ONVIF®

Uplink Device Test Specification

Version 26.06

June 2026

REVISION HISTORY

Vers.	Date	Description
23.06	Jan 04, 2023	Initial version
25.06	Mar 04, 2025	The following new test cases were added as a part of #301 task: UPLINK-3-1-1 Uplink over WebSocket with mTLS Authentication (new) UPLINK-3-1-2 Uplink over WebSocket with access token authentication (OAuthClientCredentials, client_secret_basic) (new) UPLINK-3-1-3 Uplink over WebSocket with access token authentication (OAuthClientCredentials, private_key_jwt) (new) UPLINK-3-1-4 Uplink over WebSocket with mTLS Authentication - Invalid Server Certificate (new) UPLINK-3-1-5 Uplink over WebSocket with access token authentication (OAuthClientCredentials, client_secret_basic) - Invalid Uplink Client Certificate (new) UPLINK-3-1-6 Uplink over WebSocket with access token authentication (OAuthClientCredentials, client_secret_basic) - Invalid Authentication Server Certificate (new) Annex HelperCreateKeyPairAndReceivePublicKey Create Key Pair and Receive Public Key (new) Annex HelperGetAuthorizationServerConfigurations Get Authorization Server Configurations List (added) Annex HelperEmptySpaceForOneAuthorizationServerConfiguration Make Sure That At Least One Authorization Server Configuration Could Be Created (added) Annex HelperCreateCertPathValidationPolicyWithCertID Create a certification path validation policy with provided certificate identifier (added) Annex HelperCreateSignedCertificate Provide certificate signed by private key of other certificate (added) Annex HelperCreateCertPathValidationPolicyForAuthServer Create a certification path validation policy for authentication server configuration (added) Introduction\Scope\Authentication and Authorization (new) Test Overview\Test Policy\Authentication and Authorization (new)
25.12	Oct 02, 2025	The following test and annexes were updated in the scope of #429 (Authorization server uri was replaced with Authentication server metadata endpoint): Annex HelperConfigureAndStartAuthServer Configure Authorization Server On Device and Start It UPLINK-3-1-2 Uplink over WebSocket with access token authentication (OAuthClientCredentials, client_secret_basic)

		UPLINK-3-1-3 Uplink over WebSocket with access token authentication (OAuthClientCredentials, private_key_jwt) UPLINK-3-1-5 Uplink over WebSocket with access token authentication (OAuthClientCredentials, client_secret_basic) - Invalid Uplink Client Certificate UPLINK-3-1-6 Uplink over WebSocket with access token authentication (OAuthClientCredentials, client_secret_basic) - Invalid Authentication Server Certificate
26.06	Feb 17, 2026	The following test case/annexes were updated/added according #463: HelperStorageConfigurationPrecondition (key pair algorithm selection logic was updated to use HelperKeyAlgorithmSelection) HelperKeyAlgorithmSelection (new) Editorial changes that not affect tests implementation
26.06	Feb 17, 2026	The following test case/annexes were updated/added according #463: HelperStorageConfigurationPrecondition (key pair algorithm selection logic was updated to use HelperKeyAlgorithmSelection) HelperKeyAlgorithmSelection (new) Editorial changes that not affect tests implementation
26.06	Feb 17, 2026	The following test case/annexes were updated/added according #463: UPLINK-2-1-1 Uplink Configuration Management (steps 4 and 5 were added for key pair algorithm selection using HelperKeyAlgorithmSelection) UPLINK-3-1-1 Uplink over WebSocket with mTLS Authentication (key pair algorithm selection logic was updated to use HelperKeyAlgorithmSelection) UPLINK-3-1-2 Uplink over WebSocket with access token authentication (OAuthClientCredentials, client_secret_basic) (key pair algorithm selection logic was updated to use HelperKeyAlgorithmSelection) UPLINK-3-1-3 Uplink over WebSocket with access token authentication (OAuthClientCredentials, private_key_jwt) (key pair algorithm selection logic was updated to use HelperKeyAlgorithmSelection) UPLINK-3-1-4 Uplink over WebSocket with mTLS Authentication - Invalid Server Certificate (key pair algorithm selection logic was updated to use HelperKeyAlgorithmSelection) UPLINK-3-1-5 Uplink over WebSocket with access token authentication (OAuthClientCredentials, client_secret_basic) - Invalid Uplink Client Certificate (key pair algorithm selection logic was updated to use HelperKeyAlgorithmSelection) UPLINK-3-1-6 Uplink over WebSocket with access token authentication (OAuthClientCredentials, client_secret_basic) - Invalid Authentication Server Certificate (key pair algorithm selection logic was updated to use HelperKeyAlgorithmSelection)

		HelperConfigureUplinkSecurity (key pair algorithm selection logic was updated to use HelperKeyAlgorithmSelection) HelperKeyAlgorithmSelection (new) Editorial changes that not affect tests implementation
26.06	Apr 21, 2026	The following test case was updated according #479: UPLINK-1-1-2 Connect And Disconnect (native h2c-reverse) (step for cipher selection was added, uplink server start step was updated with ciphers list) UPLINK-1-2-1 Connect And Disconnect (h2c-reverse over WebSocket) (step for cipher selection was added, uplink server start step was updated with ciphers list) UPLINK-3-1-3 Uplink over WebSocket with access token authentication (OAuthClientCredentials, private_key_jwt, RS256) (test was updated to use RS256 JWT algorithm) HelperCreateJWTKeyPairAndReceivePublicKey (new) HelperCreateKeyPair (key parameters optional input was added, capabilities become optional input) UPLINK-3-1-7 Uplink over WebSocket with access token authentication (OAuthClientCredentials, private_key_jwt, ES256) (new)
26.06	Mar 15, 2026	The following test case was updated according #517: HelperCreateJWTKeyPairAndReceivePublicKey (possibility to generate key pair for JWT by CSR key pair algorithm was added) HelperConfigureUplinkSecurity (key pair generation helper for private_key_jwt was replaced with HelperCreateJWTKeyPairAndReceivePublicKey) Editorial changes

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 Uplink Specification] and [ONVIF Conformance] requirements. It also describes the test framework, test setup, pre-requisites, test policies needed for the execution of the described test cases.

This ONVIF Uplink Device Test Specification acts as a supplementary document to the [ONVIF Uplink Specification], illustrating test cases need to be executed and passed. And also this specification acts as an input document to the development of test tool which 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 Uplink 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 ONVIF Uplink which is defined in [ONVIF Uplink Specification].

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).
- Network protocol implementation Conformance test for HTTP, HTTPS, RTP and RTSP protocol.
- Poor streaming performance test (audio/video distortions, missing audio/video frames, incorrect lib synchronization etc.).

  Wi-Fi Conformance test

The set of ONVIF Test Specification will not cover the complete set of requirements as defined in [ONVIF Uplink Specification]; instead it would cover subset of it. The scope of this specification is to

derive all the normative requirements of [ONVIF Uplink Specification] which are related to ONVIF Uplink and some of the optional requirements.

This ONVIF Uplink Device Test Specification covers Uplink service which is a functional block of [ONVIF Network Interface Specs]. The following sections give the brief overview of and scope of each functional block.

1.1.1 Uplink Connection

Uplink Connection section covers the test cases needed for uplink connection management to remote client.

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

- Connection establishment to remote client
  - Native h2c-reverse connection
  - h2c-reverse over WebSocket connection
- Automatic reconnection connection was interrupted
- Concurrent uplink connections

1.1.2 Uplink Configuration

Uplink Connection section covers the test cases needed for uplink configuration management.

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

- Uplink configuration creation
- Uplink configuration update
- Uplink configuration deletion

1.1.3 Authentication and Authorization

The Authentication and Authorization section covers the test cases needed for authentication and authorization for uplink connection.

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

- mTLS authentication
- Access token authentication with following settings:

  - OAuthClientCredentials, client_secret_basic
  - OAuthClientCredentials, private_key_jwt
- Uplink authorization
- Uplink client certificate validation
- Authorization server certificate validation

1.1.4 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 Uplink service address with GetServices command via Device service
- Getting capabilities with GetServiceCapabilities command
- Getting capabilities with GetServices command via Device service

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 Uplink Specification] Uplink Specification:
  https://www.onvif.org/profiles/specifications/
- [ONVIF Base Test] ONVIF Base Device Test Specification:
  https://www.onvif.org/profiles/conformance/device-test/
- [ONVIF RTSP via Media2 Test] Real Time Streaming using Media2 Test Specification:
  https://www.onvif.org/profiles/specifications/
- [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

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 Uplink Service and tests. For the list of applicable general terms and definitions, please see [ONVIF Base Test].

Uplink Doing something in advance to prepare for something else.

Video Source Entity defined by [ONVIF Device I/O Specification]

Video Source Token Token referencing a Device I/O Video Source

3.3 Abbreviations

This section describes abbreviations used in this document.

HTTP Hyper Text Transport Protocol.

AAC Advanced Audio Coding.

URI Uniform Resource Identifier.

WSDL Web Services Description Language.

XML eXtensible Markup Language.

JPEG Joint Photographic Experts Group.

TTL Time To Live.

4 Test Overview

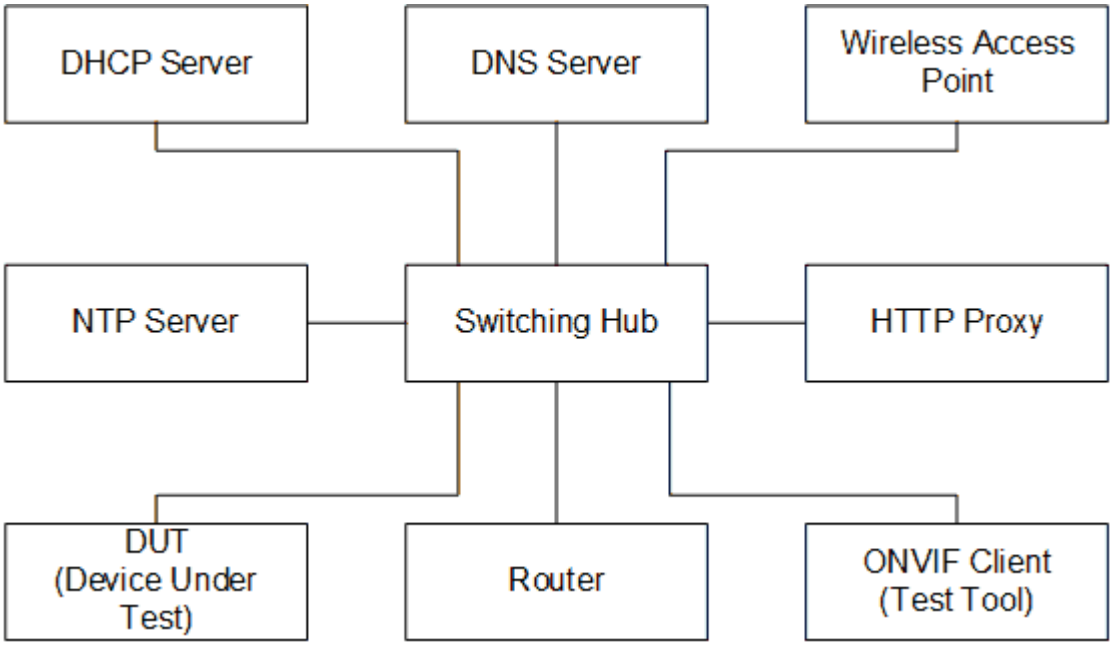
This section provides information about the test setup procedure and required prerequisites, and the test policies that should be followed for test case execution.

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 1).

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.

Service Endpoint: Cloud connection endpoint according to the ONVIF Uplink Specification to which the DUT can connect.

4.2 Prerequisites

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

1. The DUT shall be configured with an IPv4 address.
2. The DUT shall be IP reachable [in the test configuration].
3. The DUT shall be able to be discovered by the Test Tool.
4. 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.
5. The DUT time and Test tool time shall be synchronized with each other either manually or by a common NTP server.

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.

DUT shall provide Uplink Service entry point by GetServices command, if DUT supports this service. Otherwise, test cases of the documnet will be skipped.

4.3.1 Uplink Connection

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

- The DUT shall automatically establish connection to the remote client if uplink configuration was added or changed.
- The DUT shall automatically reestablish connection to the remote client if connection was interrupted.
- The DUT shall close and reconnect the uplink whenever no packets have been received from the remote client for more than 30 seconds.
- The DUT shall signal connection state via Status field of Uplink Configuration.
- The DUT shall support concurrent uplink connections up to specified in MaxUplink capability.
- The DUT shall provide access to all supported services through uplink connection.
- The DUT signaling support for wss via its Protocol capability shall support h2c-reverse over WebSocket.
- The DUT signaling support for https via its Protocol capability shall support native h2c-reverse.

Please refer to Section 5.1 for Uplink Test Cases.

4.3.2 Uplink Configuration

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

- The DUT shall allow to retrieve uplink configurations with GetUplinks command.
- The DUT shall allow creation of uplink configuration with SetUplink command.
- The DUT shall allow update of uplink configuration with SetUplink command.
- The Status property of the UplinkConfiguration shall be ignored by the DUT at SetUplink request.
- The DUT shall use the field RemoteAddress to decide whether to update an existing entry or create a new entry.
- The DUT shall interpret the field RemoteAddress as key to a specific uplink configuration.
- The DUT shall allow creation of uplink configurations up two number defined in MaxUplinks capability.
- The DUT shall allow deletion of uplink configurations with DeleteUplink command.
- The DUT shall reject a configuration containing both CertificateID and AuthorizationServer.

Please refer to Section 5.2 for Uplink Test Cases.

