification path identifier (if any)
- out *certID1* - certificate identifier
- out *keyID1* - key pair identifier

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.32.

Procedure Result:

PASS –

- DUT passes all assertions.

A.38 Prepare certificate on the DUT

Name: HelperPrepareCertificate

Procedure Purpose: Helper procedure to create or upload certificate on the DUT.

Pre-requisite: Security Configuration Service is received from the DUT. Create self-signed certificate supported by the DUT as indicated by the SelfSignedCertificateCreation or SelfSignedCertificateCreationWithRSA capability and On-board ECC or RSA key pair generation is supported by the DUT as indicated by the ECCKeyPairGeneration or RSAKeyPairGeneration capability or create PKCS#10 supported by the DUT as indicated by the PKCS10 or PKCS10ExternalCertificationWithRSA capability, or certificate along with an ECC or RSA private key in a PKCS#12 data structure upload is supported by the DUT as indicated by the PKCS12 or 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 key pair. The DUT shall have enough free storage capacity for one additional certificate.

Input: The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*). The subject (*subject*) of CA certificate (optional input parameter, could be skipped).

Returns: The identifier of the new certificate (*certID*), key pair (*keyID*) and certification path if any (*certificationPathID*).

Procedure:

1. If (*keyAlgorithm* = RSA and *cap*.KeystoreCapabilities.PKCS10ExternalCertificationWithRSA) or *cap*.KeystoreCapabilities.PKCS10:

   1.1. ONVIF Client creates an CA certificate by following the procedure mentioned in Annex A.8 with the following input and output parameters

   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - in *subject* - CA certificate subject
   - out *publicKey* - public key for the CA certificate

   1.2. ONVIF Client uploads a CA certificate (out *certID*, in *CAcert*) and new key pair with the public key from the CA certificate (out *keyID*) by following the procedure described in Annex A.14.

   1.3. Set:

   - *certificationPathID* := null

   1.4. Skip other steps.

2. If (*keyAlgorithm* = RSA and *cap*.KeystoreCapabilities.SelfSignedCertificateCreationWithRSA and *cap*.KeystoreCapabilities.RSAKeyPairGeneration) or (*cap*.KeystoreCapabilities.SelfSignedCertificateCreation and *cap*.KeystoreCapabilities.*keyAlgorithm*KeyPairGeneration) :

   2.1. ONVIF Client creates a self-signed certificate and related key pair by following the procedure mentioned in Annex A.12 with the following input and output parameters:

   - in *cap* - service capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *keyID* - key pair ID
   - out *certID* - certificate ID

   2.2. Set:

   - *certificationPathID* := null

2.3. Skip other steps.

3. If (*keyAlgorithm* = RSA and *cap*.KeystoreCapabilities.PKCS12CertificateWithRSAPrivateKeyUpload) or *cap*.KeystoreCapabilities.PKCS12:

   3.1. ONVIF Client creates an CA certificate in the form of a PKCS#12 file and uploads it by following the procedure mentioned in Annex A.39 with the following input and output parameters

   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - in *subject* - CA certificate subject
   - out *certificationPathID* - certification path identifier
   - out *keyID* - key pair identifier
   - out *PKCS12data* - PKCS#12 file
   - out *CAcert* - CA certificate
   - out *publicKey* - public key for the CA certificate
   - out *privateKey* - private key for the CA certificate

   3.2. ONVIF Client invokes **GetCertificationPath** message with parameters

   - CertificationPathID =: *certificationPathID*

   3.3. The DUT responds with a **GetCertificationPathResponse** message with parameters

   - CertificationPath.CertificateID[0] =: *certID*
   - CertificationPath.Alias

   3.4. Skip other steps.

4. FAIL the test and skip other steps.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.39 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: Security Configuration Service is received from the DUT. Certificate along with an ECC or RSA private key in a PKCS#12 data structure upload is supported by the DUT as indicated by the PKCS12 or PKCS12CertificateWithRSAPrivateKeyUpload 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. The DUT shall have enough free storage capacity for one additional certification path.

Input: The subject (*subject*) of CA certificate (optional input parameter, could be skipped). The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*).