4.3.3 Authentication and Authorization

The test policies specific to the test case execution of Authentication and Authorization functional block:

- DUT shall give the ONVIF Uplink Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.
- 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.
- DUT shall support Uplink over WebSocket connection. Otherwise, these test cases will be skipped.
- If DUT supports access token authentication, it shall support Authorization Server Configuration. Otherwise, these test cases will be failed.

- The DUT shall support certification path validation policy.
- The following tests are performed about Authentication and Authorization
  - The DUT will not establish connection if uplink client certificate does not pass certification path validation policy configured at uplink connection.
  - The DUT will not require additional authentication for requests over uplink connection.
  - If DUT supports mTLS authentication:
    - The DUT will establish connection with mTLS authentication configured.
  - If DUT supports access token authentication:
    - The DUT will establish connection with access token authentication configured with the following configurations:
      - OAuthClientCredentials, client_secret_basic (if this combination is supported)
      - OAuthClientCredentials, private_key_jwt (if this combination is supported)
    - The DUT will re-use access token if it was not expired.
    - The DUT will renew access token if it was rejected by the server.
    - The DUT will not establish connection if authentication server does not pass certification path validation policy configured at authentication server configuration.

Please, refer to Section 5.3 for Authentication and Authorization Test Cases.

4.3.4 Capabilities

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

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

- Getting capabilities with GetServices command

Please refer to Section 5.4 for Uplink Test Cases.

5 Uplink Test Cases

5.1 Uplink Connection

5.1.1 Native h2c-reverse

5.1.1.1 Connect And Disconnect (native h2c-reverse)

Test Case ID: UPLINK-1-1-2

Specification Coverage: Native h2c-reverse (ONVIF Uplink Specification)

Feature Under Test: Native h2c-reverse connection.

WSDL Reference: uplink.wsdl, advancedsecurity.wsdl

Test Purpose:

- To verify that a DUT can connect to remote client using uplink native h2c-reverse connection.
- To verify that a DUT automatically restores uplink connection.
- To verify that a DUT reports valid uplink connection state.
- To verify that a DUT supports TLS ciphers that required by ONVIF Security Baseline.

Pre-Requisite:

- Security Configuration Service is received from the DUT.
- Uplink Service is received from the DUT.
- Uplink native h2c-reverse connection is supported by the DUT as indicated by the Protocols = https capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the uplink service capabilities by following the procedure mentioned in Annex A.10 with the following input and output parameters:
   - out *capUplink* - Uplink Service Capabilities

4. ONVIF Client tries to configure uplink:

   4.1. ONVIF Client configures security primitives needed for uplink connection and starts required servers by following the procedure described in Annex A.1 with the following input and output parameters:

   - in *capUplink* - uplink capabilities
   - out *certPathValidationPolicyIDUplink* - certification path validation policy for uplink connection
   - out *certID1* - certificate ID to be used for mTLS authentication (optional)
   - out *authServerToken1* - Authorization server configuration that could be used for access token authentication (optional)

   4.2. ONVIF Client removes any existing uplink configurations by following the procedure mentioned in Annex A.19.

   4.3. ONVIF Client invokes **SetUplink** request with parameters

   - RemoteAddress := *uplinkAddress1* (address with https schema which will be used for uplink client at step 9)
   - CertificateID := *certID1*, if *certID1* was defined, otherwise field is skipped
   - UserLevel := **Administrator**
   - Status is skipped
   - CertPathValidationPolicyID := *certPathValidationPolicyIDUplink*
   - AuthorizationServer := *authServerToken1*, if *authServerToken1* was defined, otherwise field is skipped
   - Error is skipped

   4.4. The DUT responds with **SetUplinkResponse** message.

   4.5. If *authServerToken1* was used for configuration:

   4.5.1. ONVIF Client starts Authorization server with metadata endpoint *authServerMetadataEndpoint1* conforming to RFC8414 with settings which corresponds to create authentication server configuration (*authServerToken1*).

5. ONVIF Client invokes **GetUplinks**.

6. The DUT responds with a **GetUplinksResponse** message with parameters:
   - Configuration list =: *configurationList1*
7. If *configurationList1* item with RemoteAddress = *uplinkAddress1* has Status field value other then 'Offline' or 'Connecting', FAIL the test, restore the DUT state, and skip other steps.
8. ONVIF Client checks which TLS Ciphers (*ciphersList*) the DUT shall support depending on required Security Baseline versions by the rules defined in section TLS Cipher Requirements of [ONVIF Security Baseline Device Test Spec].
9. ONVIF Client start service endpoint for uplink connection at address *uplinkAddress1* which accepts authentication defined for configuration created at step 4.3 and with accepted TLS ciphers limited to *ciphersList*.
10. ONVIF Client waits for the DUT to establish connection to *uplinkAddress1* endpoint.
11. ONVIF Client invokes **GetUplinks**.
12. The DUT responds with a **GetUplinksResponse** message with parameters:
    - Configuration list =: *configurationList2*
13. If *configurationList2* item with RemoteAddress = *uplinkAddress1* has Status field value other then 'Connected', FAIL the test, restore the DUT state, and skip other steps.
14. ONVIF Client stops uplink server.
15. ONVIF Client invokes **GetUplinks**.
16. The DUT responds with a **GetUplinksResponse** message with parameters:
    - Configuration list =: *configurationList3*
17. If *configurationList3* item with RemoteAddress = *uplinkAddress1* has Status field value other then 'Offline' or 'Connecting', FAIL the test, restore the DUT state, and skip other steps.
18. ONVIF Client repeat steps from 9 to 17 3 more times.
19. ONVIF Client restores the DUT state.

Test Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **SetUplinkResponse** message.
- DUT did not send **GetUplinkResponse** message(s).
- DUT did not establish uplink connection at step 10 within *timeout1*.

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

Note: *configurationList2* checking repeats from step 11 untill get required value during operation delay timeout

Note: *configurationList3* checking repeats from step 15 untill get required value during operation delay timeout

5.1.2 h2c-reverse over WebSocket

5.1.2.1 Connect And Disconnect (h2c-reverse over WebSocket)

Test Case ID: UPLINK-1-2-1

Specification Coverage: WebSocket h2c-reverse (ONVIF Uplink Specification)

Feature Under Test: WebSocket h2c-reverse connection.

WSDL Reference: uplink.wsdl, advancedsecurity.wsdl

Test Purpose:

- To verify that a DUT can connect to remote client using uplink h2c-reverse over WebSocket connection.
- To verify that a DUT automatically restores uplink connection.
- To verify that a DUT reports valid uplink connection state.
- To verify that a DUT supports TLS ciphers that required by ONVIF Security Baseline.

Pre-Requisite:

- Security Configuration Service is received from the DUT.
- Uplink Service is received from the DUT.

- Uplink h2c-reverse over WebSocket connection is supported by the DUT as indicated by the Protocols = wss capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the uplink service capabilities by following the procedure mentioned in Annex A.10 with the following input and output parameters:
   - out *capUplink* - Uplink Service Capabilities
4. ONVIF Client tries to configure uplink:
   4.1. ONVIF Client configures security primitives needed for uplink connection and starts required servers by following the procedure described in Annex A.1 with the following input and output parameters:
      - in *capUplink* - uplink capabilities
      - out *certPathValidationPolicyIDUplink* - certification path validation policy for uplink connection
      - out *certID1* - certificate ID to be used for mTLS authentication (optional)
      - out *authServerToken1* - Authorization server configuration that could be used for access token authentication (optional)
   4.2. ONVIF Client removes any existing uplink configurations by following the procedure mentioned in Annex A.19.
   4.3. ONVIF Client invokes **SetUplink** request with parameters
      - RemoteAddress := *uplinkAddress1* (address with wss schema which will be used for uplink client at step 9)
      - CertificateID := *certID1*, if *certID1* was defined, otherwise field is skipped
      - UserLevel := **Administrator**
      - Status is skipped
      - CertPathValidationPolicyID := *certPathValidationPolicyIDUplink*

- AuthorizationServer := *authServerToken1*, if *authServerToken1* was defined, otherwise field is skipped
- Error is skipped

4.4. The DUT responds with **SetUplinkResponse** message.

4.5. If *authServerToken1* was used for configuration:

4.5.1. ONVIF Client starts Authorization server with metadata endpoint *authServerMetadataEndpoint1* conforming to RFC8414 with settings which corresponds to create authentication server configuration (*authServerToken1*).

5. ONVIF Client invokes **GetUplinks**.
6. The DUT responds with a **GetUplinksResponse** message with parameters:
   - Configuration list =: *configurationList1*
7. If *configurationList1* item with RemoteAddress = *uplinkAddress1* has Status field value other then 'Offline' or 'Connecting', FAIL the test, restore the DUT state, and skip other steps.
8. ONVIF Client checks which TLS Ciphers (*ciphersList*) the DUT shall support depending on required Security Baseline versions by the rules defined in section TLS Cipher Requirements of [ONVIF Security Baseline Device Test Spec].
9. ONVIF Client start service endpoint for uplink connection at address *uplinkAddress1* which accepts authentication defined for configuration created at step 4.3 and with accepted TLS ciphers limited to *ciphersList*.
10. ONVIF Client waits for the DUT to establish connection to *uplinkAddress1* endpoint.
11. ONVIF Client invokes **GetUplinks**.
12. The DUT responds with a **GetUplinksResponse** message with parameters:
    - Configuration list =: *configurationList2*
13. If *configurationList2* item with RemoteAddress = *uplinkAddress1* has Status field value other then 'Connected', FAIL the test, restore the DUT state, and skip other steps.
14. ONVIF Client stops uplink server.
15. ONVIF Client invokes **GetUplinks**.
16. The DUT responds with a **GetUplinksResponse** message with parameters:

- Configuration list =: *configurationList3*

17. If *configurationList3* item with RemoteAddress = *uplinkAddress1* has Status field value other then 'Offline' or 'Connecting', FAIL the test, restore the DUT state, and skip other steps.
18. ONVIF Client repeat steps from 9 to 17 3 more times.
19. ONVIF Client restores the DUT state.

Test Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **SetUplinkResponse** message.
- DUT did not send **GetUplinkResponse** message(s).
- DUT did not establish uplink connection at step 10 within *timeout1*.

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

Note: *configurationList2* checking repeats from step 11 untill get required value during operation delay timeout

Note: *configurationList3* checking repeats from step 15 untill get required value during operation delay timeout

5.1.2.2 Automatic Reconnection After 30 Seconds Timeout (h2c-reverse over WebSocket)

Test Case ID: UPLINK-1-2-2

Specification Coverage: WebSocket h2c-reverse (ONVIF Uplink Specification), Connection Management (ONVIF Uplink Specification)

Feature Under Test: WebSocket h2c-reverse connection.

WSDL Reference: uplink.wsdl, advancedsecurity.wsdl

Test Purpose:

- To verify that a DUT close and reconnect the uplink whenever no packets have been received from the remote client for more than 30 seconds.
- To verify that a DUT keep the uplink connection if packets have been received from the remote client within 30 seconds.

Pre-Requisite:

- Security Configuration Service is received from the DUT.
- Uplink Service is received from the DUT.
- Uplink h2c-reverse over WebSocket connection is supported by the DUT as indicated by the Protocols = wss capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the uplink service capabilities by following the procedure mentioned in Annex A.10 with the following input and output parameters:
   - out *capUplink* - Uplink Service Capabilities
4. ONVIF Client tries to configure uplink:
   4.1. ONVIF Client configures security primitives needed for uplink connection and starts required servers by following the procedure described in Annex A.1 with the following input and output parameters:
      - in *capUplink* - uplink capabilities
      - out *certPathValidationPolicyIDUplink* - certification path validation policy for uplink connection
      - out *certID1* - certificate ID to be used for mTLS authentication (optional)
      - out *authServerToken1* - Authorization server configuration that could be used for access token authentication (optional)
   4.2. ONVIF Client removes any existing uplink configurations by following the procedure mentioned in Annex A.19.

4.3. ONVIF Client invokes **SetUplink** request with parameters

- RemoteAddress := *uplinkAddress1* (address with wss schema which will be used for uplink client at step 8)
- CertificateID := *certID1*, if *certID1* was defined, otherwise field is skipped
- UserLevel := **Administrator**
- Status is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDUplink*
- AuthorizationServer := *authServerToken1*, if *authServerToken1* was defined, otherwise field is skipped
- Error is skipped

4.4. The DUT responds with **SetUplinkResponse** message.

4.5. If *authServerToken1* was used for configuration:

4.5.1. ONVIF Client starts Authorization server with metadata endpoint *authServerMetadataEndpoint1* conforming to RFC8414 with settings which corresponds to create authentication server configuration (*authServerToken1*).

5. ONVIF Client invokes **GetUplinks**.
6. The DUT responds with a **GetUplinksResponse** message with parameters:
   - Configuration list =: *configurationList1*
7. If *configurationList1* item with RemoteAddress = *uplinkAddress1* has Status field value other then 'Offline' or 'Connecting', FAIL the test, restore the DUT state, and skip other steps.
8. ONVIF Client start service endpoint for uplink connection at address *uplinkAddress1* which accepts authentication defined for configuration created at step 4.3.
9. ONVIF Client waits for the DUT to establish connection to *uplinkAddress1* endpoint.
10. ONVIF Client waits for 30 seconds after the DUT established connection without any requests to the DUT.
11. The DUT closed connection.
12. ONVIF Client waits for the DUT to establish connection to *uplinkAddress1* endpoint.

13. ONVIF Client waits for 60 seconds after the DUT established connection with sending **GetDeviceInformation** request as keep alive message.
14. ONVIF Client restores the DUT state.

Test Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **SetUplinkResponse** message.
- DUT did not establish uplink connection at step 9 within *timeout1*.
- DUT did not close uplink connection at step 11 within *timeout1*.
- DUT did not restore uplink connection at step 12 within *timeout1*.
- DUT closed connection during step 13.

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

5.1.2.3 Multiple Uplink Connections (h2c-reverse over WebSocket)

Test Case ID: UPLINK-1-2-3

Specification Coverage: WebSocket h2c-reverse (ONVIF Uplink Specification), Capabilities (ONVIF Uplink Specification)

Feature Under Test: WebSocket h2c-reverse connection.

WSDL Reference: uplink.wsdl, advancedsecurity.wsdl

Test Purpose:

- To verify that a DUT can support concurrent connections close as defined by MaxUplinks capability.

Pre-Requisite:

- Security Configuration Service is received from the DUT.
- Uplink Service is received from the DUT.
- Uplink h2c-reverse over WebSocket connection is supported by the DUT as indicated by the Protocols = wss capability.

- At least 2 uplink connections are supported by MaxUplinks > 1 capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the uplink service capabilities by following the procedure mentioned in Annex A.10 with the following input and output parameters:
   - out *capUplink* - Uplink Service Capabilities
4. Set:
   4.1. *numberOfConnection* := minimal from 3 and *capUplink*.MaxUplinks
5. ONVIF Client tries to configure uplink:
   5.1. ONVIF Client configures security primitives needed for uplink connection and starts required servers by following the procedure described in Annex A.1 with the following input and output parameters:
      - in *capUplink* - uplink capabilities
      - out *certPathValidationPolicyIDUplink* - certification path validation policy for uplink connection
      - out *certID1* - certificate ID to be used for mTLS authentication (optional)
      - out *authServerToken1* - Authorization server configuration that could be used for access token authentication (optional)
   5.2. ONVIF Client removes any existing uplink configurations by following the procedure mentioned in Annex A.19.
   5.3. ONVIF Client creates number of Uplinks Configurations equal to *numberOfConnection*:
      5.3.1. ONVIF Client invokes **SetUplink** request with parameters
         - RemoteAddress := *uplinkAddressN* (address with wss schema which will be used for uplink client at step 6, each configuration shall have unique address)
         - CertificateID := *certID1*, if *certID1* was defined, otherwise field is skipped

- UserLevel := **Administrator**
- Status is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDUplink*
- AuthorizationServer := *authServerToken1*, if *authServerToken1* was defined, otherwise field is skipped
- Error is skipped

5.3.2. The DUT responds with **SetUplinkResponse** message.

5.4. If *authServerToken1* was used for configuration:

5.4.1. ONVIF Client starts Authorization server with metadata endpoint *authServerMetadataEndpoint1* conforming to RFC8414 with settings which corresponds to create authentication server configuration (*authServerToken1*).

6. ONVIF Client starts service endpoint for uplink connection for each address *uplinkAddressN* that was used at step 4.3 which accepts authentication defined for configuration created at step 4.3.
7. ONVIF Client waits for the DUT to establish connection to all *uplinkAddressN* endpoints.
8. ONVIF Client waits for 60 seconds after the DUT established connection with sending **GetDeviceInformation** request as keep alive message for all opened connections.
9. ONVIF Client restores the DUT state.

Test Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **SetUplinkResponse** message.
- DUT did not establish at least one uplink connection at step 7 within *timeout1*.
- DUT closed at least one connection during step 8.

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

5.1.2.4 Services Accessibility Through Uplink Connection (h2c-reverse over WebSocket)

Test Case ID: UPLINK-1-2-4

Specification Coverage: WebSocket h2c-reverse (ONVIF Uplink Specification), Connection Management (ONVIF Uplink Specification)

Feature Under Test: WebSocket h2c-reverse connection.

WSDL Reference: uplink.wsdl, advancedsecurity.wsdl

Test Purpose:

- To verify that a all services supported by the DUT is accessible with uplink connection.

Pre-Requisite:

- Security Configuration Service is received from the DUT.
- Uplink Service is received from the DUT.
- Uplink h2c-reverse over WebSocket connection is supported by the DUT as indicated by the Protocols = wss capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the uplink service capabilities by following the procedure mentioned in Annex A.10 with the following input and output parameters:
   - out *capUplink* - Uplink Service Capabilities
4. ONVIF Client tries to configure uplink:
   4.1. ONVIF Client configures security primitives needed for uplink connection and starts required servers by following the procedure described in Annex A.1 with the following input and output parameters:
      - in *capUplink* - uplink capabilities
      - out *certPathValidationPolicyIDUplink* - certification path validation policy for uplink connection

- out *certID1* - certificate ID to be used for mTLS authentication (optional)
- out *authServerToken1* - Authorization server configuration that could be used for access token authentication (optional)

4.2. ONVIF Client removes any existing uplink configurations by following the procedure mentioned in Annex A.19.

4.3. ONVIF Client invokes **SetUplink** request with parameters

- RemoteAddress := *uplinkAddress1* (address with wss schema which will be used for uplink client at step 5)
- CertificateID := *certID1*, if *certID1* was defined, otherwise field is skipped
- UserLevel := **Administrator**
- Status is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDUplink*
- AuthorizationServer := *authServerToken1*, if *authServerToken1* was defined, otherwise field is skipped
- Error is skipped

4.4. The DUT responds with **SetUplinkResponse** message.

4.5. If *authServerToken1* was used for configuration:

4.5.1. ONVIF Client starts Authorization server with metadata endpoint *authServerMetadataEndpoint1* conforming to RFC8414 with settings which corresponds to create authentication server configuration (*authServerToken1*).

5. ONVIF Client start service endpoint for uplink connection at address *uplinkAddress1* which accepts authentication defined for configuration created at step 4.3.
6. ONVIF Client waits for the DUT to establish connection to *uplinkAddress1* endpoint.
7. ONVIF Client communicates through established connection:

7.1. ONVIF Client invokes **GetServices** with parameters

- IncludeCapability := false

7.2. The DUT responds with a **GetServicesResponse** message with parameters

- Services list =: *servicesList*

7.3. For each service at *servicesList*:

7.3.1. ONVIF Client invokes **GetServiceCapabilities** message for corresponding service.

7.3.2. The DUT responds with a **GetServicesResponse** message.

8. ONVIF Client restores the DUT state.

Test Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **SetUplinkResponse** message.
- DUT did not send **GetServicesResponse** message.
- DUT did not send **GetServiceCapabilitiesResponse** message for at least one service.
- DUT did not establish uplink connection at step 6 within *timeout1*.

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

Note: GetServiceCapabilities the following services is supported for step 7.3.

- Device Management Service
- Event Service
- Analytics Service
- Device IO Service
- Imaging Service
- Media 2 Service
- PTZ Service
- Receiver Service
- Recording Control Service
- Uplink Service

5.1.2.5 Possibility to delete configuration for open uplink connection (h2c-reverse over WebSocket)

Test Case ID: UPLINK-1-2-5

Specification Coverage: WebSocket h2c-reverse (ONVIF Uplink Specification), Connection Management (ONVIF Uplink Specification)

Feature Under Test: WebSocket h2c-reverse connection.

WSDL Reference: uplink.wsdl, advancedsecurity.wsdl

Test Purpose:

- To verify that it is possible to delete uplink configuration if connection was opened.

Pre-Requisite:

- Security Configuration Service is received from the DUT.
- Uplink Service is received from the DUT.
- Uplink h2c-reverse over WebSocket connection is supported by the DUT as indicated by the Protocols = wss capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the uplink service capabilities by following the procedure mentioned in Annex A.10 with the following input and output parameters:
   - out *capUplink* - Uplink Service Capabilities
4. ONVIF Client tries to configure uplink:
   4.1. ONVIF Client configures security primitives needed for uplink connection and starts required servers by following the procedure described in Annex A.1 with the following input and output parameters:
      - in *capUplink* - uplink capabilities
      - out *certPathValidationPolicyIDUplink* - certification path validation policy for uplink connection

- out *certID1* - certificate ID to be used for mTLS authentication (optional)
- out *authServerToken1* - Authorization server configuration that could be used for access token authentication (optional)

4.2. ONVIF Client removes any existing uplink configurations by following the procedure mentioned in Annex A.19.

4.3. ONVIF Client invokes **SetUplink** request with parameters

- RemoteAddress := *uplinkAddress1* (address with wss schema which will be used for uplink client at step 5)
- CertificateID := *certID1*, if *certID1* was defined, otherwise field is skipped
- UserLevel := **Administrator**
- Status is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDUplink*
- AuthorizationServer := *authServerToken1*, if *authServerToken1* was defined, otherwise field is skipped
- Error is skipped

4.4. The DUT responds with **SetUplinkResponse** message.

4.5. If *authServerToken1* was used for configuration:

4.5.1. ONVIF Client starts Authorization server with metadata endpoint *authServerMetadataEndpoint1* conforming to RFC8414 with settings which corresponds to create authentication server configuration (*authServerToken1*).

5. ONVIF Client start service endpoint for uplink connection at address *uplinkAddress1* which accepts authentication defined for configuration created at step 4.3.
6. ONVIF Client waits for the DUT to establish connection to *uplinkAddress1* endpoint.
7. ONVIF Client invokes **DeleteUplink** request with parameters
   - RemoteAddress := *uplinkAddress1*
8. The DUT responds with **DeleteUplinkResponse** message.
9. ONVIF Client invokes **GetUplinks**.

10. The DUT responds with a **GetUplinksResponse** message with parameters:
    - Configuration list =: *configurationList1*
11. If *configurationList1* does contains item with *uplinkAddress1*, FAIL the test, restore the DUT state, and skip other steps.
12. ONVIF Client restores the DUT state.

Test Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **SetUplinkResponse** message.
- DUT did not send **DeleteUplinkResponse** message.
- DUT did not send **GeUplinksResponse** message.
- DUT did not establish uplink connection at step 6 within *timeout1*.

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

5.2 Uplink Configuration

5.2.1 Uplink Configuration Management

Test Case ID: UPLINK-2-1-1

Specification Coverage: GetUplinks (ONVIF Uplink Specification), SetUplink (ONVIF Uplink Specification), DeleteUplink (ONVIF Uplink Specification)

Feature Under Test: Uplink Configuration Creation, Uplink Configuration Update, Uplink Configuration Deletion.

WSDL Reference: uplink.wsdl, advancedsecurity.wsdl

Test Purpose:

- To verify that a DUT can create new uplink configuration by using SetUplink command and returning it back by using GetUplinks command.
- To verify that a DUT can modify existing uplink configuration by using SetUplink command.

- To verify that a DUT can delete existing uplink configuration by using DeleteUplink command.

Pre-Requisite:

- Security Configuration Service is received from the DUT.
- Uplink 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 gets the uplink service capabilities by following the procedure mentioned in Annex A.10 with the following input and output parameters:
   - out *capUplink* - Uplink Service Capabilities
4. ONVIF Client gets the security service capabilities by following the procedure mentioned in Annex A.8 with the following input and output parameters:
   - out *capSecurity* - Security Service Capabilities
5. ONVIF Client selects key pair algorithm by following the procedure mentioned in Annex A.17 with the following input and output parameters:
   - in *cap* - security configuration service capabilities
   - out *keyAlgorithm* - key pair algorithm
6. Set:
   - in *uplinkAddress1* := any valid address with wss schema if *capUplink*.Protocols contains 'wss', otherwise with https schema to be used.
   - in *uplinkAddress2* := any valid address (other then *uplinkAddress1*) with wss schema if *capUplink*.Protocols contains 'wss', otherwise with https schema to be used.
7. ONVIF Client configures security primitives needed for uplink connection and starts required servers by following the procedure described in Annex A.1 with the following input and output parameters:
   - in *capUplink* - uplink capabilities
   - out *certPathValidationPolicyIDUplink* - certification path validation policy for uplink connection

- out *certID1* - certificate ID to be used for mTLS authentication (optional)
- out *authServerToken1* - Authorization server configuration that could be used for access token authentication (optional)

8. ONVIF Client removes any existing uplink configurations by following the procedure mentioned in Annex A.19.
9. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.2 with the following input and output parameters
   - in *capSecurity* - security capabilities
   - in *keyAlgorithm* - key pair algorithm
   - in "CN=ONVIF TT Uplink 2,C=US" - CA certificate subject
   - in "Test CertPathValidationPolicy Uplink Alias2" - certification path validation policy alias
   - out *certPathValidationPolicyIDUplink2* - certification path validation policy identifier
   - out *certIDUplink2* - certificate identifier
   - out *keyIDUplink2* - key pair identifier
   - out *CACertUplink2* - CA certificate
   - out *privateKeyCACertUplink2* - CA certificate private key
10. ONVIF Client tries to create uplink configuration(s):
    10.1. ONVIF Client invokes **SetUplink** request with parameters
    - RemoteAddress := *uplinkAddress1*
    - CertificateID := *certID1*, if *certID1* was defined, otherwise field is skipped
    - UserLevel := **Administrator**
    - Status := Connected
    - CertPathValidationPolicyID := *certPathValidationPolicyIDUplink*
    - AuthorizationServer := *authServerToken1*, if *authServerToken1* was defined, otherwise field is skipped
    - Error is skipped

10.2. The DUT responds with **SetUplinkResponse** message.

10.3. ONVIF Client invokes **GetUplinks**.

10.4. The DUT responds with a **GetUplinksResponse** message with parameters:

- Configuration list =: *configurationList1*

10.5. If *configurationList1* does not contain exactly one configuration, FAIL the test, restore the DUT state, and skip other steps.

10.6. If *configurationList1* does not contains item with *uplinkAddress1*, FAIL the test, restore the DUT state, and skip other steps.

10.7. If *configurationList1* item with RemoteAddress = *uplinkAddress1* has fields values other that was set at step 10.1, FAIL the test, restore the DUT state, and skip other steps.