Returns: A [PKCS#12] compliant PKCS#12 data structure (*PKCS12data*) with CA certificate (*CAcert*) and a corresponding key pair (*keyPair*) with a corresponding public key in the certificate along with the corresponding private key and certification path ID (*certificationPathID*) with corresponding key pair ID (*keyID*) for uploaded PKCS#12 data structure.

Procedure:

1. ONVIF Client generates an encryption passphrase *passphrase1* (see Annex A.27).

2. ONVIF Client creates an CA certificate in a form of PKCS#12 file with with given passphrase by following the procedure mentioned in Annex A.28 with the following input and output parameters

   - in *cap* - DUT capabilities

   - in *keyAlgorithm* - key pair algorithm

   - in *subject* - CA certificate subject

   - in *passphrase1* - passphrase

   - out *PKCS12data* - PKCS#12 file

   - out *CAcert* - CA certificate

- out *keyPair* - key pair for the CA certificate

3. ONVIF Client invokes **UploadCertificateWithPrivateKeyInPKCS12** with parameters
    - CertWithPrivateKey := *PKCS12data*
    - CertificationPathAlias := "ONVIF_Certification_Path_Test"
    - KeyAlias := "ONVIF_Key_Test"
    - IgnoreAdditionalCertificates skipped
    - IntegrityPassphraseID skipped
    - EncryptionPassphraseID skipped
    - Passphrase := *passphrase1*
4. The DUT responds with a **UploadCertificateWithPrivateKeyInPKCS12Response** message with parameters
    - CertificationPathID =: *certificationPathID*
    - KeyID =: *keyID*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.40 Delete a passphrase

Name: HelperDeletePassphrase

Procedure Purpose: Helper procedure to delete a passphrase.

Pre-requisite: Security Configuration 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 –

- DUT passes all assertions.

FAIL –

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

A.41 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: The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*).

Returns: A [RFC 5958, RFC 5959] compliant PKCS#8 data structure (*keyPairInPKCS8*) with new key pair (*keyPair*) with public key and private key.

Procedure:

1. ONVIF Client generates key pair with public key and private key by following the procedure mentioned in Annex A.9 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *keyPair* - key pair
2. ONVIF Client generates a PKCS#8 data structure (out *keyPairInPKCS8*) with existing key pair (in *keyPair*) by following the procedure mentioned in Annex A.42.

A.42 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 or [RFC 5480, RFC 5915] compliant ECC key pair (*keyPair*) with public key and private key.

Returns: A [RFC 5958, RFC 5959] compliant PKCS#8 data structure (*keyPairInPKCS8*) for provided key pair.

Procedure:

1. Create an [RFC 5958, RFC 5959] compliant PKCS#8 data structure (*keyPairInPKCS8*) with the following properties:

   - PrivateKeyInfo
     - version:= v2
     - privateKeyAlgorithm := *keyPair*.algorithm
     - privateKey := *keyPair*.privateKey
     - attributes
       - publicKey := *keyPair*.publicKey

A.43 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. The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*).

Returns: A [RFC 5958, RFC 5959] compliant PKCS#8 data structure (*keyPairInPKCS8*) with new key pair (*keyPair*) with public key and private key.

Procedure:

1. ONVIF Client generates a key pair by following the procedure mentioned in Annex A.9 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *keyPair* - key pair
2. ONVIF Client generates a PKCS#8 data structure (out *keyPairInPKCS8*) with existing key pair (in *keyPair*) with encryption passphrase (in *passphrase*) by following the procedure mentioned in Annex A.44.

A.44 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 or [RFC 5480, RFC 5915] compliant ECC key pair (*keyPair*) with public key and private key. The passphrase (*passphrase*) to use in encryption.

Returns: A [RFC 5958, RFC 5959] compliant PKCS#8 data structure (*keyPairInPKCS8*) for provided key pair.

Procedure:

1. Use the current PrivateKeyInfo data:
   - PrivateKeyInfo
     - version := v2
     - privateKeyAlgorithm := *keyPair*.algorithm
     - privateKey := *keyPair*.privateKey
     - attributes

        - publicKey := *keyPair* .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.45 Provide CRL

Name: HelperCreateCRL

Procedure Purpose: Helper procedure to create CRL.

Pre-requisite: None.

Input: The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*).

Returns: A CRL (*crl*) that is compliant to [RFC 5280].

Procedure:

1. ONVIF Client selects signature algorithm by following the procedure mentioned in Annex A.5 with the following input and output parameters:

    - in *cap* - DUT capabilities

    - in *keyAlgorithm* - key pair algorithm

    - out *signatureAlgorithm* - signature algorithm

2. ONVIF Client creates a CRL that is compliant to [RFC 5280] and has the following properties:

    - tbsCertList[0].version := v2

    - tbsCertList[0].signature.algorithm := *signatureAlgorithm*

    - 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 := *signatureAlgorithm*
- signatureValue := *signatureAlgorithm* signature

Note: ONVIF Client may return the same CRL in subsequent invocations of this procedure.

A.46 Configuring HTTPS if Required

Name: HelperCheckAndConfigureHTTPS

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification

Procedure Purpose: Helper Procedure to check and configure HTTPS using Security Configuration Service if required.

Pre-requisite: RTP/RTSP/HTTPS feature is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Security Configuration Service is received from the DUT, if TLS Server is supported by DUT.

Input: The service capabilities (*cap*) (optional input parameter, could be skipped).

Returns: None.

Procedure:

1. ONVIF Client invokes **GetNetworkProtocols** request.
2. The DUT responds with **GetNetworkProtocolsResponse** with parameters
   - NetworkProtocols list =: *networkProtocolsList*
3. If *networkProtocolsList* contains item with Name = HTTPS and Enabled = true, return to the test and skip other procedure steps.
4. If the DUT does not support TLS Server, FAIL the test and skip other steps.
5. If *cap* is empty, ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.1 with the following input and output parameters:

- out *cap* - Security Configuration Service Capabilities

6. ONVIF Client configures HTTPS by following the procedure mentioned in Annex A.47 with the following input parameters:

   - in *cap* - DUT capabilities

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.47 Configuring HTTPS using Security Configuration Service

Name: HelperConfigureHTTPS

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification

Procedure Purpose: Helper Procedure to configure HTTPS using Security Configuration Service.

Pre-requisite: Security Configuration Service is received from the DUT. TLS Server is supported by the DUT. 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. 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.

Input: The service capabilities (*cap*). The key pair algorithm (*certKeyPairAlgorithm*) for Certificate. The key pair algorithm (*caKeyPairAlgorithm*) for CA.

Returns: None

Procedure:

1. ONVIF Client invokes **GetAssignedServerCertificates**.

2. The DUT responds with a **GetAssignedServerCertificatesResponse** message with parameters

   - CertificationPathID list =: *initialCertificationPathList*

3. If number of items in *initialCertificationPathList* >= 1, go to the step 6.

4. If Create self-signed certificate is supported by the DUT:

   4.1. ONVIF Client adds server certification assignment and creates related certification path by following the procedure mentioned in Annex A.25 with the following input and output parameters:

   - in *cap* - DUT capabilities
   - in *certKeyAlgorithm* - key pair algorithm
   - out *certPathID* - certification path
   - out *certID* - self-signed certificate
   - out *keyID* - RSA key pair

   4.2. Go to the step 6.

5. ONVIF Client creates a certification path based on CA-signed certificate and related key pair and a corresponding CA certificate and related key pair by following the procedure mentioned in Annex A.48 with the following input and output parameters:

   - in *cap* - DUT capabilities
   - in *certKeyPairAlgorithm* - key pair algorithm for Certificate
   - in *caKeyPairAlgorithm* - key pair algorithm for CA
   - out *certPathID* - certification path
   - out *certID* - certificate
   - out *keyID* - key pair

6. ONVIF Client invokes **SetNetworkProtocols** request with parameters

   - NetworkProtocols[0].Name := HTTPS
   - NetworkProtocols[0].Enabled := true
   - NetworkProtocols[0].Port := 443

- NetworkProtocols[0].Extension skipped

7. The DUT responds with **SetNetworkProtocolsResponse** message.
8. ONVIF Client waits until *operationDelay* timeout expires.
9. ONVIF Client checks that HTTPS protocol Port 443 is open. If HTTPS protocol port 443 is not open, FAIL the test and skip other steps.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

A.48 Add server certificate assignment with corresponding certification path, CA certificate and key pair

Name: HelperAddServerCertAssign_CACertificate

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification

Procedure Purpose: Helper Procedure to configure HTTPS using Security Configuration Service.