10.8. If the DUT supports more then one uplink connection as indicated by *capUplink*.MaxUplinks > 1:

10.8.1 ONVIF Client invokes **SetUplink** request with parameters

- RemoteAddress := *uplinkAddress2* (any valid address other then *uplinkAddress1*)
- CertificateID := *certID1*, if *certID1* was defined, otherwise field is skipped
- UserLevel := **Administrator**
- Status is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDUplink*
- AuthorizationServer := *authServerToken1*, if *authServerToken1* was defined, otherwise field is skipped
- Error is skipped

10.8.2 The DUT responds with **SetUplinkResponse** message.

10.8.3 ONVIF Client invokes **GetUplinks**.

10.8.4 The DUT responds with a **GetUplinksResponse** message with parameters:

- Configuration list =: *configurationList2*

10.8.5 If *configurationList2* does not contain exactly two configurations, FAIL the test, restore the DUT state, and skip other steps.

10.8.6 If *configurationList2* does not contains item with *uplinkAddress1*, FAIL the test, restore the DUT state, and skip other steps.

10.8.7 If *configurationList2* does not contains item with *uplinkAddress2*, FAIL the test, restore the DUT state, and skip other steps.

10.8.8 If *configurationList1* item with RemoteAddress = *uplinkAddress2* has fields values other that was set at step 10.8.1, FAIL the test, restore the DUT state, and skip other steps.

11. ONVIF Client tries to update uplink configuration:

11.1. ONVIF Client invokes **SetUplink** request with parameters

- RemoteAddress := *uplinkAddress1*
- CertificateID := *certID1*, if *certID1* was defined, otherwise field is skipped
- UserLevel := **Administrator**
- Status is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDUplink2*
- AuthorizationServer := *authServerToken1*, if *authServerToken1* was defined, otherwise field is skipped
- Error is skipped

11.2. The DUT responds with **SetUplinkResponse** message.

11.3. ONVIF Client invokes **GetUplinks**.

11.4. The DUT responds with a **GetUplinksResponse** message with parameters:

- Configuration list =: *configurationList3*

11.5. If *configurationList3* does not contain exactly number of configurations created at the steps 10.1 and 10.8.1 (if second step was executed), FAIL the test, restore the DUT state, and skip other steps.

11.6. If *configurationList3* does not contains item with *uplinkAddress1*, FAIL the test, restore the DUT state, and skip other steps.

11.7. If *configurationList3* does not contains item with *uplinkAddress2* (if it was created before), FAIL the test, restore the DUT state, and skip other steps.

11.8. If *configurationList3* item with RemoteAddress = *uplinkAddress1* has fields values other that was set at step 11.1, FAIL the test, restore the DUT state, and skip other steps.

12. ONVIF Client tries to delete uplink configuration:

12.1. ONVIF Client invokes **DeleteUplink** request with parameters

- RemoteAddress := *uplinkAddress1*

12.2. The DUT responds with **DeleteUplinkResponse** message.

12.3. ONVIF Client invokes **GetUplinks**.

12.4. The DUT responds with a **GetUplinksResponse** message with parameters:

- Configuration list =: *configurationList4*

12.5. If *configurationList4* does contains item with *uplinkAddress1*, FAIL the test, restore the DUT state, and skip other steps.

12.6. If *configurationList4* does not contains item with *uplinkAddress2* (if it was created before), FAIL the test, restore the DUT state, and skip other steps.

13. ONVIF Client restores the DUT state.

Test Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **SetUplinkResponse** message(s).
- DUT did not send **GetUplinkResponse** message(s).
- DUT did not send **DeleteUplinkResponse** message.

Note: The following fields are compared at steps 10.7, 10.8.8, 11.8:

- CertificateID

- UserLevel
- CertPathValidationPolicyID
- AuthorizationServer

5.2.2 Uplink Configuration Creation - Ambiguous Authentication

Test Case ID: UPLINK-2-1-2

Specification Coverage: SetUplink (ONVIF Uplink Specification)

Feature Under Test: Uplink Configuration Creation.

WSDL Reference: uplink.wsdl, advancedsecurity.wsdl

Test Purpose: To verify that a DUT rejects a configuration containing both CertificateID and AuthorizationServer.

Pre-Requisite: Security Configuration Service is received from the DUT. Uplink Service is received from the DUT. mTLS authentication is supported by the DUT as indicated by the AuthorizationModes = mTLS capability. Access token authentication is supported by the DUT as indicated by the AuthorizationModes = AccessToken capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the uplink service capabilities by following the procedure mentioned in Annex A.10 with the following input and output parameters:
   - out *capUplink* - Uplink Service Capabilities
4. Set:
   - in *uplinkAddress1* := any valid address with wss schema if *capUplink*.Protocols contains 'wss', otherwise with https schema to be used.
5. ONVIF Client configures security primitives needed for uplink connection and starts required servers by following the procedure described in Annex A.1 with the following input and output parameters:

- in *capUplink* - uplink capabilities
- in **true** - generate security primitives for both authentication types
- out *certPathValidationPolicyIDUplink1* - certification path validation policy for uplink connection
- out *certID1* - certificate ID to be used for mTLS authentication (optional)
- out *authServerToken1* - Authorization server configuration that could be used for access token authentication (optional)
- out *keyAlgorithm* - key pair algorithm

6. ONVIF Client removes any existing uplink configurations by following the procedure mentioned in Annex A.19.
7. ONVIF Client invokes **SetUplink** request with parameters
   - RemoteAddress := *uplinkAddress1*
   - CertificateID := *certID1*
   - UserLevel := **Administrator**
   - Status skipped
   - CertPathValidationPolicyID := *certPathValidationPolicyIDUplink*
   - AuthorizationServer := *authServerToken1*
   - Error is skipped
8. DUT send **env:Sender/ter:InvalidArgs/ter:AmbiguousAuthentication** SOAP 1.2 fault message.

Test Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **env:Sender/ter:InvalidArgs/ter:AmbiguousAuthentication** SOAP 1.2 fault message.

5.3 Authentication and Authorization

5.3.1 Uplink over WebSocket with mTLS Authentication

Test Case ID: UPLINK-3-1-1

Specification Coverage: mTLS authentication (ONVIF Uplink Specification)

Feature Under Test: Uplink WebSocket Connection using mTLS authentication.

WSDL Reference: uplink.wsdl, advancedsecurity.wsdl

Test Purpose: To verify that DUT can establish connection to ONVIF Client via Uplink over WebSocket using mTLS authentication. To verify that DUT do not require additional authentication inside established Uplink connection.

Pre-Requisite:

- Security Configuration Service is received from the DUT.
- Uplink Service is received from the DUT.
- mTLS authentication is supported by the DUT as indicated by the AuthorizationModes = mTLS capability.
- Uplink WebSocket connection is supported by the DUT as indicated by the Protocols = wss capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the security configuration service capabilities by following the procedure mentioned in Annex A.8 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client selects key pair algorithm by following the procedure mentioned in Annex A.17 with the following input and output parameters:
   - in *cap* - security configuration service capabilities

- out *keyAlgorithm* - key pair algorithm

5. ONVIF Client configures uplink connection using the following steps:

5.1. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.2 with the following input and output parameters

- in *cap* - DUT capabilities
- in *keyAlgorithm* - key pair algorithm
- in "CN=ONVIF TT Uplink 1,C=US" - CA certificate subject
- in "Test CertPathValidationPolicy Uplink Alias" - certification path validation policy alias
- out *certPathValidationPolicyIDUplink* - certification path validation policy identifier
- out *certIDUplink* - certificate identifier
- out *keyIDUplink* - key pair identifier
- out *CACertUplink* - CA certificate
- out *privateKeyCACertUplink* - CA certificate private key

5.2. 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.9 with the following input and output parameters:

- in *cap* - DUT capabilities
- in "CN=ONVIF TT Uplink 2,C=US" - certificate subject
- in *CACertUplink* - CA-certificate
- in *privateKeyCACertUplink* - private key of CA-certificate for certificate signature
- out *certUplink* - certificate
- out *publicKeyUplink* - public key of certificate
- out *privateKeyUplink* - private key of certificate

5.3. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.3 with the following input and output parameters:

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

5.4. ONVIF Client creates and uploads a CA-signed certificate for key pair based on associated CA certificate and a corresponding private key by following the procedure described in Annex A.11 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 identificator
- out *keyID1* - key pair
- out *cert1* - CA-signed certificate

5.5. ONVIF Client creates and uploads a CA-signed certificate for key pair based on associated CA certificate and a corresponding private key by following the procedure described in Annex A.11 with the following input and output parameters:

- in *cap* - DUT capabilities
- in *CAcert* - CA certificate
- in *CAkeyPair*.privateKey - CA certificate private key
- out *certID2* - CA-signed certificate identificator
- out *keyID2* - key pair
- out *cert2* - CA-signed certificate

5.6. ONVIF Client configure endpoint for WebSocket uplink connection at address *wssUplinkAddress1* using *certUplink* certificate with the following certificates accepted from the client:

- *cert1*
- *cert2*

5.7. ONVIF Client invokes **SetUplink** request with parameters

- RemoteAddress := *wssUplinkAddress1*
- CertificateID := *certID1*
- UserLevel := **Administrator**
- Status is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDUplink*
- AuthorizationServer is skipped
- Error is skipped

5.8. The DUT responds with **SetUplinkResponse** message.

6. ONVIF Client verifies initial connection establishment:

6.1. ONVIF Client awaits device connecting to *wssUplinkAddress1* endpoint.

6.2. DUT opens connection to *wssUplinkAddress1* with *cert1* used for client authentication.

6.3. DUT verifies certificate provided by ONVIF Client at uplink connection establishment based on *certPathValidationPolicyIDUplink* certification path validation policy.

6.4. ONVIF Client verify certificate received from the DUT. If received certificate is not *cert1*, FAIL the test, restore the DUT state, and skip other steps.

7. ONVIF Client verifies connection establishment after connection parameters were changed:

7.1. ONVIF Client invokes **SetUplink** request over uplink connection with parameters without any additional authentication with parameters

- RemoteAddress := *wssUplinkAddress1*
- CertificateID := *certID2*
- UserLevel := **Administrator**
- Status is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDUplink*
- AuthorizationServer is skipped
- Error is skipped

7.2. The DUT responds with **SetUplinkResponse** message.

7.3. ONVIF Client close device connection to *wssUplinkAddress1* endpoint.

7.4. ONVIF Client awaits device connecting to *wssUplinkAddress1* endpoint.

7.5. DUT opens connection to *wssUplinkAddress1* with *cert2* used for client authentication.

7.6. DUT verifies certificate provided by ONVIF Client at uplink connection establishment based on *certPathValidationPolicyIDUplink* certification path validation policy.

7.7. ONVIF Client verify certificate received from the DUT. If received certificate is not *cert2*, FAIL the test, restore the DUT state, and skip other steps.

8. ONVIF Client restores the DUT state.

Test Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **SetUplinkResponse** message.
- DUT does not establish WebSocket Uplink connection at step 6.2.
- DUT requests additional authentication at steps 7.1 and 7.2.
- DUT does not establish WebSocket Uplink connection at step 7.5.

5.3.2 Uplink over WebSocket with access token authentication (OAuthClientCredentials, client_secret_basic)

Test Case ID: UPLINK-3-1-2

Specification Coverage: Device authentication and authorization (ONVIF Security Service Specification), Access token authentication (ONVIF Uplink Specification)

Feature Under Test: Uplink Connection using access token authentication using OAuthClientCredentials and client_secret_basic settings with authentication server certificate validation.

WSDL Reference: uplink.wsdl, advancedsecurity.wsdl

Test Purpose:

- To verify that DUT can receive access token from the authentication server using OAuthClientCredentials authentication method with client_secret_basic type.
- To verify that DUT can establish connection to ONVIF Client via Uplink using access token.
- To verify that DUT will reuse token if it is not expired.
- To verify that DUT will renew token when authentication parameters will be changed.
- To verify that DUT do not require additional authentication inside established Uplink connection.

Pre-Requisite:

- Security Configuration Service is received from the DUT.
- Uplink Service is received from the DUT.
- Authorization Server Configuration is supported by the DUT as indicated by the AuthorizationServer.MaxConfigurations capability.
- OAuthClientCredentials authentication method is supported by the DUT as indicated by the AuthorizationServer.ConfigurationTypesSupported capability.
- client_secret_basic authentication is supported by the DUT as indicated by the AuthorizationServer.ClientAuthenticationMethodsSupported capability.
- Access token authentication is supported by the DUT as indicated by the AuthorizationModes = AccessToken capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the security configuration service capabilities by following the procedure mentioned in Annex A.8 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client deletes one authorization server configuration if maximum is reached by following the procedure described in Annex A.20 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - out *itemToRestore1* - deleted authorization server configuration if any

5. ONVIF Client selects key pair algorithm by following the procedure mentioned in Annex A.17 with the following input and output parameters:
   - in *cap* - security configuration service capabilities
   - out *keyAlgorithm* - key pair algorithm
6. ONVIF Client configures authentication server connection using the following steps:
   6.1. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.2 with the following input and output parameters
      - in *cap* - DUT capabilities
      - in *keyAlgorithm* - key pair algorithm
      - in "CN=ONVIF TT AuthServer 1,C=US" - CA certificate subject
      - in "Test CertPathValidationPolicy AuthServer Alias" - certification path validation policy alias
      - out *certPathValidationPolicyIDAuthServer* - certification path validation policy identifier
      - out *certIDAuthServer* - certificate identifier
      - out *keyIDAuthServer* - key pair identifier
      - out *CACertAuthServer* - CA certificate
      - out *privateKeyCACertAuthServer* - CA certificate private key
   6.2. 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.9 with the following input and output parameters:
      - in *cap* - DUT capabilities
      - in "CN=ONVIF TT AuthServer 2,C=US" - certificate subject
      - in *CACertAuthServer* - CA-certificate
      - in *privateKeyCACertAuthServer* - private key of CA-certificate for certificate signature
      - out *certAuthServer* - certificate

- out *publicKeyAuthServer* - public key of certificate
- out *privateKeyAuthServer* - private key of certificate

6.3. ONVIF Client starts Authorization server with metadata endpoint *authServerMetadataEndpoint1* conforming to RFC8414 with following settings:

- It is configured to *certAuthServer* as a server certificate.
- It is configured to accept OAuth2 client credentials grant flow per [RFC 6749] authentication method only.
- It is configured to accept client_secret_basic authentication method only.
- Client with client identifier *clientID1* with client secret *clientSecret1* and scope *scope1* is added to be authorized.
- Client with client identifier *clientID1* with client secret *clientSecret1* and scope *scope2* is added to be authorized.

6.4. ONVIF Client invokes **CreateAuthorizationServerConfiguration** request with parameters

- Type := **"OAuthClientCredentials"**
- ClientAuth := **"client_secret_basic"**
- ServerUri := *authServerMetadataEndpoint1*
- ClientID := *clientID1*
- ClientSecret := *clientSecret1*
- Scope := *scope1*
- KeyID is skipped
- CertificateID is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDAuthServer*

6.5. The DUT responds with **CreateAuthorizationServerConfigurationResponse** message with parameters

- Token =: *authServerToken1*

7. ONVIF Client configures uplink connection using the following steps:

7.1. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.2 with the following input and output parameters

- in *cap* - DUT capabilities
- in *keyAlgorithm* - key pair algorithm
- in "CN=ONVIF TT Uplink 1,C=US" - CA certificate subject
- in "Test CertPathValidationPolicy Uplink Alias" - certification path validation policy alias
- out *certPathValidationPolicyIDUplink* - certification path validation policy identifier
- out *certIDUplink* - certificate identifier
- out *keyIDUplink* - key pair identifier
- out *CACertUplink* - CA certificate
- out *privateKeyCACertUplink* - CA certificate private key

7.2. 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.9 with the following input and output parameters:

- in *cap* - DUT capabilities
- in "CN=ONVIF TT Uplink 2,C=US" - certificate subject
- in *CACertUplink* - CA-certificate
- in *privateKeyCACertUplink* - private key of CA-certificate for certificate signature
- out *certUplink* - certificate
- out *publicKeyUplink* - public key of certificate
- out *privateKeyUplink* - private key of certificate

7.3. ONVIF Client configure and start service endpoint for uplink connection at address *uplinkAddress1* using *certUplink* certificate.

7.4. ONVIF Client invokes **SetUplink** request with parameters

- RemoteAddress := *uplinkAddress1*

- CertificateID is skipped
- UserLevel := **Administrator**
- Status is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDUplink*
- AuthorizationServer := *authServerToken1*
- Error is skipped

7.5. The DUT responds with **SetUplinkResponse** message.

8. ONVIF Client verifies initial connection establishment:

8.1. The DUT opens connection to *authServerMetadataEndpoint1* with client identifier *clientID1* and client secret *clientSecret1* using client_secret_basic method authentication and OAuth2 client credentials grant flow.

8.2. The DUT verifies certificate provided by authentication server based on *certPathValidationPolicyIDAuthServer* certification path validation policy.

8.3. The DUT receives access token *accessToken1* from authorization server for scope *scope1*.

8.4. ONVIF Client awaits device connecting to *uplinkAddress1* endpoint.

8.5. DUT opens connection to *uplinkAddress1* with *accessToken1*.

8.6. DUT verifies certificate provided by ONVIF Client at uplink connection establishment based on *certPathValidationPolicyIDUplink* certification path validation policy.

8.7. ONVIF Client verify access token received from the DUT with Authentication Server. If received access token *accessToken1* is not valid for *scope1*, FAIL the test, restore the DUT state, and skip other steps.

9. ONVIF Client verifies that access token will be reused if it is not expired:

9.1. ONVIF Client close device connection to *uplinkAddress1* endpoint.

9.2. ONVIF Client awaits device connecting to *uplinkAddress1* endpoint.

9.3. DUT opens connection to *uplinkAddress1* with *accessToken1*.

9.4. DUT verifies certificate provided by ONVIF Client at uplink connection establishment based on *certPathValidationPolicyIDUplink* certification path validation policy.

9.5. ONVIF Client verify access token received from the DUT with Authentication Server. If received access token *accessToken1* is not the same as was used at step 8.5, FAIL the test, restore the DUT state, and skip other steps.

10. ONVIF Client verifies connection establishment after connection parameters were changed:

10.1. ONVIF Client invokes **SetAuthorizationServerConfiguration** request using uplink connection with parameters without any additional authentication

- @token := *authServerToken1*
- Data.Type := **"OAuthClientCredentials"**
- Data.ClientAuth := **"client_secret_basic"**
- Data.ServerUri := *authServerMetadataEndpoint1*
- Data.ClientID := *clientID1*
- Data.ClientSecret := *clientSecret1*
- Data.Scope := *scope2*
- Data.KeyID is skipped
- Data.CertificateID is skipped
- Data.CertPathValidationPolicyID := *certPathValidationPolicyIDAuthServer*

10.2. The DUT responds with **SetAuthorizationServerConfigurationResponse** message.

10.3. ONVIF Client close device connection to *uplinkAddress1* endpoint.

10.4. The DUT opens connection to *authServerMetadataEndpoint1* with client identifier *clientID1* and client secret *clientSecret1* using client_secret_basic method authentication and OAuth2 client credentials grant flow.

10.5. The DUT receives access token *accessToken2* from authorization server for scope *scope2*.

10.6. ONVIF Client awaits device connecting to *uplinkAddress1* endpoint.

10.7. DUT opens connection to *uplinkAddress1* with *accessToken2*.

10.8. DUT verifies certificate provided by ONVIF Client at uplink connection establishment based on *certPathValidationPolicyIDUplink* certification path validation policy.

10.9. ONVIF Client verify access token received from the DUT with Authentication Server. If received access token *accessToken2* is not valid for *scope2*, FAIL the test, restore the DUT state, and skip other steps.

11. ONVIF Client restores the DUT state.

Test Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **CreateAuthorizationServerConfigurationResponse** message.
- DUT did not send **SetUplinkResponse** message.
- DUT did not send **SetAuthorizationServerConfigurationResponse** message.
- DUT does not establish connection at step 8.5.
- DUT does not establish connection at step 9.3.
- DUT requests additional authentication at steps 10.1 and 10.2.
- DUT does not establish connection at step 10.7.

5.3.3 Uplink over WebSocket with access token authentication (OAuthClientCredentials, private_key_jwt, RS256)

Test Case ID: UPLINK-3-1-3

Specification Coverage: Device authentication and authorization (ONVIF Security Service Specification), Access token authentication (ONVIF Uplink Specification)

Feature Under Test: Uplink Connection using access token authentication using OAuthClientCredentials and private_key_jwt settings with authentication server certificate validation.

WSDL Reference: uplink.wsdl, advancedsecurity.wsdl

Test Purpose:

- To verify that DUT can receive access token from the authentication server using OAuthClientCredentials authentication method with private_key_jwt type.
- To verify that DUT can establish connection to ONVIF Client via Uplink using access token.
- To verify that DUT do not require additional authentication inside established Uplink connection.
- To verify that a DUT supports JWT authentication algorithm required by ONVIF Security Baseline (RS256).

Pre-Requisite:

- Security Configuration Service is received from the DUT.
- Uplink Service is received from the DUT.
- Authorization Server Configuration is supported by the DUT as indicated by the AuthorizationServer.MaxConfigurations capability.
- OAuthClientCredentials authentication method is supported by the DUT as indicated by the AuthorizationServer.ConfigurationTypesSupported capability.
- private_key_jwt authentication is supported by the DUT as indicated by the AuthorizationServer.ClientAuthenticationMethodsSupported capability.
- Access token authentication is supported by the DUT as indicated by the AuthorizationModes = AccessToken capability.
- RS256 is required to be supported by Security Baseline version required for the DUT as defined in section Private Key JWT Algorithms for Baseline Versions of [ONVIF Security Baseline Device Test Spec].

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the security configuration service capabilities by following the procedure mentioned in Annex A.8 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities

4. ONVIF Client deletes one authorization server configuration if maximum is reached by following the procedure described in Annex A.20 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - out *itemToRestore1* - deleted authorization server configuration if any
5. ONVIF Client selects key pair algorithm by following the procedure mentioned in Annex A.17 with the following input and output parameters:
   - in *cap* - security configuration service capabilities
   - out *keyAlgorithm* - key pair algorithm
6. ONVIF Client configures authentication server connection using the following steps:
   6.1. ONVIF Client creates a key pair and get public key for JWT authentication from device by following the procedure described in Annex A.18 with the following input and output parameters:
      - in **RS256** - JWT Algorithm
      - out *keyID1Auth1* - key pair
      - out *publicKeyAuth1* - public key
   6.2. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.2 with the following input and output parameters
      - in *cap* - DUT capabilities
      - in *keyAlgorithm* - CSR key pair algorithm
      - in "CN=ONVIF TT AuthServer 1,C=US" - CA certificate subject
      - in "Test CertPathValidationPolicy AuthServer Alias" - certification path validation policy alias
      - out *certPathValidationPolicyIDAuthServer* - certification path validation policy identifier
      - out *certIDAuthServer* - certificate identifier
      - out *keyIDAuthServer* - key pair identifier
      - out *CACertAuthServer* - CA certificate

- out *privateKeyCACertAuthServer* - CA certificate private key

6.3. 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.9 with the following input and output parameters:

- in *cap* - DUT capabilities
- in "CN=ONVIF TT AuthServer 2,C=US" - certificate subject
- in *CACertAuthServer* - CA-certificate
- in *privateKeyCACertAuthServer* - private key of CA-certificate for certificate signature
- out *certAuthServer* - certificate
- out *publicKeyAuthServer* - public key of certificate
- out *privateKeyAuthServer* - private key of certificate

6.4. ONVIF Client starts Authorization server with metadata endpoint *authServerMetadataEndpoint1* conforming to RFC8414 with following settings:

- It is configured to *certAuthServer* as a server certificate.
- It is configured to accept OAuth2 client credentials grant flow per [RFC 6749] authentication method only.
- It is configured to accept private_key_jwt authentication method with **RS256** algorithm only.
- Client with client identifier *clientID1* with client secret *clientSecret1*, *publicKeyAuth1* as key, and scope *scope1* is added to be authorized.
- Client with client identifier *clientID1* with client secret *clientSecret1*, *publicKeyAuth1* as key, and scope *scope2* is added to be authorized.

6.5. ONVIF Client invokes **CreateAuthorizationServerConfiguration** request with parameters

- Type := **"OAuthClientCredentials"**
- ClientAuth := **"private_key_jwt"**

- ServerUri := *authServerMetadataEndpoint1*
- ClientID := *clientID1*
- ClientSecret skipped
- Scope := *scope1*
- KeyID := *publicKeyAuth1*
- CertificateID is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDAuthServer*

6.6. The DUT responds with **CreateAuthorizationServerConfigurationResponse** message with parameters

- Token =: *authServerToken1*

7. ONVIF Client configures uplink connection using the following steps:

7.1. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.2 with the following input and output parameters

- in *cap* - DUT capabilities
- in *keyAlgorithm* - key pair algorithm
- in "CN=ONVIF TT Uplink 1,C=US" - CA certificate subject
- in "Test CertPathValidationPolicy Uplink Alias" - certification path validation policy alias
- out *certPathValidationPolicyIDUplink* - certification path validation policy identifier
- out *certIDUplink* - certificate identifier
- out *keyIDUplink* - key pair identifier
- out *CACertUplink* - CA certificate
- out *privateKeyCACertUplink* - CA certificate private key

7.2. 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.9 with the following input and output parameters:

- in *cap* - DUT capabilities
- in "CN=ONVIF TT Uplink 2,C=US" - certificate subject
- in *CACertUplink* - CA-certificate
- in *privateKeyCACertUplink* - private key of CA-certificate for certificate signature
- out *certUplink* - certificate
- out *publicKeyUplink* - public key of certificate
- out *privateKeyUplink* - private key of certificate

7.3. ONVIF Client configure and start service endpoint for uplink connection at address *uplinkAddress1* using *certUplink* certificate.

7.4. ONVIF Client invokes **SetUplink** request with parameters

- RemoteAddress := *uplinkAddress1*
- CertificateID is skipped
- UserLevel := **Administrator**
- Status is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDUplink*
- AuthorizationServer := *authServerToken1*
- Error is skipped

7.5. The DUT responds with **SetUplinkResponse** message.

8. ONVIF Client verifies initial connection establishment:

8.1. The DUT opens connection to *authServerMetadataEndpoint1* with client identifier *clientID1* and client secret *clientSecret1* using client_secret_basic method authentication and OAuth2 client credentials grant flow.

8.2. The DUT verifies certificate provided by authentication server based on *certPathValidationPolicyIDAuthServer* certification path validation policy.

8.3. The DUT receives access token *accessToken1* from authorization server for scope *scope1*.

8.4. ONVIF Client awaits device connecting to *uplinkAddress1* endpoint.

8.5. DUT opens connection to *uplinkAddress1* with *accessToken1*.

8.6. DUT verifies certificate provided by ONVIF Client at uplink connection establishment based on *certPathValidationPolicyIDUplink* certification path validation policy.

8.7. ONVIF Client verify access token received from the DUT with Authentication Server. If received access token *accessToken1* is not valid for *scope1*, FAIL the test, restore the DUT state, and skip other steps.

8.8. ONVIF Client invokes **SetAuthorizationServerConfiguration** request using uplink connection with parameters without any additional authentication

- @token := *authServerToken1*
- Data.Type := **"OAuthClientCredentials"**
- Data.ClientAuth := **"client_secret_basic"**
- Data.ServerUri := *authServerMetadataEndpoint1*
- Data.ClientID := *clientID1*
- Data.ClientSecret := *clientSecret1*
- Data.Scope := *scope2*
- Data.KeyID is skipped
- Data.CertificateID is skipped
- Data.CertPathValidationPolicyID := *certPathValidationPolicyIDAuthServer*

8.9. The DUT responds with **SetAuthorizationServerConfigurationResponse** message.

9. ONVIF Client restores the DUT state.

Test Result:

PASS –

- DUT passed all assertions.

FAIL –

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

- DUT did not send **SetUplinkResponse** message.
- DUT did not send **SetAuthorizationServerConfigurationResponse** message.
- DUT does not establish connection at step 8.5.

5.3.4 Uplink over WebSocket with mTLS Authentication - Invalid Server Certificate

Test Case ID: UPLINK-3-1-4

Specification Coverage: mTLS authentication (ONVIF Uplink Specification)

Feature Under Test: Uplink Connection using mTLS authentication.

WSDL Reference: uplink.wsdl, advancedsecurity.wsdl

Test Purpose: To verify that DUT will not establish connection with invalid server certificate.

Pre-Requisite:

- Security Configuration Service is received from the DUT.
- Uplink Service is received from the DUT.
- mTLS authentication is supported by the DUT as indicated by the AuthorizationModes = mTLS capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the security configuration service capabilities by following the procedure mentioned in Annex A.8 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client selects key pair algorithm by following the procedure mentioned in Annex A.17 with the following input and output parameters:
   - in *cap* - security configuration service capabilities

- out *keyAlgorithm* - key pair algorithm

5. ONVIF Client configures uplink connection using the following steps:

5.1. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.2 with the following input and output parameters

- in *cap* - DUT capabilities
- in *keyAlgorithm* - key pair algorithm
- in "CN=ONVIF TT Uplink 1,C=US" - CA certificate subject
- in "Test CertPathValidationPolicy Uplink Alias" - certification path validation policy alias
- out *certPathValidationPolicyIDUplink* - certification path validation policy identifier
- out *certIDUplink* - certificate identifier
- out *keyIDUplink* - key pair identifier
- out *CACertUplink* - CA certificate
- out *privateKeyCACertUplink* - CA certificate private key

5.2. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.3 with the following input and output parameters:

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

5.3. ONVIF Client creates and uploads a CA-signed certificate for key pair based on associated CA certificate and a corresponding private key by following the procedure described in Annex A.11 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 identificator
- out *keyID1* - key pair
- out *cert1* - CA-signed certificate

5.4. ONVIF Client configure endpoint for uplink connection at address *uplinkAddress1* using any certificate that will not pass *certPathValidationPolicyIDUplink* with the following certificates accepted from the client:

- *cert1*

5.5. ONVIF Client invokes **SetUplink** request with parameters

- RemoteAddress := *uplinkAddress1*
- CertificateID := *certID1*
- UserLevel := **Administrator**
- Status is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDUplink*
- AuthorizationServer is skipped
- Error is skipped

5.6. The DUT responds with **SetUplinkResponse** message.

6. ONVIF Client verifies connection will not be establishment:

6.1. ONVIF Client awaits device connecting to *uplinkAddress1* endpoint.

6.2. DUT tries to open connection to *uplinkAddress1* with *cert1* used for client authentication.

6.3. DUT verifies certificate provided by ONVIF Client at uplink connection establishment based on *certPathValidationPolicyIDUplink* certification path validation policy and prevent connection.

7. ONVIF Client restores the DUT state.

Test Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **SetUplinkResponse** message.
- DUT establishes connection at step 6.3 during *operationDelay* timeout.

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

5.3.5 Uplink over WebSocket with access token authentication (OAuthClientCredentials, client_secret_basic) - Invalid Uplink Client Certificate

Test Case ID: UPLINK-3-1-5

Specification Coverage: Device authentication and authorization (ONVIF Security Service Specification), Access token authentication (ONVIF Uplink Specification)

Feature Under Test: Uplink Connection using access token authentication when token received from authentication server using OAuthClientCredentials and client_secret_basic settings with authentication server certificate validation.

WSDL Reference: uplink.wsdl, advancedsecurity.wsdl

Test Purpose: To verify that DUT will not establish connection with uplink client which has invalid certificate.

Pre-Requisite:

- Security Configuration Service is received from the DUT.
- Uplink Service is received from the DUT.
- Authorization Server Configuration is supported by the DUT as indicated by the AuthorizationServer.MaxConfigurations capability.
- OAuthClientCredentials authentication method is supported by the DUT as indicated by the AuthorizationServer.ConfigurationTypesSupported capability.
- client_secret_basic authentication is supported by the DUT as indicated by the AuthorizationServer.ClientAuthenticationMethodsSupported capability.
- Access token authentication is supported by the DUT as indicated by the AuthorizationModes = AccessToken capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the security configuration service capabilities by following the procedure mentioned in Annex A.8 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client deletes one authorization server configuration if maximum is reached by following the procedure described in Annex A.20 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - out *itemToRestore1* - deleted authorization server configuration if any
5. ONVIF Client selects key pair algorithm by following the procedure mentioned in Annex A.17 with the following input and output parameters:
   - in *cap* - security configuration service capabilities
   - out *keyAlgorithm* - key pair algorithm
6. ONVIF Client configures authentication server connection using the following steps:
   6.1. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.2 with the following input and output parameters
      - in *cap* - DUT capabilities
      - in *keyAlgorithm* - key pair algorithm
      - in "CN=ONVIF TT AuthServer 1,C=US" - CA certificate subject
      - in "Test CertPathValidationPolicy AuthServer Alias" - certification path validation policy alias
      - out *certPathValidationPolicyIDAuthServer* - certification path validation policy identifier
      - out *certIDAuthServer* - certificate identifier
      - out *keyIDAuthServer* - key pair identifier
      - out *CACertAuthServer* - CA certificate
      - out *privateKeyCACertAuthServer* - CA certificate private key

6.2. 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.9 with the following input and output parameters:

- in *cap* - DUT capabilities
- in "CN=ONVIF TT AuthServer 2,C=US" - certificate subject
- in *CACertAuthServer* - CA-certificate
- in *privateKeyCACertAuthServer* - private key of CA-certificate for certificate signature
- out *certAuthServer* - certificate
- out *publicKeyAuthServer* - public key of certificate
- out *privateKeyAuthServer* - private key of certificate

6.3. ONVIF Client starts Authorization server with metadata endpoint *authServerMetadataEndpoint1* conforming to RFC8414 with following settings:

- It is configured to *certAuthServer* as a server certificate.
- It is configured to accept OAuth2 client credentials grant flow per [RFC 6749] authentication method only.
- It is configured to accept client_secret_basic authentication method only.
- Client with client identifier *clientID1* with client secret *clientSecret1* and scope *scope1* is added to be authorized.
- Client with client identifier *clientID1* with client secret *clientSecret1* and scope *scope2* is added to be authorized.

6.4. ONVIF Client invokes **CreateAuthorizationServerConfiguration** request with parameters

- Type := **"OAuthClientCredentials"**
- ClientAuth := **"client_secret_basic"**
- ServerUri := *authServerMetadataEndpoint1*
- ClientID := *clientID1*

- ClientSecret := *clientSecret1*
- Scope := *scope1*
- KeyID is skipped
- CertificateID is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDAuthServer*

6.5. The DUT responds with **CreateAuthorizationServerConfigurationResponse** message with parameters

- Token =: *authServerToken1*

7. ONVIF Client configures uplink connection using the following steps:

7.1. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.2 with the following input and output parameters

- in *cap* - DUT capabilities
- in *keyAlgorithm* - key pair algorithm
- in "CN=ONVIF TT Uplink 1,C=US" - CA certificate subject
- in "Test CertPathValidationPolicy Uplink Alias" - certification path validation policy alias
- out *certPathValidationPolicyIDUplink* - certification path validation policy identifier
- out *certIDUplink* - certificate identifier
- out *keyIDUplink* - key pair identifier
- out *CACertUplink* - CA certificate
- out *privateKeyCACertUplink* - CA certificate private key

7.2. ONVIF Client configure and start service endpoint for uplink connection at address *uplinkAddress1* using any certificate that will **not** pass *certPathValidationPolicyIDUplink* certification path validation policy.

7.3. ONVIF Client invokes **SetUplink** request with parameters

- RemoteAddress := *uplinkAddress1*

- CertificateID is skipped
- UserLevel := **Administrator**
- Status is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDUplink*
- AuthorizationServer := *authServerToken1*
- Error is skipped

7.4. The DUT responds with **SetUplinkResponse** message.

8. ONVIF Client verifies initial connection establishment:

8.1. The DUT opens connection to *authServerMetadataEndpoint1* with client identifier *clientID1* and client secret *clientSecret1* using client_secret_basic method authentication and OAuth2 client credentials grant flow.

8.2. The DUT verifies certificate provided by authentication server based on *certPathValidationPolicyIDAuthServer* certification path validation policy.

8.3. The DUT receives access token *accessToken1* from authorization server for scope *scope1*.

8.4. ONVIF Client awaits device connecting to *uplinkAddress1* endpoint.

8.5. DUT opens connection to *uplinkAddress1* with *accessToken1*.

8.6. DUT verifies certificate provided by ONVIF Client at uplink connection establishment based on *certPathValidationPolicyIDUplink* certification path validation policy and prevent connection.

9. ONVIF Client restores the DUT state.

Test Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **CreateAuthorizationServerConfigurationResponse** message.
- DUT did not send **SetUplinkResponse** message.

- DUT establishes connection at step 8.6 during *operationDelay* timeout.

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

5.3.6 Uplink over WebSocket with access token authentication (OAuthClientCredentials, client_secret_basic) - Invalid Authentication Server Certificate

Test Case ID: UPLINK-3-1-6

Specification Coverage: Device authentication and authorization (ONVIF Security Service Specification), Access token authentication (ONVIF Uplink Specification)

Feature Under Test: Uplink Connection using access token authentication when token received from authentication server using OAuthClientCredentials and client_secret_basic settings with authentication server certificate validation.

WSDL Reference: uplink.wsdl, advancedsecurity.wsdl

Test Purpose: To verify that DUT will not establish connection with invalid authentication server certificate.

Pre-Requisite:

- Security Configuration Service is received from the DUT.
- Uplink Service is received from the DUT.
- Authorization Server Configuration is supported by the DUT as indicated by the AuthorizationServer.MaxConfigurations capability.
- OAuthClientCredentials authentication method is supported by the DUT as indicated by the AuthorizationServer.ConfigurationTypesSupported capability.
- client_secret_basic authentication is supported by the DUT as indicated by the AuthorizationServer.ClientAuthenticationMethodsSupported capability.
- Access token authentication is supported by the DUT as indicated by the AuthorizationModes = AccessToken capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

3. ONVIF Client gets the security configuration service capabilities by following the procedure mentioned in Annex A.8 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client deletes one authorization server configuration if maximum is reached by following the procedure described in Annex A.20 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - out *itemToRestore1* - deleted authorization server configuration if any
5. ONVIF Client selects key pair algorithm by following the procedure mentioned in Annex A.17 with the following input and output parameters:
   - in *cap* - security configuration service capabilities
   - out *keyAlgorithm* - key pair algorithm
6. ONVIF Client configures authentication server connection using the following steps:
   6.1. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.2 with the following input and output parameters
      - in *cap* - DUT capabilities
      - in *keyAlgorithm* - key pair algorithm
      - in "CN=ONVIF TT AuthServer 1,C=US" - CA certificate subject
      - in "Test CertPathValidationPolicy AuthServer Alias" - certification path validation policy alias
      - out *certPathValidationPolicyIDAuthServer* - certification path validation policy identifier
      - out *certIDAuthServer* - certificate identifier
      - out *keyIDAuthServer* - key pair identifier
      - out *CACertAuthServer* - CA certificate
      - out *privateKeyCACertAuthServer* - CA certificate private key

   6.2. ONVIF Client starts Authorization server with metadata endpoint *authServerMetadataEndpoint1* conforming to RFC8414 with following settings:

- It is configured to any certificate that will **not** pass *certPathValidationPolicyIDAuthServer* validation as a server certificate.
- It is configured to accept OAuth2 client credentials grant flow per [RFC 6749] authentication method only.
- It is configured to accept client_secret_basic authentication method only.
- Client with client identifier *clientID1* with client secret *clientSecret1* and scope *scope1* is added to be authorized.
- Client with client identifier *clientID1* with client secret *clientSecret1* and scope *scope2* is added to be authorized.

6.3. ONVIF Client invokes **CreateAuthorizationServerConfiguration** request with parameters

- Type := **"OAuthClientCredentials"**
- ClientAuth := **"client_secret_basic"**
- ServerUri := *authServerMetadataEndpoint1*
- ClientID := *clientID1*
- ClientSecret := *clientSecret1*
- Scope := *scope1*
- KeyID is skipped
- CertificateID is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDAuthServer*

6.4. The DUT responds with **CreateAuthorizationServerConfigurationResponse** message with parameters

- Token =: *authServerToken1*

7. ONVIF Client configures uplink connection using the following steps:

7.1. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.2 with the following input and output parameters

- in *cap* - DUT capabilities

- in *keyAlgorithm* - key pair algorithm
- in "CN=ONVIF TT Uplink 1,C=US" - CA certificate subject
- in "Test CertPathValidationPolicy Uplink Alias" - certification path validation policy alias
- out *certPathValidationPolicyIDUplink* - certification path validation policy identifier
- out *certIDUplink* - certificate identifier
- out *keyIDUplink* - key pair identifier
- out *CACertUplink* - CA certificate
- out *privateKeyCACertUplink* - CA certificate private key

7.2. 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.9 with the following input and output parameters:

- in *cap* - DUT capabilities
- in "CN=ONVIF TT Uplink 2,C=US" - certificate subject
- in *CACertUplink* - CA-certificate
- in *privateKeyCACertUplink* - private key of CA-certificate for certificate signature
- out *certUplink* - certificate
- out *publicKeyUplink* - public key of certificate
- out *privateKeyUplink* - private key of certificate

7.3. ONVIF Client configure and start service endpoint for uplink connection at address *uplinkAddress1* using *certUplink* certificate.

7.4. ONVIF Client invokes **SetUplink** request with parameters

- RemoteAddress := *uplinkAddress1*
- CertificateID is skipped
- UserLevel := **Administrator**

- Status is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDUplink*
- AuthorizationServer := *authServerToken1*
- Error is skipped

7.5. The DUT responds with **SetUplinkResponse** message.

8. ONVIF Client verifies initial connection establishment:

8.1. The DUT opens connection to *authServerMetadataEndpoint1* with client identifier *clientID1* and client secret *clientSecret1* using client_secret_basic method authentication and OAuth2 client credentials grant flow.

8.2. The DUT verifies certificate provided by authentication server based on *certPathValidationPolicyIDAuthServer* certification path validation policy and prevent connection.

9. ONVIF Client restores the DUT state.

Test Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **CreateAuthorizationServerConfigurationResponse** message.
- DUT did not send **SetUplinkResponse** message.
- DUT establishes connection at step 8.2 during *operationDelay* timeout.

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

5.3.7 Uplink over WebSocket with access token authentication (OAuthClientCredentials, private_key_jwt, ES256)

Test Case ID: UPLINK-3-1-7

Specification Coverage: Device authentication and authorization (ONVIF Security Service Specification), Access token authentication (ONVIF Uplink Specification)

Feature Under Test: Uplink Connection using access token authentication using OAuthClientCredentials and private_key_jwt settings with authentication server certificate validation.

WSDL Reference: uplink.wsdl, advancedsecurity.wsdl

Test Purpose:

- To verify that DUT can receive access token from the authentication server using OAuthClientCredentials authentication method with private_key_jwt type.
- To verify that DUT can establish connection to ONVIF Client via Uplink using access token.
- To verify that DUT do not require additional authentication inside established Uplink connection.
- To verify that a DUT supports JWT authentication algorithm required by ONVIF Security Baseline (ES256).

Pre-Requisite:

- Security Configuration Service is received from the DUT.
- Uplink Service is received from the DUT.
- Authorization Server Configuration is supported by the DUT as indicated by the AuthorizationServer.MaxConfigurations capability.
- OAuthClientCredentials authentication method is supported by the DUT as indicated by the AuthorizationServer.ConfigurationTypesSupported capability.
- private_key_jwt authentication is supported by the DUT as indicated by the AuthorizationServer.ClientAuthenticationMethodsSupported capability.
- Access token authentication is supported by the DUT as indicated by the AuthorizationModes = AccessToken capability.
- ES256 is required to be supported by Security Baseline version required for the DUT as defined in section Private Key JWT Algorithms for Baseline Versions of [ONVIF Security Baseline Device Test Spec].

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

3. ONVIF Client gets the security configuration service capabilities by following the procedure mentioned in Annex A.8 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
4. ONVIF Client deletes one authorization server configuration if maximum is reached by following the procedure described in Annex A.20 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - out *itemToRestore1* - deleted authorization server configuration if any
5. ONVIF Client selects key pair algorithm by following the procedure mentioned in Annex A.17 with the following input and output parameters:
   - in *cap* - security configuration service capabilities
   - out *keyAlgorithm* - key pair algorithm
6. ONVIF Client configures authentication server connection using the following steps:
   6.1. ONVIF Client creates a key pair and get public key for JWT authentication from device by following the procedure described in Annex A.18 with the following input and output parameters:
      - in **ES256** - JWT Algorithm
      - out *keyID1Auth1* - key pair
      - out *publicKeyAuth1* - public key
   6.2. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.2 with the following input and output parameters
      - in *cap* - DUT capabilities
      - in *keyAlgorithm* - CSR key pair algorithm
      - in "CN=ONVIF TT AuthServer 1,C=US" - CA certificate subject
      - in "Test CertPathValidationPolicy AuthServer Alias" - certification path validation policy alias
      - out *certPathValidationPolicyIDAuthServer* - certification path validation policy identifier

- out *certIDAuthServer* - certificate identifier
- out *keyIDAuthServer* - key pair identifier
- out *CACertAuthServer* - CA certificate
- out *privateKeyCACertAuthServer* - CA certificate private key

6.3. 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.9 with the following input and output parameters:

- in *cap* - DUT capabilities
- in "CN=ONVIF TT AuthServer 2,C=US" - certificate subject
- in *CACertAuthServer* - CA-certificate
- in *privateKeyCACertAuthServer* - private key of CA-certificate for certificate signature
- out *certAuthServer* - certificate
- out *publicKeyAuthServer* - public key of certificate
- out *privateKeyAuthServer* - private key of certificate

6.4. ONVIF Client starts Authorization server with metadata endpoint *authServerMetadataEndpoint1* conforming to RFC8414 with following settings:

- It is configured to *certAuthServer* as a server certificate.
- It is configured to accept OAuth2 client credentials grant flow per [RFC 6749] authentication method only.
- It is configured to accept private_key_jwt authentication method with **ES256** algorithm only.
- Client with client identifier *clientID1* with client secret *clientSecret1*, *publicKeyAuth1* as key, and scope *scope1* is added to be authorized.
- Client with client identifier *clientID1* with client secret *clientSecret1*, *publicKeyAuth1* as key, and scope *scope2* is added to be authorized.

6.5. ONVIF Client invokes **CreateAuthorizationServerConfiguration** request with parameters

- Type := **"OAuthClientCredentials"**
- ClientAuth := **"private_key_jwt"**
- ServerUri := *authServerMetadataEndpoint1*
- ClientID := *clientID1*
- ClientSecret skipped
- Scope := *scope1*
- KeyID := *publicKeyAuth1*
- CertificateID is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDAuthServer*

6.6. The DUT responds with **CreateAuthorizationServerConfigurationResponse** message with parameters

- Token =: *authServerToken1*

7. ONVIF Client configures uplink connection using the following steps:

7.1. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.2 with the following input and output parameters

- in *cap* - DUT capabilities
- in *keyAlgorithm* - key pair algorithm
- in "CN=ONVIF TT Uplink 1,C=US" - CA certificate subject
- in "Test CertPathValidationPolicy Uplink Alias" - certification path validation policy alias
- out *certPathValidationPolicyIDUplink* - certification path validation policy identifier
- out *certIDUplink* - certificate identifier
- out *keyIDUplink* - key pair identifier
- out *CACertUplink* - CA certificate

- out *privateKeyCACertUplink* - CA certificate private key

7.2. 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.9 with the following input and output parameters:

- in *cap* - DUT capabilities
- in "CN=ONVIF TT Uplink 2,C=US" - certificate subject
- in *CACertUplink* - CA-certificate
- in *privateKeyCACertUplink* - private key of CA-certificate for certificate signature
- out *certUplink* - certificate
- out *publicKeyUplink* - public key of certificate
- out *privateKeyUplink* - private key of certificate

7.3. ONVIF Client configure and start service endpoint for uplink connection at address *uplinkAddress1* using *certUplink* certificate.

7.4. ONVIF Client invokes **SetUplink** request with parameters

- RemoteAddress := *uplinkAddress1*
- CertificateID is skipped
- UserLevel := **Administrator**
- Status is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDUplink*
- AuthorizationServer := *authServerToken1*
- Error is skipped

7.5. The DUT responds with **SetUplinkResponse** message.

8. ONVIF Client verifies initial connection establishment:

8.1. The DUT opens connection to *authServerMetadataEndpoint1* with client identifier *clientID1* and client secret *clientSecret1* using client_secret_basic method authentication and OAuth2 client credentials grant flow.

8.2. The DUT verifies certificate provided by authentication server based on *certPathValidationPolicyIDAuthServer* certification path validation policy.

8.3. The DUT receives access token *accessToken1* from authorization server for scope *scope1*.

8.4. ONVIF Client awaits device connecting to *uplinkAddress1* endpoint.

8.5. DUT opens connection to *uplinkAddress1* with *accessToken1*.

8.6. DUT verifies certificate provided by ONVIF Client at uplink connection establishment based on *certPathValidationPolicyIDUplink* certification path validation policy.

8.7. ONVIF Client verify access token received from the DUT with Authentication Server. If received access token *accessToken1* is not valid for *scope1*, FAIL the test, restore the DUT state, and skip other steps.

8.8. ONVIF Client invokes **SetAuthorizationServerConfiguration** request using uplink connection with parameters without any additional authentication

- @token := *authServerToken1*
- Data.Type := **"OAuthClientCredentials"**
- Data.ClientAuth := **"client_secret_basic"**
- Data.ServerUri := *authServerMetadataEndpoint1*
- Data.ClientID := *clientID1*
- Data.ClientSecret := *clientSecret1*
- Data.Scope := *scope2*
- Data.KeyID is skipped
- Data.CertificateID is skipped
- Data.CertPathValidationPolicyID := *certPathValidationPolicyIDAuthServer*

8.9. The DUT responds with **SetAuthorizationServerConfigurationResponse** message.

9. ONVIF Client restores the DUT state.

Test Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **CreateAuthorizationServerConfigurationResponse** message.
- DUT did not send **SetUplinkResponse** message.
- DUT did not send **SetAuthorizationServerConfigurationResponse** message.
- DUT does not establish connection at step 8.5.

5.4 Capabilities

5.4.1 GET SERVICES AND GET UPLINK SERVICE CAPABILITIES CONSISTENCY

Test Case ID: UPLINK-4-1-1

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

Feature Under Test: GetServices, GetServiceCapabilities (Uplink)

WSDL Reference: devicemgmt.wsdl, uplink.wsdl

Test Purpose: To verify getting Uplink Service using GetServices request. To verify Get Services and Uplink Service Capabilities consistency.

Pre-Requisite: Uplink 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 **GetServices** message with parameters:
   - IncludeCapability := false
4. The DUT responds with a **GetServicesResponse** message with parameters:
   - Service list =: *listOfServicesWithoutCapabilities*

5. If *listOfServicesWithoutCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/uplink/wsdl", FAIL the test and skip other steps.
6. Set *uplinkServ* := item from *listOfServicesWithoutCapabilities* list with Namespace = "http://www.onvif.org/ver10/uplink/wsdl".
7. If *uplinkServ*.Capabilities is specified, FAIL the test and skip other steps.
8. ONVIF Client invokes **GetServices** message with parameters:
   - IncludeCapability := true
9. The DUT responds with a **GetServicesResponse** message with parameters:
   - Service list =: *listOfServicesWithCapabilities*
10. If *listOfServicesWithCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/uplink/wsdl", FAIL the test and skip other steps.
11. Set *uplinkServ* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver10/uplink/wsdl".
12. If *uplinkServ*.Capabilities is not specified, FAIL the test and skip other steps.
13. If *uplinkServ*.Capabilities does not contain valid Capabilities element for Uplink service from "http://www.onvif.org/ver10/uplink/wsdl" namespace, FAIL the test and skip other steps.
14. ONVIF Client invokes **GetServiceCapabilities** (Uplink) message.
15. The DUT responds with **GetServiceCapabilitiesResponse** message with parameters
    - Capabilities =: *cap*
16. If *cap* differs from *uplinkServ*.Capabilities, FAIL the test and skip other steps.
17. If *cap.MaxUplinks* is not specified or *cap.MaxUplinks* is less than or equal to 0, FAIL the test and skip other steps.
18. If *cap.Protocols* is not specified or *cap.Protocols* field is empty, FAIL the test and skip other steps.
19. If *cap.AuthorizationModes* is not specified or *cap.AuthorizationModes* field is empty, FAIL the test.

Test Result:

PASS –

- DUT passed all assertions.

FAIL –

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

Note: The following fields are compared at step 16:

- MaxUplinks
- Protocols
- AuthorizationModes
- StreamingOverUplink

Annex A Helper Procedures and Additional Notes

A.1 Configure Uplink Security

Name: HelperConfigureUplinkSecurity

Notes: Annex is used at:

- ONVIF Uplink Device Test Specification

Procedure Purpose: Helper procedure to prepare security parameters for uplink connection.

Pre-requisite:

- Uplink Service is received from the DUT.
- Security Configuration Service is received from the DUT.

Input:

- Uplink capabilities (*capUplink*).
- Flag to generate security primitives for both authentication types (*generateAllSecurityPrimitives*, optional, false by default).

Returns:

- Certification path validation policy for uplink connection (*certPathValidationPolicyIDUplink*).
- Certificate that could be used for mTLS authentication (*certID1*) (optional).
- Authorization server configuration that could be used for access token authentication (*authServerToken1*) (optional).
- key pair algorithm that will be used for security primitives generation (*keyAlgorithm*).

Procedure:

1. ONVIF Client gets the security configuration service capabilities by following the procedure mentioned in Annex A.8 with the following input and output parameters:
   - out *capSecurity* - Security Configuration Service Capabilities
2. ONVIF Client selects key pair algorithm by following the procedure mentioned in Annex A.17 with the following input and output parameters:
   - in *cap* - security configuration service capabilities

- out *keyAlgorithm* - key pair algorithm

3. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.2 with the following input and output parameters
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - in "CN=ONVIF TT Uplink 1,C=US" - CA certificate subject
   - in "Test CertPathValidationPolicy Uplink Alias" - certification path validation policy alias
   - out *certPathValidationPolicyIDUplink* - certification path validation policy identifier
   - out *certIDUplink* - certificate identifier
   - out *keyIDUplink* - key pair identifier
   - out *CACertUplink* - CA certificate
   - out *privateKeyCACertUplink* - CA certificate private key
4. If the DUT supports mTLS authentication as indicated by *capUplink*.AuthorizationModes = mTLS, ONVIF Client configures mTLS certificate and uplink server using the following steps:
   4.1. 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.9 with the following input and output parameters:
      - in *cap* - DUT capabilities
      - in "CN=ONVIF TT Uplink 2,C=US" - certificate subject
      - in *CACertUplink* - CA-certificate
      - in *privateKeyCACertUplink* - private key of CA-certificate for certificate signature
      - out *certUplink* - certificate
      - out *publicKeyUplink* - public key of certificate
      - out *privateKeyUplink* - private key of certificate

   4.2. ONVIF Client creates a CA certificate and a corresponding key pair by following the procedure described in Annex A.3 with the following input and output parameters:

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

4.3. ONVIF Client creates and uploads a CA-signed certificate for key pair based on associated CA certificate and a corresponding private key by following the procedure described in Annex A.11 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 identificator
- out *keyID1* - key pair
- out *cert1* - CA-signed certificate

4.4. If *generateAllSecurityPrimitives* = true go to the step 5, otherwise end the procedure and continue with the test.

5. If the DUT supports Access Token authentication as indicated by *capUplink*.AuthorizationModes = AccessToken, ONVIF Client configures authentication server using the following steps:

5.1. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.2 with the following input and output parameters

- in *cap* - DUT capabilities
- in *keyAlgorithm* - key pair algorithm
- in "CN=ONVIF TT AuthServer 1,C=US" - CA certificate subject
- in "Test CertPathValidationPolicy AuthServer Alias" - certification path validation policy alias
- out *certPathValidationPolicyIDAuthServer* - certification path validation policy identifier

- out *certIDAuthServer* - certificate identifier
- out *keyIDAuthServer* - key pair identifier
- out *CACertAuthServer* - CA certificate
- out *privateKeyCACertAuthServer* - CA certificate private key

5.2. If the DUT supports client_secret_basic authentication as indicated by "client_secret_basic" in *capSecurity*.ClientAuthenticationMethodsSupported list, ONVIF Client configures authentication server using the following steps:

5.2.1. 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.9 with the following input and output parameters:

- in *cap* - DUT capabilities
- in "CN=ONVIF TT AuthServer 2,C=US" - certificate subject
- in *CACertAuthServer* - CA-certificate
- in *privateKeyCACertAuthServer* - private key of CA-certificate for certificate signature
- out *certAuthServer* - certificate
- out *publicKeyAuthServer* - public key of certificate
- out *privateKeyAuthServer* - private key of certificate

5.2.2. ONVIF Client starts Authorization server with metadata endpoint *authServerMetadataEndpoint1* conforming to RFC8414 with following settings:

- It is configured to *certAuthServer* as a server certificate.
- It is configured to accept OAuth2 client credentials grant flow per [RFC 6749] authentication method only.
- It is configured to accept client_secret_basic authentication method only.
- Client with client identifier *clientID1* with client secret *clientSecret1* and scope *scope1* is added to be authorized.

5.2.3. Set *clientAuth1* := **"client_secret_basic"**.

5.2.4. Go to step 5.5.

5.3. If the DUT supports private_key_jwt authentication as indicated by "private_key_jwt" in *capSecurity*.ClientAuthenticationMethodsSupported list, ONVIF Client configures authentication server using the following steps:

5.3.1. ONVIF Client creates a key pair and get public key for JWT authentication from device by following the procedure described in Annex A.18 with the following input and output parameters:

- in *keyAlgorithm* - CSR key pair algorithm
- out *keyID1Auth1* - key pair
- out *publicKeyAuth1* - public key

5.3.2. ONVIF Client starts Authorization server with metadata endpoint *authServerMetadataEndpoint1* conforming to RFC8414 with following settings:

- It is configured to *certAuthServer* as a server certificate.
- It is configured to accept OAuth2 client credentials grant flow per [RFC 6749] authentication method only.
- It is configured to accept private_key_jwt authentication method only.
- Client with client identifier *clientID1* with *publicKeyAuth1* as key, and scope *scope1* is added to be authorized.

5.3.3. Set *clientAuth1* := **"private_key_jwt"**.

5.3.4. Go to step 5.5.

5.4. FAIL the test, restore the DUT state, and skip other steps.

5.5. ONVIF Client invokes **CreateAuthorizationServerConfiguration** request with parameters

- Type := **"OAuthClientCredentials"**
- ClientAuth := *clientAuth1*
- ServerUri := *authServerMetadataEndpoint1*
- ClientID := *clientID1*

- ClientSecret := *clientSecret1*, if *clientSecret1* was defined, otherwise field is skipped
- Scope := *scope1*
- KeyID := *keyID1Auth1*, if *keyID1Auth1* was defined, otherwise field is skipped
- CertificateID is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDAuthServer*

5.6. The DUT responds with **CreateAuthorizationServerConfigurationResponse** message with parameters

- Token =: *authServerToken1*

5.7. End the procedure and continue with the test.

6. FAIL the test, restore the DUT state, and skip other steps.

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

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

A.2 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 Device Test Specification
- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Media2 Configuration Device Test Specification
- ONVIF Base 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.3 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.7 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.3 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
- ONVIF Uplink Device Test Specification
- ONVIF Recording Control 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.4 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.5 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.4 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
- ONVIF Uplink Device Test Specification

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to select signature algorithm which 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)
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)
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.5 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
- ONVIF Uplink Device Test Specification
- ONVIF Recording Control 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 parameters by following the procedure mentioned in Annex A.6 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *keyGenParams* - key pair generation parameters
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.6 Determine key pair generation parameters

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
- ONVIF Uplink Device Test Specification
- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to determine the key pair generation parameters 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 RSAKeyPairGeneration capability, or by the ECCKeyPairGeneration capability, or by the KeyPairGeneration capability, which contains RSA or ECC.

Input: The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*).

Returns: The key pair generation parameters (*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 parameters (*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 parameters (*keyGenParams*) with the elliptic curve (*ellipticCurve*).

3. If previous steps is done with empty key pair generation parameters (*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.7 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 Device Test Specification
- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Recording Control Device Test Specification
- ONVIF Base 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 is 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.8 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
- ONVIF Uplink Device Test Specification
- ONVIF Recording Control 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** request.
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.9 Provide certificate signed by private key of other certificate

Name: HelperCreateSignedCertificate

Notes: Annex is used at:

- ONVIF Security Configuration Device Test Specification
- ONVIF Uplink Device Test Specification
- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Base 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.5 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.4 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.10 Get Uplink Service Capabilities

Name: HelperGetServiceCapabilities_Uplink

Notes: Annex is used at:

- ONVIF Uplink Device Test Specification

Procedure Purpose: Helper procedure to get Uplink Service Capabilities from the DUT.

Pre-requisite: Uplink Service is received from the DUT.

Input: None

Returns: The service capabilities (*cap*).

Procedure:

1. ONVIF Client invokes **GetServiceCapabilities** request.
2. The DUT responds with **GetServiceCapabilitiesResponse** message with parameters
   - Capabilities =: *cap*

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

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

A.11 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 RSAKeyPairGeneration capability, or by the ECCKeyPairGeneration capability, or by the KeyPairGeneration capability, which contains RSA or ECC.
- 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.12 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.12 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 RSAKeyPairGeneration capability, or by the ECCKeyPairGeneration capability, or by the KeyPairGeneration capability, which contains RSA or ECC.
- 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.13 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.4 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.15)
- 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.16 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.13 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

- ONVIF Uplink 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 RSAKeyPairGeneration capability, or by the ECCKeyPairGeneration capability, or by the KeyPairGeneration capability, which contains RSA or ECC.
- The DUT shall have enough free storage capacity for one additional key pair.

Input:

- The service capabilities (*cap*) (optional, required if *keyPairParameters* are not defined).
- The key pair algorithm (*keyAlgorithm*).
- The key pair parameters (key length for RSA; curve for ECC) (*keyGenParams*) (optional).

Returns:

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

Procedure:

1. If *keyGenParams* is not defined, ONVIF Client determines the key pair generation parameters by following the procedure mentioned in Annex A.6 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *keyGenParams* - key pair generation parameters
2. If *keyAlgorithm* = 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 *keyAlgorithm* = 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.14.
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.14.

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.14 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
- ONVIF Uplink Device Test Specification

Procedure Purpose: Helper procedure to delete a 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 RSAKeyPairGeneration capability, or by the ECCKeyPairGeneration capability, or by the KeyPairGeneration capability, which contains RSA or ECC.

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.15 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
- ONVIF Uplink Device Test Specification

Use the following subject for test cases:

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

A.16 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.4 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.17 Selection of key pair algorithm

Name: HelperKeyAlgorithmSelection

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 Uplink Device Test Specification
- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to select key pair algorithm for the test depending on device capabilities.

Pre-requisite:

- Security Configuration Service is received from the DUT.
- On-board key pair generation is supported by the DUT as indicated by the RSAKeyPairGeneration capability, or by the ECCKeyPairGeneration capability, or by the KeyPairGeneration capability.

Input: Advanced security capabilities (*cap*).

Returns: key pair algorithm (*keyAlgorithm*).

Procedure:

1. If *cap*.KeystoreCapabilities.ECCKeyPairGeneration = true or *cap*.KeystoreCapabilities.KeyPairGeneration contains **ECC**:

- Set *keyAlgorithm* := **ECC**.
- Skip other steps of procedure and return to the test.

2. If *cap*.KeystoreCapabilities.RSAKeyPairGeneration = true or *cap*.KeystoreCapabilities.KeyPairGeneration contains **RSA**:
   - Set *keyAlgorithm* := **RSA**.
   - Skip other steps of procedure and return to the test.
3. FAIL the test, restore the DUT state, and skip other steps.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

A.18 Create Key Pair and Receive Public Key

Name: HelperCreateJWTKeyPairAndReceivePublicKey

Notes: Annex is used at:

- ONVIF Base Device Test Specification
- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Media2 Configuration Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Uplink Test Specification

Procedure Purpose: Helper procedure to create a key pair and get public key from the device.

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 RSAKeyPairGeneration capability, or by the ECCKeyPairGeneration capability, or by the KeyPairGeneration capability, which contains RSA or ECC.
- 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:

- The JWT algorithm (*jwtAlgorithm*) (optional).
- The CSR key pair algorithm (*keyAlgorithm*) (optional).
- **Note:** At least one of *keyAlgorithm* and *jwtAlgorithm* to be defined.

Returns:

- The identifier of the new key pair (*keyID*).
- Th public key (*publicKey*).

Procedure:

1. If *jwtAlgorithm* is not defined, set
   - *jwtAlgorithm* := [key pair algorithm as defined for *keyAlgorithm* for latest [ONVIF Security Baseline] version supported by the DUT in section Private Key JWT Algorithms for Baseline Versions of [ONVIF Security Baseline Device Test Spec]]
2. ONVIF Client creates a key pair by following the procedure mentioned in Annex A.13 with the following input and output parameters:
   - in [key pair algorithm as defined for *jwtAlgorithm* in section Private Key JWT Algorithms for Baseline Versions of [ONVIF Security Baseline Device Test Spec]] - key pair algorithm
   - in [key pair parameters as defined for *jwtAlgorithm* in section Private Key JWT Algorithms for Baseline Versions of [ONVIF Security Baseline Device Test Spec]] - key pair algorithm parameters
   - out *csrKeyID* - key pair ID
3. ONVIF Client invokes **CreatePKCS10CSR** with parameters
   - Subject := *subject* (see Annex A.15)
   - KeyID := *csrKeyID*

- CSRAttribute skipped
- SignatureAlgorithm.algorithm := signature algorithm, used for *jwtAlgorithm*

4. The DUT responds with **CreatePKCS10CSRResponse** message with parameters
   - PKCS10CSR =: *PKCS10request*
5. ONVIF Client extracts public key *publicKey* from *PKCS10request*.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.19 Clean Up Uplink Configurations

Name: HelperCleanUpUplinkConfigurations

Procedure Purpose: Helper procedure to remove all uplink configurations from the DUT.

Pre-requisite: Uplink Service is received from the DUT.

Input: None

Returns: None.

Procedure:

1. ONVIF Client invokes **GetUplinks** request.
2. The DUT responds with **GetUplinksResponse** message with parameters
   - Uplink Configurations list =: *uplinkConfigList*
3. For each uplink configuration (*uplinkConfig*) from *uplinkConfigList* repeat the following steps:
   3.1. ONVIF Client invokes **DeleteUplink** request with parameters
      - RemoteAddress := *uplinkConfig*.RemoteAddress
   3.2. The DUT responds with **DeleteUplinkResponse** message.

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **GetUplinksResponse** message.
- DUT did not send **DeleteUplinkResponse** message.

A.20 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 Device Test Specification
- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Base 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.21 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.21 Get Authorization Server Configurations List

Name: HelperGetAuthorizationServerConfigurations

Notes: Annex is used at:

- ONVIF Security Configuration Device Test Specification
- ONVIF Uplink Device Test Specification
- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Base 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.