Pre-requisite: Security Configuration Service is received from the DUT. TLS Server is supported by the DUT. Create PCKS#10 supported by the DUT. Key pair generation is supported by the DUT. 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. 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.

Input: The service capabilities (*cap*). The key pair algorithm (*certKeyPairAlgorithm*) for Certificate. The key pair algorithm (*caKeyPairAlgorithm*) for CA.

Returns: The identifiers of the new certification path (*certPathID*), certificate (*certID*) and key pair (*keyID*).

Procedure:

1. ONVIF Client creates a key pair by following the procedure mentioned in Annex A.3 with the following input and output parameters
   - in *cap* - DUT capabilities
   - in *certKeyAlgorithm* - key pair algorithm
   - out *keyID* - key pair ID
2. ONVIF Client selects signature algorithm by following the procedure mentioned in Annex A.5 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *caKeyAlgorithm* - key pair algorithm
   - out *signatureAlgorithm* - signature algorithm
3. ONVIF Client invokes **CreatePKCS10CSR** with parameter
   - Subject := subject (see Annex A.6)
   - KeyID := *keyID*
   - CSRAttribute skipped
   - SignatureAlgorithm.algorithm := *signatureAlgorithm*
4. The DUT responds with **CreatePKCS10CSRResponse** message with parameters
   - PKCS10CSR =: *pkcs10*
5. ONVIF Client creates an CA certificate by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - in *cap* - DUT capabilities
   - in *caKeyAlgorithm* - key pair algorithm
   - out *CAcert* - CA certificate
   - out *privateKey* - private key for the CA certificate
   - out *publicKey* - public key for the CA certificate

6. Create an [RFC5280] compliant X.509 certificate (*cert*) from the PKCS#10 request (*pkcs10*) with the following properties:
    - version:= v3
    - signature := *signatureAlgorithm*
    - 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*)
7. ONVIF Client invokes **UploadCertificate** with parameters
    - Certificate := *cert*
    - Alias := "ONVIF_Test1"
    - PrivateKeyRequired := true
8. The DUT responds with a **UploadCertificateResponse** message with parameters
    - CertificateID =: *certID*
    - KeyID =: *keyID*
9. ONVIF Client invokes **CreateCertificationPath** with parameters
    - CertficateIDs.CertificateID[0] := *certID*
    - Alias := "ONVIF_Test2"
10. The DUT responds with a **CreateCertificationPathResponse** message with parameters
    - CertificationPathID =: *certPathID*
11. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
    - CertificationPathID := *certPathID*
12. The DUT responds with an **AddServerCertificateAssignmentResponse** message.
13. ONVIF Client waits for time *operationDelay*.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **CreatePKCS10CSRResponse** message.
- DUT did not send **UploadCertificateResponse** message.
- DUT did not send **CreateCertificationPathResponse** message.
- DUT did not send **AddServerCertificateAssignmentResponse** message.

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

A.49 Basic TLS Handshake With Protocol Version Alert

Name: HelperBasicTLSHandshakeProtocolVersionAlert

Procedure Purpose: Helper procedure to verify basic TLS handshake in case of "protocol_version" alert message.

Pre-requisite: TLS is supported by the DUT as indicated by the TLSServerSupported capability. TLS is configured. HTTPS protocol is enabled.

Input: Disabled TLS version (*disabledTLSVersion*).

Returns: None

Procedure:

1. ONVIF Client invokes **ClientHello** with parameters
   - ClientVersion := *disabledTLSVersion*
   - 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 **protocol_version** alert message and close connection.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **protocol_version** alert message.

A.50 Create a certification path validation policy for authentication server configuration

Name: HelperCreateCertPathValidationPolicyForAuthServer

Notes: Annex is used at:

- ONVIF Security Configuration Device Test Specification
- ONVIF Uplink Test Specification
- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Media2 Configuration Device Test Specification

Procedure Purpose: Helper procedure to create a certification path validation policy for authentication server configuration.

Pre-requisite: Security Configuration Service is received from the DUT. Certification path validation policy supported by the DUT as indicated by the MaximumNumberOfCertificationPathValidationPolicies capability. UploadCertificate is supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA or PKCS10 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 certificate. The DUT shall have enough free storage capacity for one additional key pair.

Input: The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*). The certification path validation policy alias (*certPathValidationPolicyAlias*). The subject (*subject*) of CA certificate (optional input parameter, could be skipped).

Returns: The certification path validation policy identifier (*certPathValidationPolicyID*), related certificate (*certID*), key pair (*keyID*), CA certificate (out *CAcert*) CA certificate private key (out *privateKey*).

Procedure:

1. ONVIF Client creates a CA certificate and a corresponding private key by following the procedure described in Annex A.8 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - out *CAcert* - CA certificate
   - out *privateKey* - private key
2. ONVIF Client invokes **UploadCertificate** with parameters
   - Certificate := *CAcert*
   - Alias := "ONVIF Test"
   - PrivateKeyRequired : = false
3. The DUT responds with a **UploadCertificateResponse** message with parameters
   - CertificateID =: *certID*
   - KeyID =: *keyID*
4. ONVIF Client creates certification path validation policy identifier with specified alias and the certificate identifier for trust anchor by following the procedure mentioned in Annex A.32 with the following input and output parameters:.
   - in *certID* - certificate identifier for trust anchor
   - in *certPathValidationPolicyAlias* - certification path validation policy alias
   - out *certPathValidationPolicyID* - certification path validation policy identifier

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.51 Make Sure That At Least One Authorization Server Configuration Could Be Created

Name: HelperEmptySpaceForOneAuthorizationServerConfiguration

Notes: Annex is used at:

- ONVIF Security Configuration Device Test Specification
- ONVIF Uplink Test Specification
- ONVIF Real Time Streaming using Media2 Device Test Specification

Procedure Purpose: Helper procedure to remove one authorization server configuration if maximum is reached.

Pre-requisite: Security Configuration Service is received from the DUT. Authorization Server Configuration is supported by the DUT as indicated by the AuthorizationServer.MaxConfigurations capability.

Input: The service capabilities (*cap*).

Returns: Removed authorization server configuration (*removedAuthServerConfiguration*), could be empty.

Procedure:

1. ONVIF Client gets current authorization server configurations by following the procedure mentioned in Annex A.52 with the following input and output parameters:
   - out *authServerConfigurations1* list - authorization server configurations
2. If *authServerConfigurations1* items number is equal to *cap*.AuthorizationServer.MaxConfigurations:

   2.1. Set the following:
   - *removedAuthServerConfiguration* := *authServerConfigurations1*[0]

   2.2. ONVIF Client invokes **DeleteAuthorizationServerConfiguration** with parameter
   - Token := *removedAuthServerConfiguration*.token

   2.3. The DUT responds with **DeleteAuthorizationServerConfigurationResponse** message.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **DeleteAuthorizationServerConfigurationResponse** message.

A.52 Get Authorization Server Configurations List

Name: HelperGetAuthorizationServerConfigurations

Notes: Annex is used at:

- ONVIF Security Configuration Device Test Specification
- ONVIF Uplink Test Specification
- ONVIF Real Time Streaming using Media2 Device Test Specification

Procedure Purpose: Helper procedure to get authorization server configurations list.

Pre-requisite: Security Configuration Service is received from the DUT. Authorization Server Configuration is supported by the DUT as indicated by the AuthorizationServer.MaxConfigurations capability.

Input: None

Returns: Authorization server configurations list (*authServerConfigurations*).

Procedure:

1. ONVIF Client invokes **GetAuthorizationServerConfigurations** request with parameters
    - Token is skipped
2. The DUT responds with **GetAuthorizationServerConfigurationsResponse** message with parameters
    - Configuration list =: *authServerConfigurations*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetAuthorizationServerConfigurationsResponse** message.