ONVIF®

Authentication Behavior
Device Test Specification

Version 18.12

December 2018
This document is provided "as is," and the corporation and its members and their affiliates, make no representations or warranties, express or implied, including but not limited to, warranties of merchantability, fitness for a particular purpose, non-infringement, or title; that the contents of this document are suitable for any purpose; or that the implementation of such contents will not infringe any patents, copyrights, trademarks or other rights.

In no event will the corporation or its members or their affiliates be liable for any direct, indirect, special, incidental, punitive or consequential damages, arising out of or relating to any use or distribution of this document, whether or not (1) the corporation, members or their affiliates have been advised of the possibility of such damages, or (2) such damages were reasonably foreseeable, and arising out of or relating to any use or distribution of this document. The foregoing disclaimer and limitation on liability do not apply to, invalidate, or limit representations and warranties made by the members and their respective affiliates to the corporation and other members in certain written policies of the corporation.
### REVISION HISTORY

<table>
<thead>
<tr>
<th>Vers.</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.12</td>
<td>Aug 17, 2018</td>
<td>First issue of Authentication Behavior Device Test Specification</td>
</tr>
<tr>
<td>18.12</td>
<td>Dec 21, 2018</td>
<td>Switching Hub description in 'Network Configuration for DUT' section was added according to #1737</td>
</tr>
</tbody>
</table>
# Table of Contents

1 **Introduction** ........................................................................................................................ 9

1.1 **Scope** .......................................................................................................................... 9

1.1.1 Capabilities ............................................................................................................ 10

1.1.2 Authentication Profile Info .................................................................................. 10

1.1.3 Authentication Profile ......................................................................................... 10

1.1.4 Authentication Profile Management ................................................................. 10

1.1.5 Security Level Info .............................................................................................. 11

1.1.6 Security Level ...................................................................................................... 11

1.1.7 Security Level Management ............................................................................. 11

1.1.8 Authentication Behavior Events ................................................................... 12

2 **Normative references** ...................................................................................................... 13

3 **Terms and Definitions** ..................................................................................................... 15

3.1 Conventions ................................................................................................................. 15

3.2 Definitions .................................................................................................................. 15

3.3 Abbreviations .............................................................................................................. 15

4 **Test Overview** .................................................................................................................. 16

4.1 Test Setup .................................................................................................................. 16

4.1.1 Network Configuration for DUT ..................................................................... 16

4.2 Prerequisites .............................................................................................................. 17

4.3 Test Policy .................................................................................................................. 17

4.3.1 Capabilities ......................................................................................................... 17

4.3.2 Authentication Profile Info ................................................................................ 18

4.3.3 Authentication Profile ......................................................................................... 19

4.3.4 Authentication Profile Management ................................................................. 20

4.3.5 Security Level Info .............................................................................................. 23

4.3.6 Security Level ...................................................................................................... 24

4.3.7 Security Level Management ............................................................................. 26

4.3.8 Authentication Behavior Events ................................................................... 29

5 **Authentication Profile Test Cases** .................................................................................. 31

5.1 Capabilities .................................................................................................................. 31
5.1.1 AUTHENTICATION BEHAVIOR SERVICE CAPABILITIES ................................. 31
5.2 Authentication Profile Info ................................................................................ 33
  5.2.1 GET AUTHENTICATION PROFILE INFO .................................................... 33
  5.2.2 GET AUTHENTICATION PROFILE INFO LIST - LIMIT ............................... 35
  5.2.3 GET AUTHENTICATION PROFILE INFO LIST - START REFERENCE AND LIMIT ................................................................. 37
  5.2.4 GET AUTHENTICATION PROFILE INFO LIST - NO LIMIT .................... 41
  5.2.5 GET AUTHENTICATION PROFILE INFO WITH INVALID TOKEN .......... 43
  5.2.6 GET AUTHENTICATION PROFILE INFO - TOO MANY ITEMS ............... 45
5.3 Authentication Profile ........................................................................................... 47
  5.3.1 GET AUTHENTICATION PROFILES ........................................................... 47
  5.3.2 GET AUTHENTICATION PROFILE LIST - LIMIT ........................................ 49
  5.3.3 GET AUTHENTICATION PROFILE LIST - START REFERENCE AND LIMIT ........................................................................................................... 52
  5.3.4 GET AUTHENTICATION PROFILE LIST - NO LIMIT ............................... 56
  5.3.5 GET AUTHENTICATION PROFILES WITH INVALID TOKEN .................... 58
  5.3.6 GET AUTHENTICATION PROFILE - TOO MANY ITEMS ........................ 60
5.4 Authentication Profile Management ..................................................................... 61
  5.4.1 CREATE AUTHENTICATION PROFILE WITHOUT AUTHENTICATION POLICIES ......................................................................................... 61
  5.4.2 CREATE AUTHENTICATION PROFILE WITH AUTHENTICATION POLICY ................................................................................................. 65
  5.4.3 MODIFY AUTHENTICATION PROFILE ..................................................... 65
  5.4.4 DELETE AUTHENTICATION PROFILE ...................................................... 70
  5.4.5 SET NEW AUTHENTICATION PROFILE WITHOUT AUTHENTICATION POLICIES ......................................................................................... 77
  5.4.6 SET NEW AUTHENTICATION PROFILE WITH AUTHENTICATION POLICY ................................................................................................. 80
  5.4.7 SET AUTHENTICATION PROFILE .............................................................. 84
  5.4.8 CREATE AUTHENTICATION PROFILE - NOT EMPTY TOKEN ............... 88
5.4.9 CREATE AUTHENTICATION PROFILE - CAPABILITY VIOLATED (MAX POLICIES PER AUTHENTICATION PROFILE) .......................................................... 97
5.4.10 MODIFY AUTHENTICATION PROFILE - INVALID TOKEN .................... 100
5.4.11 MODIFY AUTHENTICATION PROFILE - CAPABILITY VIOLATED (MAX POLICIES PER AUTHENTICATION PROFILE) ................................................... 101
5.4.12 SET AUTHENTICATION PROFILE - CAPABILITY VIOLATED (MAX POLICIES PER AUTHENTICATION PROFILE) ..................................................... 104
5.4.13 SET AUTHENTICATION PROFILE - EMPTY TOKEN ............................. 106
5.4.14 DELETE AUTHENTICATION PROFILE - INVALID TOKEN .................... 108
5.4.15 DELETE AUTHENTICATION PROFILE - NO TOKEN ............................. 109

5.5 Security Level Info ............................................................................................. 110
5.5.1 GET SECURITY LEVEL INFO ................................................................. 110
5.5.2 GET SECURITY LEVEL INFO LIST - LIMIT .............................................. 112
5.5.3 GET SECURITY LEVEL INFO LIST - START REFERENCE AND LIMIT .... 114
5.5.4 GET SECURITY LEVEL INFO LIST - NO LIMIT ........................................... 117
5.5.5 GET SECURITY LEVEL INFO WITH INVALID TOKEN ............................. 119
5.5.6 GET SECURITY LEVEL INFO - TOO MANY ITEMS ................................. 121

5.6 Security Level .................................................................................................... 122
5.6.1 GET SECURITY LEVELS ............................................................................. 122
5.6.2 GET SECURITY LEVEL LIST - LIMIT ....................................................... 124
5.6.3 GET SECURITY LEVEL LIST - START REFERENCE AND LIMIT ............. 126
5.6.4 GET SECURITY LEVEL LIST - NO LIMIT .................................................. 131
5.6.5 GET SECURITY LEVELS WITH INVALID TOKEN .................................... 133
5.6.6 GET SECURITY LEVEL - TOO MANY ITEMS .......................................... 134

5.7 Security Level Management .............................................................................. 135
5.7.1 CREATE SECURITY LEVEL WITHOUT RECOGNITION GROUPS .......... 135
5.7.2 CREATE SECURITY LEVEL WITHOUT RECOGNITION METHODS ........ 139
5.7.3 CREATE SECURITY LEVEL WITH RECOGNITION METHODS ............... 142
5.7.4 MODIFY SECURITY LEVEL ................................................................. 146
5.7.5 DELETE SECURITY LEVEL ................................................................. 152
5.7.6 SET SECURITY LEVEL WITHOUT RECOGNITION GROUPS ............... 154
5.7.7 SET SECURITY LEVEL WITHOUT RECOGNITION METHODS ............... 158
5.7.8 SET SECURITY LEVEL WITH RECOGNITION METHODS ............... 161
5.7.9 SET SECURITY LEVEL .......................................................... 165
5.7.10 CREATE SECURITY LEVEL - NOT EMPTY TOKEN .................... 171
5.7.11 CREATE SECURITY LEVEL - CAPABILITY VIOLATED (MAX
RECOGNITION GROUPS PER SECURITY LEVEL) ............................ 172
5.7.12 CREATE SECURITY LEVEL - CAPABILITY VIOLATED (MAX
RECOGNITION METHODS PER RECOGNITION GROUP) ............. 174
5.7.13 CREATE SECURITY LEVEL - DUPLICATE PRIORITY ............... 176
5.7.14 MODIFY SECURITY LEVEL - INVALID TOKEN ................. 178
5.7.15 MODIFY SECURITY LEVEL - CAPABILITY VIOLATED (MAX
RECOGNITION GROUPS PER SECURITY LEVEL) .................... 179
5.7.16 MODIFY SECURITY LEVEL - CAPABILITY VIOLATED (MAX
RECOGNITION METHODS PER RECOGNITION GROUP) .......... 181
5.7.17 MODIFY SECURITY LEVEL - DUPLICATE PRIORITY ............ 184
5.7.18 SET SECURITY LEVEL - CAPABILITY VIOLATED (MAX
RECOGNITION GROUPS PER SECURITY LEVEL) ..................... 186
5.7.19 SET SECURITY LEVEL - CAPABILITY VIOLATED (MAX
RECOGNITION METHODS PER RECOGNITION GROUP) ........... 188
5.7.20 SET SECURITY LEVEL - DUPLICATE PRIORITY ............. 190
5.7.21 SET SECURITY LEVEL - EMPTY TOKEN ......................... 192
5.7.22 DELETE SECURITY LEVEL - INVALID TOKEN ............... 193
5.7.23 DELETE SECURITY LEVEL - NO TOKEN ....................... 194
5.8 Authentication Behavior Events .............................................. 195
5.8.1 AUTHENTICATION PROFILE CHANGED EVENT ................. 195
5.8.2 AUTHENTICATION PROFILE REMOVED EVENT ................. 196
5.8.3 AUTHENTICATION PROFILE CHANGED EVENT ................. 198
5.8.4 AUTHENTICATION PROFILE REMOVED EVENT ............... 199
A Helper Procedures and Additional Notes ...................................... 201
A.1 Get Authentication Profiles Information List .......................... 201
A.2 Get Service Capabilities ......................................................... 202
A.3 Get Authentication Profiles List ................................................................. 202
A.4 Create Number of Authentication Profiles .................................................. 204
A.5 Find or Create Security Level ................................................................. 205
A.6 Get Security Levels Information List ......................................................... 207
A.7 Compare Authentication Profile List and Authentication Profile Info List .......... 208
A.8 Create Pull Point Subscription ............................................................... 209
A.9 Delete Subscription ............................................................................. 210
A.10 Retrieve Authentication Profile Changed Event by PullPoint ....................... 210
A.11 Get Authentication Profile .................................................................... 211
A.12 Get Authentication Profile Info ............................................................ 212
A.13 Delete Authentication Profile ............................................................... 213
A.14 Find or Create Schedule ...................................................................... 213
A.15 Generate iCalendar Value for Schedule .................................................... 214
A.16 Create Authentication Profile ............................................................... 215
A.17 Create Security Level ........................................................................ 217
A.18 Get Schedules Information List ............................................................. 218
A.19 Get Schedule Service Capabilities .......................................................... 219
A.20 Create Schedule .................................................................................. 219
A.21 Retrieve Authentication Profile Removed Event by PullPoint ..................... 220
A.22 Delete Security Level ........................................................................... 222
A.23 Delete Schedule .................................................................................. 222
A.24 Create Number of Security Levels ........................................................ 223
A.25 Get Security Level List ........................................................................ 224
A.26 Compare Security Level List and Security Level Info List ......................... 225
A.27 Retrieve Security Level Changed Event by PullPoint .................................. 226
A.28 Retrieve Security Level Removed Event by PullPoint ................................ 228
A.29 Get Security Level Info ....................................................................... 229
A.30 Get Security Level ............................................................................... 229
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 Network Interface Specs] and [ONVIF Conformance] requirements. In addition, the test cases are to be basic inputs for some Profile specification requirements. It also describes the test framework, test setup, pre-requisites, test policies needed for the execution of the described test cases.

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

1.1 Scope

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

The principal intended purposes are:


2. Provide comprehensive test suite coverage for [ONVIF Network Interface Specs].

This specification does not address the following:

1. Product use cases and non-functional (performance and regression) testing.

2. SOAP Implementation Interoperability test i.e. Web Services Interoperability Basic Profile version 2.0 (WS-I BP2.0).

3. Network protocol implementation Conformance test for HTTPS, HTTP, RTP and RTSP protocols.

4. Wi-Fi Conformance test

The set of ONVIF Test Specification will not cover the complete set of requirements as defined in [ONVIF Network Interface Specs]; instead it will cover its subset.
This ONVIF Authentication Behavior Test Specification covers the ONVIF Authentication Behavior Service, which is a functional block of [ONVIF Network Interface Specs]. The following section gives a brief overview of each functional block and its scope.

1.1.1 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 Authentication Behavior service address with GetServices command via Device service
- Getting capabilities with GetServiceCapabilities command
- Getting capabilities with GetServices command via Device service

1.1.2 Authentication Profile Info

The Authentication Profile Info section covers the test cases needed for getting authentication profile list and information from an ONVIF device.

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

- Getting authentication profile information list with GetAuthenticationProfileInfoList command
- Getting authentication profile information with GetAuthenticationProfileInfo command

1.1.3 Authentication Profile

The Authentication Profile section covers the test cases needed for getting authentication profile list from an ONVIF device.

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

- Getting authentication profile information list with GetAuthenticationProfileList command
- Getting authentication profile information with GetAuthenticationProfiles command

1.1.4 Authentication Profile Management

The Authentication Profile section covers the test cases needed for create, modify, delete and set authentication profile on an ONVIF device.
The scope of this specification section is to cover the following functions:

- Creating authentication profile with CreateAuthenticationProfile command
- Modifying authentication profile with ModifyAuthenticationProfile command
- Deleting authentication profile with DeleteAuthenticationProfile command
- Set authentication profile with SetAuthenticationProfile command

1.1.5 Security Level Info

The Security Level Info section covers the test cases needed for getting security level list and information from an ONVIF device.

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

- Getting security level information list with GetSecurityLevelInfoList command
- Getting security level information with GetSecurityLevelInfo command

1.1.6 Security Level

The Security Level section covers the test cases needed for getting security level list from an ONVIF device.

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

- Getting security level information list with GetSecurityLevelList command
- Getting security level information with GetSecurityLevels command

1.1.7 Security Level Management

The Security Level section covers the test cases needed for create, modify, delete and set security level on an ONVIF device.

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

- Creating security level with CreateSecurityLevel command
- Modifying security level with ModifySecurityLevel command
- Deleting security level with DeleteSecurityLevel command
- Set security level with SetSecurityLevel command
1.1.8 Authentication Behavior Events

The Authentication Behavior Events section covers the test cases needed for checking specified events format.

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

- Getting event properties with GetEventProperties command for the following events:
  - tns1:Configuration/AuthenticationProfile/Changed
  - tns1:Configuration/AuthenticationProfile/Removed
  - tns1:Configuration/SecurityLevel/Changed
  - tns1:Configuration/SecurityLevel/Removed
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 Specification:  
  https://www.onvif.org/profiles/specifications/

- [ONVIF Schedule Spec] ONVIF Schedule Specification:  
  https://www.onvif.org/profiles/specifications/

- [ONVIF Authentication Behavior Spec] ONVIF Authentication Behavior Specification:  
  https://www.onvif.org/profiles/specifications/

  http://www.iso.org/directives


- [SOAP 1.2, Part 1] W3C SOAP 1.2, Part 1, Messaging Framework:  
  http://www.w3.org/TR/soap12-part1/

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

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

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 Authentication Behavior Spec] and tests.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication Policy</td>
<td>Each authentication policy associates a security level with a schedule (during which the specified security level will be required at the access point).</td>
</tr>
<tr>
<td>Authentication Profile</td>
<td>Authentication profiles are used to define authentication behaviour for a type of access points. For instance, all entrance access points are configured to require Card access during office hours, Card+PIN access during night-time, and no access during holidays.</td>
</tr>
<tr>
<td>Recognition</td>
<td>Recognition is the action of identifying authorized users requesting access by the comparison of presented credential data with recorded credential data.</td>
</tr>
<tr>
<td>Recognition Group</td>
<td>Recognition groups are used to define a logical OR between the recognition methods in a security level. Example: One recognition group contains the recognition methods pt:Card and pt:Fingerprint. Another group contains the recognition methods pt:Card and pt:Face. The resulting effect is that the access point will require either Card+Fingerprint, or Card+Face.</td>
</tr>
<tr>
<td>Recognition Method</td>
<td>A recognition method is either memorized, biometric or held within a physical credential.</td>
</tr>
<tr>
<td>Recognition Type</td>
<td>A recognition type is either a recognition method or a physical input such as a request-to-exit button.</td>
</tr>
<tr>
<td>Security Level</td>
<td>Security Levels are defined as individual recognition methods, combinations of recognition methods (using logical AND or OR), or no recognition methods (open). Security levels are given explanatory names, such as “Card”, “Card+ PIN”, “Fingerprint or Iris”, “Open”, etc.</td>
</tr>
</tbody>
</table>

3.3 Abbreviations

This section describes abbreviations used in this document.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUT</td>
<td>Device Under Test</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>PACS</td>
<td>Physical Access Control System</td>
</tr>
</tbody>
</table>
4 Test Overview

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

4.1 Test Setup

4.1.1 Network Configuration for DUT

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

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

Figure 4.1. Test Configuration for DUT

- **DUT**: ONVIF device to be tested. Hereafter, this is referred to as DUT (Device Under Test).
- **ONVIF Client (Test Tool)**: Tests are executed by this system and it controls the behavior of the DUT. It handles both expected and unexpected behavior.
- **HTTP Proxy**: provides facilitation in case of RTP and RTSP tunneling over HTTP.
- **Wireless Access Point**: provides wireless connectivity to the devices that support wireless connection.
DNS Server: provides DNS related information to the connected devices.

DHCP Server: provides IPv4 Address to the connected devices.

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

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

Router: provides router advertisements for IPv6 configuration.

4.2 Prerequisites

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

• The DUT shall be configured with an IPv4 address.

• The DUT shall be IP reachable in the test configuration.

• The DUT shall be able to be discovered by the Test Tool.

• The DUT shall be configured with the time, i.e. manual configuration of UTC time and if NTP is supported by DUT, then NTP time shall be synchronized with 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.

4.3.1 Capabilities

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

• DUT shall give the Authentication Behavior 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 5.1 for Capabilities Test Cases.

### 4.3.2 Authentication Profile Info

The test policies specific to the test case execution of Authentication Profile Info functional block:

- DUT shall give the Authentication Behavior 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
  - GetAuthenticationProfileInfo
  - GetAuthenticationProfileInfoList
  - CreateAuthenticationProfile
  - DeleteAuthenticationProfile
  - CreateSecurityLevel
  - GetSecurityLevelInfoList
  - DeleteSecurityLevel

- DUT shall not return more items in GetAuthenticationProfileInfo and GetAuthenticationProfileInfoList responses than specified in service capabilities by MaxLimit.

- DUT shall not return more items in GetAuthenticationProfileInfoList response than specified by Limit parameter in a request.

- DUT shall not return items with the same tokens in GetAuthenticationProfileInfoList responses for one authentication profile info list receiving.

- DUT shall not return more AuthenticationProfileInfo items in GetAuthenticationProfileInfoList responses than specified in service capabilities by MaxAuthenticationProfiles.
• DUT shall not return any fault if GetAuthenticationProfileInfo was invoked for non-exciting authentication profile token. Such tokens shall be ignored.

• DUT shall return SOAP 1.2 fault message (InvalidArgs/TooManyItems) if more items than MaxLimit was requested by GetAuthenticationProfileInfo command.

• The following tests are performed
  
  • Getting authentication profile info with GetAuthenticationProfileInfo command
  
  • Getting authentication profile info list with GetAuthenticationProfileInfoList command with using different Limit and NextReference values
  
  • Getting authentication profile info with invalid authentication profile token
  
  • Getting authentication profile info with number of requested items is greater than MaxLimit

Please refer to Section 5.2 for Authentication Profile Info Test Cases.

4.3.3 Authentication Profile

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

• DUT shall give the Authentication Behavior Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.

• DUT shall support the following commands:
  
  • GetAuthenticationProfiles
  
  • GetAuthenticationProfileList
  
  • GetAuthenticationProfileInfoList
  
  • CreateAuthenticationProfile
  
  • DeleteAuthenticationProfile
  
  • CreateSecurityLevel
  
  • GetSecurityLevelInfoList
  
  • DeleteSecurityLevel

• DUT shall return only requested items in GetAuthenticationProfiles response that specified in GetAuthenticationProfiles request.
• DUT shall return all requested items in GetAuthenticationProfiles response that specified in GetAuthenticationProfiles request.

• DUT shall not return more items in GetAuthenticationProfiles responses than specified in service capabilities by MaxLimit.

• DUT shall return the same information in GetAuthenticationProfiles responses and in GetAuthenticationProfileInfoList responses for the items with the same token.

• DUT shall not return more items in GetAuthenticationProfileList response than specified by Limit parameter in a request.

• DUT shall not return items with the same tokens in GetAuthenticationProfileList responses for one authentication profile list resieving.

• DUT shall return the same information in GetAuthenticationProfiles responses and in GetAuthenticationProfileList responses for the items with the same token.

• DUT shall return the same information in GetAuthenticationProfileList responses and in GetAuthenticationProfileInfoList responses for the items with the same token.

• DUT shall return the same authentication profiles in GetAuthenticationProfileList responses and in GetAuthenticationProfileInfoList responses.

• DUT shall return SOAP 1.2 fault message (InvalidArgs/TooManyItems) if more items than MaxLimit was requested by GetAuthenticationProfiles command.

• The following tests are performed
  
  • Getting authentication profile with GetSchedule command and test that it includes the same information with GetAuthenticationProfileInfoList command
  
  • Getting authentication profile info list with GetAuthenticationProfileList command with using different Limit and NextReference values and test that it includes the same information with GetAuthenticationProfileInfoList command
  
  • Getting authentication profiles with invalid authentication profile token
  
  • Getting authentication profiles with number of requested items is greater than MaxLimit

Please refer to Section 5.3 for Authentication Profile Test Cases.

4.3.4 Authentication Profile Management

The test policies specific to the test case execution of Authentication Profile Management functional block:
• DUT shall give the Authentication Behavior Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.

• DUT shall support the following commands and notification topics:

  • GetAuthenticationProfiles
  • GetAuthenticationProfileInfo
  • GetAuthenticationProfileList
  • GetAuthenticationProfileInfoList
  • CreateAuthenticationProfile
  • ModifyAuthenticationProfile
  • SetAuthenticationProfile
  • DeleteAuthenticationProfile
  • tns1:Configuration/AuthenticationProfile/Changed
  • tns1:Configuration/AuthenticationProfile/Removed
  • tns1:Configuration/SecurityLevel/Changed
  • tns1:Configuration/SecurityLevel/Removed
  • CreateSecurityLevel
  • GetSecurityLevelInfoList
  • DeleteSecurityLevel
  • GetScheduleInfoList
  • GetServiceCapabilities (Schedule Service)
  • CreateSchedule
  • DeleteSchedule

• The DUT shall support creation of authentication profile with sending tns1:Configuration/AuthenticationProfile/Changed notification.

• The DUT shall support modification of authentication profile with sending tns1:Configuration/AuthenticationProfile/Changed notification.
• The DUT shall support deletion of authentication profile with sending tns1:Configuration/AuthenticationProfile/Removed notification.

• DUT shall return SOAP 1.2 fault message (InvalidArgVal) if authentication profile token is specified in CreateAuthenticationProfile request.

• DUT should return SOAP 1.2 fault message (InvalidArgVal/NotFound) if ModifyAuthenticationProfile or DeleteAuthenticationProfile command was invoked for non-exciting authentication profile token.

• DUT should return SOAP 1.2 fault message (CapabilityViolated/MaxPoliciesPerAuthenticationProfile) if MaxPoliciesPerAuthenticationProfile capability was violated for CreateAuthenticationProfile or ModifyAuthenticationProfile command.

• If DUT supports token supplying as indicated by ClientSuppliedTokenSupported capability:
  • The DUT shall support creation or update of authentication profile by set command with sending tns1:Configuration/AuthenticationProfile/Changed notification.
  • DUT shall return SOAP 1.2 fault message (InvalidArgVal) if authentication profile token is not specified in SetAuthenticationProfile request.
  • DUT should return SOAP 1.2 fault message (CapabilityViolated/MaxPoliciesPerAuthenticationProfile) if MaxPoliciesPerAuthenticationProfile capability was violated for SetAuthenticationProfile command.

• The following tests are performed:
  • Creating authentication profile with CreateAuthenticationProfile command with empty token and test that corresponding notification message is received:
    • without any authentication policies
    • with authentication policies
  • Modifying authentication profile with ModifyAuthenticationProfile command and test that corresponding notification message is received
  • Deleting authentication profile with DeleteAuthenticationProfile command and test that corresponding notification message is received
  • Creating authentication profile with CreateAuthenticationProfile command with specified token
• Creating authentication profile with CreateAuthenticationProfile command with maximum number of security policies and with violated MaxPoliciesPerAuthenticationProfile capability

• Modifying authentication profile with ModifyAuthenticationProfile command with invalid token

• Modifying authentication profile with ModifyAuthenticationProfile command with maximum number of security policies and with violated MaxPoliciesPerAuthenticationProfile capability

• Deleting authentication profile with DeleteAuthenticationProfile command with invalid token

• Deleting authentication profile with DeleteAuthenticationProfile command with empty token

• If DUT supports token supplying as indicated by ClientSuppliedTokenSupported capability:
  • Creating authentication profile with SetAuthenticationProfile command with empty token and test that corresponding notification message is received:
    • without any authentication policies
    • with authentication policies
  • Modifying authentication profile with SetAuthenticationProfile command and test that corresponding notification message is received
  • Setting authentication profile with SetAuthenticationProfile command with maximum number of security policies and with violated MaxPoliciesPerAuthenticationProfile capability
  • Setting authentication profile with SetAuthenticationProfile command with empty token

Please refer to Section 5.4 for Authentication Profile Management Test Cases.

### 4.3.5 Security Level Info

The test policies specific to the test case execution of Security Level Info functional block:

• DUT shall give the Authentication Behavior 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
  • GetSecurityLevelInfo
  • GetSecurityLevelInfoList
  • CreateSecurityLevel
  • DeleteSecurityLevel
  • DUT shall not return more items in GetSecurityLevelInfo and GetSecurityLevelInfoList responses than specified in service capabilities by MaxLimit.
  • DUT shall not return more items in GetSecurityLevelInfoList response than specified by Limit parameter in a request.
  • DUT shall not return items with the same tokens in GetSecurityLevelInfoList responses for one security level info list receiving.
  • DUT shall not return more SecurityLevelInfo items in GetSecurityLevelInfoList responses than specified in service capabilities by MaxSecurityLevels.
  • DUT shall not return any fault if GetSecurityLevelInfo was invoked for non-exciting security level token. Such tokens shall be ignored.
  • DUT shall return SOAP 1.2 fault message (InvalidArgs/TooManyItems) if more items than MaxLimit was requested by GetSecurityLevelInfo command.

  • The following tests are performed
    • Getting security level info with GetSecurityLevelInfo command
    • Getting security level info list with GetSecurityLevelInfoList command with using different Limit and NextReference values
    • Getting security level info with invalid security level token
    • Getting security level info with number of requested items is greater than MaxLimit

Please refer to Section 5.5 for Security Level Info Test Cases.

4.3.6 Security Level

The test policies specific to the test case execution of Security Level functional block:
• DUT shall give the Authentication Behavior Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.

• DUT shall support the following commands:
  • GetSecurityLevels
  • GetSecurityLevelList
  • GetSecurityLevelInfoList
  • CreateSecurityLevel
  • DeleteSecurityLevel

• DUT shall return only requested items in GetSecurityLevels response that specified in GetSecurityLevels request.

• DUT shall return all requested items in GetSecurityLevels response that specified in GetSecurityLevels request.

• DUT shall not return more items in GetSecurityLevels responses than specified in service capabilities by MaxLimit.

• DUT shall return the same information in GetSecurityLevels responses and in GetSecurityLevelInfoList responses for the items with the same token.

• DUT shall not return more items in GetSecurityLevelList response than specified by Limit parameter in a request.

• DUT shall not return items with the same tokens in GetSecurityLevelList responses for one security level list resieving.

• DUT shall return the same information in GetSecurityLevels responses and in GetSecurityLevelList responses for the items with the same token.

• DUT shall return the same information in GetSecurityLevelList responses and in GetSecurityLevelInfoList responses for the items with the same token.

• DUT shall return the same security levels in GetSecurityLevelList responses and in GetSecurityLevelInfoList responses.

• DUT shall return SOAP 1.2 fault message (InvalidArgs/TooManyItems) if more items than MaxLimit was requested by GetSecurityLevels command.

• The following tests are performed
• Getting security level with GetSchedule command and test that it includes the same information with GetSecurityLevelInfoList command

• Getting security level info list with GetSecurityLevelList command with using different Limit and NextReference values and test that it includes the same information with GetSecurityLevelInfoList command

• Getting security levels with invalid security level token

• Getting security levels with number of requested items is greater than MaxLimit

Please refer to Section 5.6 for Security Level Test Cases.

4.3.7 Security Level Management

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

• DUT shall give the Authentication Behavior Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.

• DUT shall support the following commands and notification topics:
  • GetSecurityLevels
  • GetSecurityLevelInfo
  • GetSecurityLevelList
  • GetSecurityLevelInfoList
  • CreateSecurityLevel
  • ModifySecurityLevel
  • SetSecurityLevel
  • DeleteSecurityLevel
  • tns1:Configuration/SecurityLevel/Changed
  • tns1:Configuration/SecurityLevel/Removed
  • tns1:Configuration/SecurityLevel/Changed
  • tns1:Configuration/SecurityLevel/Removed
• The DUT shall support creation of security level with sending tns1:Configuration/SecurityLevel/Changed notification.

• The DUT shall support modification of security level with sending tns1:Configuration/SecurityLevel/Changed notification.

• The DUT shall support deletion of security level with sending tns1:Configuration/SecurityLevel/Removed notification.

• DUT shall return SOAP 1.2 fault message (InvalidArgVal) if security level token is specified in CreateSecurityLevel request.

• DUT should return SOAP 1.2 fault message (InvalidArgVal/DuplicatePriority) if duplicated priority is specified in CreateSecurityLevel or ModifySecurityLevel request.

• DUT should return SOAP 1.2 fault message (InvalidArgVal/NotFound) if ModifySecurityLevel or DeleteSecurityLevel command was invoked for non-exciting security level token token.

• DUT should return SOAP 1.2 fault message (CapabilityViolated/MaxRecognitionGroupsPerSecurityLevel) if MaxRecognitionGroupsPerSecurityLevel capability was violated for CreateSecurityLevel or ModifySecurityLevel command.

• DUT should return SOAP 1.2 fault message (CapabilityViolated/MaxRecognitionMethodsPerRecognitionGroup) if MaxRecognitionMethodsPerRecognitionGroup capability was violated for CreateSecurityLevel or ModifySecurityLevel command.

• If DUT supportes token supplying as indicated by ClientSuppliedTokenSupported capability:

  • The DUT shall support creation or update of security level by set command with sending tns1:Configuration/SecurityLevel/Changed notification.

  • DUT shall return SOAP 1.2 fault message (InvalidArgVal) if security level token is not specified in SetSecurityLevel request.

  • DUT should return SOAP 1.2 fault message (InvalidArgVal/DuplicatePriority) if duplicated priority is specified in SetSecurityLevel request.

  • DUT should return SOAP 1.2 fault message (CapabilityViolated/MaxRecognitionGroupsPerSecurityLevel) if MaxRecognitionGroupsPerSecurityLevel capability was violated for SetSecurityLevel command.

  • DUT should return SOAP 1.2 fault message (CapabilityViolated/MaxRecognitionMethodsPerRecognitionGroup) if MaxRecognitionMethodsPerRecognitionGroup capability was violated for SetSecurityLevel command.
• The following tests are performed:

• Creating security level with CreateSecurityLevel command with empty token and test that corresponding notification message is received:
  • without any recognition groups
  • without any recognition methods
  • with any recognition methods

• Modifying security level with ModifySecurityLevel command and test that corresponding notification message is received

• Deleting security level with DeleteSecurityLevel command and test that corresponding notification message is received

• Creating security level with CreateSecurityLevel command with specified token

• Creating security level with CreateSecurityLevel command with maximum number of recognition groups and with violated MaxRecognitionGroupsPerSecurityLevel capability

• Creating security level with CreateSecurityLevel command with maximum number of recognition methods and with violated MaxRecognitionMethodsPerRecognitionGroup capability

• Creating security level with CreateSecurityLevel command with duplicated priority

• Modifying security level with ModifySecurityLevel command with invalid token

• Modifying security level with ModifySecurityLevel command with maximum number of recognition groups and with violated MaxRecognitionGroupsPerSecurityLevel capability

• Modifying security level with ModifySecurityLevel command with maximum number of recognition methods and with violated MaxRecognitionMethodsPerRecognitionGroup capability

• Modifying security level with ModifySecurityLevel command with duplicated priority

• Deleting security level with DeleteSecurityLevel command with invalid token

• Deleting security level with DeleteSecurityLevel command with empty token

• If DUT supports token supplying as indicated by ClientSuppliedTokenSupported capability:
• Creating security level with SetSecurityLevel command with empty token and test that corresponding notification message is received:
  • without any recognition groups
  • without any recognition methods
  • with any recognition methods
• Modifying security level with SetSecurityLevel command and test that corresponding notification message is received
• Setting security level with SetSecurityLevel command with maximum number of recognition groups and with violated MaxRecognitionGroupsPerSecurityLevel capability
• Setting security level with SetSecurityLevel command with maximum number of recognition methods and with violated MaxRecognitionMethodsPerRecognitionGroup capability
• Setting security level with SetSecurityLevel command with duplicated priority
• Setting security level with SetSecurityLevel command with empty token

Please refer to Section 5.7 for Security Level Management Test Cases.

4.3.8 Authentication Behavior Events

The test policies specific to the test case execution of Authentication Behavior Events functional block:

• DUT shall give the Authentication Behavior Service and Event Service entry points by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.

• DUT shall support the following commands and notification topics:
  • GetServices
  • GetEventProperties
  • tns1:Configuration/AuthenticationProfile/Changed
  • tns1:Configuration/AuthenticationProfile/Removed
  • tns1:Configuration/SecurityLevel/Changed
• tns1:Configuration/SecurityLevel/Removed

• The following tests are performed

• Getting event properties with GetEventProperties command for the following notification topics:
  • tns1:Configuration/AuthenticationProfile/Changed
  • tns1:Configuration/AuthenticationProfile/Removed
  • tns1:Configuration/SecurityLevel/Changed
  • tns1:Configuration/SecurityLevel/Removed

Please refer to Section 5.8 for Authentication Behavior Events Test Cases.
5 Authentication Profile Test Cases

5.1 Capabilities

5.1.1 AUTHENTICATION BEHAVIOR SERVICE CAPABILITIES

Test Case ID: AUTH_BEHAVIOR-1-1-1

Specification Coverage: ServiceCapabilities (ONVIF Authentication Behavior Service Specification), GetServiceCapabilities command (ONVIF Authentication Behavior Service Specification)

Feature Under Test: GetServiceCapabilities (for Authentication Behavior Service)

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify that Authentication Behavior Service is received using GetServices request, to verify DUT Authentication Behavior Service Capabilities, and to verify Get Services and Authentication Behavior Service Capabilities consistency.

Pre-Requisite: Authentication Behavior 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/authenticationbehavior/wsdl", FAIL the test, restore the DUT state, and skip other steps.

6. Set authServ := item from listOfServicesWithoutCapabilities list with Namespace = "http://www.onvif.org/ver10/authenticationbehavior/wsdl".
7. If authServ.Capabilities is specified, FAIL the test, restore the DUT state, and skip other steps.

8. ONVIF Client invokes GetServices with parameters
   - IncludeCapability := true

9. The DUT responds with a GetServicesResponse message with parameters
   - Services list := servicesList

    - Services list [Namespace = "http://www.onvif.org/ver10/authenticationbehavior/wsdI"] := authServ

11. ONVIF Client invokes GetServiceCapabilities.

12. The DUT responds with a GetServiceCapabilitiesResponse message with parameters
    - Capabilities := cap

13. If cap differs from authServ.Capabilities.Capabilities, FAIL the test, restore the DUT state, and skip other steps.

Procedure Result:

PASS –
- The DUT passed all assertions.

FAIL –
- The DUT did not send GetServiceCapabilitiesResponse message.
- The DUT did not send GetServicesResponse message.

Note: The following fields are compared at step 13:
- MaxLimit
- MaxAuthenticationProfiles
- MaxPoliciesPerAuthenticationProfile
- MaxSecurityLevels
- MaxRecognitionGroupsPerSecurityLevel
5.2 Authentication Profile Info

5.2.1 GET AUTHENTICATION PROFILE INFO

Test Case ID: AUTH_BEHAVIOR-2-1-1


Feature Under Test: GetAuthenticationProfileInfo

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify Get Authentication Profile Info.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client creates number of authentication profiles by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out authProfileInfoCompleteList - complete list of authentication profiles information
   - out createdAuthTokenedList - list of created authentication profiles tokens
   - out securityLevelToken - created security level token (if any)
   - out cap - Authentication Behavior Service capabilities

4. Set tokenList := [subset of authProfileInfoCompleteList.token values with items number equal to cap.MaxLimit]

5. ONVIF client invokes GetAuthenticationProfileInfo with parameters
• Token list := tokenList

6. The DUT responds with GetAuthenticationProfileInfoResponse message with parameters

• AuthenticationProfileInfo list =: authProfileInfoList1

7. If authProfileInfoList1 does not contain AuthenticationProfileInfo item for each token from tokenList, FAIL the test, restore the DUT state, and skip other steps.

8. If authProfileInfoList1 contains at least two AuthenticationProfileInfo items with equal token, FAIL the test, restore the DUT state, and skip other steps.

9. If authProfileInfoList1 contains other AuthenticationProfileInfo items than listed in tokenList, FAIL the test, restore the DUT state, and skip other steps.

10. For each AuthenticationProfileInfo.token token from authProfileInfoCompleteList repeat the following steps:

10.1. ONVIF client invokes GetAuthenticationProfileInfo with parameters

• Token[0] := token

10.2. The DUT responds with GetAuthenticationProfileInfoResponse message with parameters

• AuthenticationProfileInfo list =: authProfileInfoList2

10.3. If authProfileInfoList2 does not contain only one AuthenticationProfileInfo item with token equal to token, FAIL the test, restore the DUT state, and skip other steps.

10.4. If authProfileInfoList2[0] item is not equal to authProfileInfoCompleteList[token = token] item, FAIL the test, restore the DUT state, and skip other steps.

11. Remove all authentication profiles with tokens from createdAuthProfileTokensList.

12. If securityLevelToken is specified:

12.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

• in securityLevelToken - security level token

Test Result:

PASS –
• The DUT passed all assertions.

FAIL –

• The DUT did not send GetAuthenticationProfileInfoResponse message.

Note: If number of items in authProfileInfoCompleteList is less than cap.MaxLimit, then all authProfileInfoCompleteList.Token items shall be used for the step 4.

Note: The following fields are compared at step 10.4:

• AuthenticationProfileInfo:

  • token
  • Name
  • Description

5.2.2 GET AUTHENTICATION PROFILE INFO LIST - LIMIT

Test Case ID: AUTH_BEHAVIOR-2-1-2


Feature Under Test: GetAuthenticationProfileInfoList

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify Get Authentication Profile Info List using Limit.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client creates number of authentication profiles by following the procedure mentioned in Annex A.4 with the following input and output parameters

   • out authProfileInfoCompleteList - complete list of authentication profiles information
4. ONVIF client invokes \texttt{GetAuthenticationProfileInfoList} with parameters
   \begin{itemize}
   \item Limit := 1
   \item StartReference skipped
   \end{itemize}

5. The DUT responds with \texttt{GetAuthenticationProfileInfoListResponse} message with parameters
   \begin{itemize}
   \item NextStartReference =: nextStartReference
   \item AuthenticationProfileInfo list =: authProfileInfoList1
   \end{itemize}

   6. If \texttt{authProfileInfoList1} contains more AuthenticationProfileInfo items than 1, FAIL the test, restore the DUT state, and skip other steps.

   7. If \texttt{cap.MaxLimit} is equal to 1, go to step 16.

   8. ONVIF client invokes \texttt{GetAuthenticationProfileInfoList} with parameters
   \begin{itemize}
   \item Limit := \texttt{cap.MaxLimit}
   \item StartReference skipped
   \end{itemize}

   9. The DUT responds with \texttt{GetAuthenticationProfileInfoListResponse} message with parameters
   \begin{itemize}
   \item NextStartReference =: nextStartReference
   \item AuthenticationProfileInfo list =: authProfileInfoList2
   \end{itemize}

   10. If \texttt{authProfileInfoList2} contains more AuthenticationProfileInfo items than \texttt{cap.MaxLimit}, FAIL the test, restore the DUT state, and skip other steps.

   11. If \texttt{cap.MaxLimit} is equal to 2, go to step 16.

   12. Set limit := \{number between 1 and \texttt{cap.MaxLimit}\}.

   13. ONVIF client invokes \texttt{GetAuthenticationProfileInfoList} with parameters
   \begin{itemize}
   \item Limit := limit
14. The DUT responds with `GetAuthenticationProfileInfoListResponse` message with parameters
   - `NextStartReference` = `nextStartReference`
   - `AuthenticationProfileInfo list` = `authProfileInfoList3`

15. If `authProfileInfoList3` contains more `AuthenticationProfileInfo` items than `limit`, FAIL the test, restore the DUT state, and skip other steps.

16. Remove all authentication profiles with tokens from `createdAuthProfileTokensList`.

17. If `securityLevelToken` is specified:
   17.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters
       - In `securityLevelToken` - security level token

Test Result:

PASS –
   - The DUT passed all assertions.

FAIL –
   - The DUT did not send `GetAuthenticationProfileInfoListResponse` message.

5.2.3 GET AUTHENTICATION PROFILE INFO LIST - START REFERENCE AND LIMIT

**Test Case ID**: AUTH_BEHAVIOR-2-1-3

**Specification Coverage**: AuthenticationProfileInfo (ONVIF Authentication Behavior Service Specification), GetAuthenticationProfileInfoList command (ONVIF Authentication Behavior Service Specification)

**Feature Under Test**: GetAuthenticationProfileInfoList

**WSDL Reference**: authenticationbehavior.wsdl

**Test Purpose**: To verify Get Authentication Profile Info List using StartReference and Limit.

**Pre-Requisite**: Authentication Behavior Service is received from the DUT.
Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client creates number of authentication profiles by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out authProfileInfoCompleteList - complete list of authentication profiles information
   - out createdAuthProfileTokensList - list of created authentication profiles tokens
   - out securityLevelToken - created security level token (if any)
   - out cap - Authentication Behavior Service capabilities

4. ONVIF client invokes GetAuthenticationProfileInfoList with parameters
   - Limit := cap.MaxLimit
   - StartReference skipped

5. The DUT responds with GetAuthenticationProfileInfoListResponse message with parameters
   - NextStartReference := nextStartReference
   - AuthenticationProfileInfo list := authProfileInfoCompleteList1

6. If authProfileInfoCompleteList1 contains more AuthenticationProfileInfo items than cap.MaxLimit, FAIL the test, restore the DUT state, and skip other steps.

7. Until nextStartReference is not null, repeat the following steps:
   7.1. ONVIF client invokes GetAuthenticationProfileInfoList with parameters
      - Limit := cap.MaxLimit
      - StartReference := nextStartReference
   7.2. The DUT responds with GetAuthenticationProfileInfoListResponse message with parameters
      - NextStartReference := nextStartReference
      - AuthenticationProfileInfo list := authProfileInfoListPart
7.3. If \( \text{authProfileInfoListPart} \) contains more AuthenticationProfileInfo items than \( \text{cap.MaxLimit} \), FAIL the test, restore the DUT state, and skip other steps.

7.4. Set \( \text{authProfileInfoCompleteList1} := \text{authProfileInfoCompleteList1} + \text{authProfileInfoListPart} \)

8. If \( \text{authProfileInfoCompleteList1} \) contains at least two AuthenticationProfileInfo item with equal token, FAIL the test, restore the DUT state, and skip other steps.

9. If \( \text{cap.MaxLimit} \) is equal to 1, go to step 26.

10. ONVIF client invokes \text{GetAuthenticationProfileInfoList} with parameters

    \begin{itemize}
    \item \text{Limit} := 1
    \item \text{StartReference} skipped
    \end{itemize}

11. The DUT responds with \text{GetAuthenticationProfileInfoListResponse} message with parameters

    \begin{itemize}
    \item \text{NextStartReference} =: \text{nextStartReference}
    \item \text{AuthenticationProfileInfo list} =: \text{authProfileInfoCompleteList2}
    \end{itemize}

12. If \( \text{authProfileInfoCompleteList2} \) contains more AuthenticationProfileInfo items than 1, FAIL the test, restore the DUT state, and skip other steps.

13. Until \text{nextStartReference} is not null, repeat the following steps:

    \begin{enumerate}
    \item ONVIF client invokes \text{GetAuthenticationProfileInfoList} with parameters
    \begin{itemize}
    \item \text{Limit} := 1
    \item \text{StartReference} := \text{nextStartReference}
    \end{itemize}
    \item The DUT responds with \text{GetAuthenticationProfileInfoListResponse} message with parameters
    \begin{itemize}
    \item \text{NextStartReference} =: \text{nextStartReference}
    \item \text{AuthenticationProfileInfo list} =: \text{authProfileInfoListPart}
    \end{itemize}
    \item If \( \text{authProfileInfoListPart} \) contains more AuthenticationProfileInfo items than 1, FAIL the test, restore the DUT state, and skip other steps.
    \item Set \( \text{authProfileInfoCompleteList2} := \text{authProfileInfoCompleteList2} + \text{authProfileInfoListPart} \)
    \end{enumerate}
14. If `authProfileInfoCompleteList2` contains at least two `AuthenticationProfileInfo` item with equal token, FAIL the test, restore the DUT state, and skip other steps.

15. If `authProfileInfoCompleteList2` does not contain all authentication profiles from `authProfileInfoCompleteList1`, FAIL the test, restore the DUT state, and skip other steps.

16. If `authProfileInfoCompleteList2` contains authentication profiles other than authentication profiles from `authProfileInfoCompleteList1`, FAIL the test, restore the DUT state, and skip other steps.

17. If `cap.MaxLimit` is equal to 2, go to step 26.

18. Set `limit :=` [number between 1 and `cap.MaxLimit`]

19. ONVIF client invokes `GetAuthenticationProfileInfoList` with parameters

   - `Limit := limit`
   - `StartReference` skipped

20. The DUT responds with `GetAuthenticationProfileInfoListResponse` message with parameters

   - `NextStartReference =: nextStartReference`
   - `AuthenticationProfileInfo list =: authProfileInfoCompleteList3`

21. If `authProfileInfoCompleteList3` contains more `AuthenticationProfileInfo` items than `limit`, FAIL the test, restore the DUT state, and skip other steps.

22. Until `nextStartReference` is not null, repeat the following steps:

   22.1. ONVIF client invokes `GetAuthenticationProfileInfoList` with parameters

      - `Limit := limit`
      - `StartReference := nextStartReference`

   22.2. The DUT responds with `GetAuthenticationProfileInfoListResponse` message with parameters

      - `NextStartReference =: nextStartReference`
      - `AuthenticationProfileInfo list =: authProfileInfoListPart`

   22.3. If `authProfileInfoListPart` contains more `AuthenticationProfileInfo` items than `limit`, FAIL the test, restore the DUT state, and skip other steps.
22.4. Set $authProfileInfoCompleteList3 := authProfileInfoCompleteList3 + authProfileInfoListPart$

23. If $authProfileInfoCompleteList3$ contains at least two AuthenticationProfileInfo item with equal token, FAIL the test, restore the DUT state, and skip other steps.

24. If $authProfileInfoCompleteList3$ does not contain all authentication profiles from $authProfileInfoCompleteList1$, FAIL the test, restore the DUT state, and skip other steps.

25. If $authProfileInfoCompleteList3$ contains authentication profiles other than authentication profiles from $authProfileInfoCompleteList1$, FAIL the test, restore the DUT state, and skip other steps.

26. Remove all authentication profiles with tokens from $createdAuthProfileTokensList$.

27. If $securityLevelToken$ is specified:

   27.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters
   
   • in $securityLevelToken$ - security level token

Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send $GetAuthenticationProfileInfoListResponse$ message.

5.2.4 GET AUTHENTICATION PROFILE INFO LIST - NO LIMIT

Test Case ID: AUTH_BEHAVIOR-2-1-4


Feature Under Test: GetAuthenticationProfileInfoList

WSDL Reference: authenticationbehavior.wsdl
Test Purpose: To verify Get Authentication Profile Info List without using Limit.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client creates number of authentication profiles by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out authProfileInfoCompleteList - complete list of authentication profiles information
   - out createdAuthProfileTokensList - list of created authentication profiles tokens
   - out securityLevelToken - created security level token (if any)
   - out cap - Authentication Behavior Service capabilities

4. ONVIF client invokes GetAuthenticationProfileInfoList with parameters
   - Limit skipped
   - StartReference skipped

5. The DUT responds with GetAuthenticationProfileInfoListResponse message with parameters
   - NextStartReference := nextStartReference
   - AuthenticationProfileInfo list := authProfileInfoCompleteList

6. If authProfileInfoCompleteList contains more AuthenticationProfileInfo items than cap.MaxLimit, FAIL the test, restore the DUT state, and skip other steps.

7. Until nextStartReference is not null, repeat the following steps:
   7.1. ONVIF client invokes GetAuthenticationProfileInfoList with parameters
         - Limit skipped
         - StartReference := nextStartReference
   7.2. The DUT responds with GetAuthenticationProfileInfoListResponse message with parameters
• NextStartReference =: nextStartReference

• AuthenticationProfileInfo list =: authProfileInfoListPart

7.3. If authProfileInfoListPart contains more AuthenticationProfileInfo items than cap.MaxLimit, FAIL the test, restore the DUT state, and skip other steps.

7.4. Set authProfileInfoCompleteList := authProfileInfoCompleteList + authProfileInfoListPart

8. If authProfileInfoCompleteList contains at least two AuthenticationProfileInfo item with equal token, FAIL the test, restore the DUT state, and skip other steps.

9. If authProfileInfoCompleteList contains more AuthenticationProfileInfo items than cap.MaxAuthenticationProfiles, FAIL the test, restore the DUT state, and skip other steps.

10. Remove all authentication profiles with tockens from createdAuthProfileTokensList.

11. If securityLevelToken is specified:

11.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

• in securityLevelToken - security level token

Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send GetAuthenticationProfileInfoListResponse message.

5.2.5 GET AUTHENTICATION PROFILE INFO WITH INVALID TOKEN

Test Case ID: AUTH_BEHAVIOR-2-1-5


Feature Under Test: GetAuthenticationProfileInfo
**WSDL Reference:** authenticationbehavior.wsdl

**Test Purpose:** To verify Get Authentication Profile Info with invalid token.

**Pre-Requisite:** Authentication Behavior Service is received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates number of authentication profiles by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out authProfileInfoCompleteList - complete list of authentication profiles information
   - out createdAuthProfileTokensList - list of created authentication profiles tokens
   - out securityLevelToken - created security level token (if any)
   - out cap - Authentication Behavior Service capabilities
4. Set invalidToken := value not equal to any authProfileInfoCompleteList.token
5. ONVIF client invokes GetAuthenticationProfileInfo with parameters
   - Token list := invalidToken
6. The DUT responds with GetAuthenticationProfileInfoResponse message with parameters
   - AuthenticationProfileInfo list =: authProfileInfoList
7. If authProfileInfoList is not empty, FAIL the test, restore the DUT state, and skip other steps.
8. If cap.MaxLimit is less than 2, go to step 14.
9. ONVIF client invokes GetAuthenticationProfileInfo with parameters
   - Token[0] := invalidToken
   - Token[1] := authProfileInfoCompleteList[0].token
10. The DUT responds with GetAuthenticationProfileInfoResponse message with parameters
• AuthenticationProfileInfo list = authProfileInfoList

11. If authProfileInfoList is empty, FAIL the test, restore the DUT state, and skip other steps.

12. If authProfileInfoList contains more than one item, FAIL the test, restore the DUT state, and skip other steps.

13. If authProfileInfoList[0].token is not equal to authProfileInfoCompleteList[0].token, FAIL the test, restore the DUT state, and skip other steps.

14. Remove all authentication profiles with tokens from createdAuthProfileTokensList.

15. If securityLevelToken is specified:

   15.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

       • in securityLevelToken - security level token

Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send GetAuthenticationProfileInfoResponse message.

5.2.6 GET AUTHENTICATION PROFILE INFO - TOO MANY ITEMS

Test Case ID: AUTH_BEHAVIOR-2-1-6


Feature Under Test: GetAuthenticationProfileInfo

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify Get Authentication Profile Info in case there are more items than MaxLimit in request.
Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client creates number of authentication profiles by following the procedure mentioned in Annex A.4 with the following input and output parameters
   • out authProfileInfoCompleteList - complete list of authentication profiles information
   • out createdAuthProfileTokensList - list of created authentication profiles tokens
   • out securityLevelToken - created security level token (if any)
   • out cap - Authentication Behavior Service capabilities

4. If authProfileInfoCompleteList.token items number is less than cap.MaxLimit or equal to cap.MaxLimit, go to step 8.

5. Set tokenList := [subset of authProfileInfoCompleteList.token values with items number equal to cap.MaxLimit + 1]

6. ONVIF client invokes GetAuthenticationProfileInfo with parameters
   • Token list := tokenList


8. Remove all authentication profiles with tokens from createdAuthProfileTokensList.

9. If securityLevelToken is specified:

   9.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters
      • in securityLevelToken - security level token

Test Result:

PASS –

• The DUT passed all assertions.
FAIL –

- The DUT did not send env:Sender/ter:InvalidArgs/ter:TooManyItems SOAP 1.2 fault.

5.3 Authentication Profile

5.3.1 GET AUTHENTICATION PROFILES

Test Case ID: AUTH_BEHAVIOR-3-1-1


Feature Under Test: GetAuthenticationProfiles

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify Get Authentication Profile.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates number of authentication profiles by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out authProfileInfoCompleteList - complete list of authentication profiles information
   - out createdAuthProfileTokensList - list of created authentication profiles tokens
   - out securityLevelToken - created security level token (if any)
   - out cap - Authentication Behavior Service capabilities
4. ONVIF Client retrieves a complete list of authentication profile info by following the procedure mentioned in Annex A.3 with the following input and output parameters
   - out authProfileCompleteList - complete list of authentication profiles information
5. Set $tokenList := \{\text{subset of } authProfileCompleteList\text{.token values with items number equal to } cap\text{.MaxLimit}\}$.

6. ONVIF client invokes $\text{GetAuthenticationProfiles}$ with parameters
   
   $\begin{itemize}
   \item Token list := $tokenList$
   \end{itemize}$

7. The DUT responds with $\text{GetAuthenticationProfilesResponse}$ message with parameters
   
   $\begin{itemize}
   \item AuthenticationProfile list =: authProfilesList1
   \end{itemize}$

8. If $authProfilesList1$ does not contain Authentication Profile item for each token from $tokenList$, FAIL the test, restore the DUT state, and skip other steps.

9. If $authProfilesList1$ contains at least two Authentication Profile items with equal token, FAIL the test, restore the DUT state, and skip other steps.

10. If $authProfilesList1$ contains other Authentication Profile items than listed in $tokenList$, FAIL the test, restore the DUT state, and skip other steps.

11. For each AuthenticationProfile.token $token$ from $authProfileCompleteList$ repeat the following steps:
   
   11.1. ONVIF client invokes $\text{GetAuthenticationProfiles}$ with parameters
   
   $\begin{itemize}
   \item Token[0] := $token$
   \end{itemize}$

   11.2. The DUT responds with $\text{GetAuthenticationProfilesResponse}$ message with parameters
   
   $\begin{itemize}
   \item AuthenticationProfile list =: authProfilesList2
   \end{itemize}$

   11.3. If $authProfilesList2$ does not contain only one AuthenticationProfile item with token equal to $token$, FAIL the test, restore the DUT state, and skip other steps.

   11.4. If $authProfilesList2[0]$ item does not have equal field values to $authProfileCompleteList[token = token]$ item, FAIL the test, restore the DUT state, and skip other steps.

12. Remove all authentication profiles with tokens from $createdAuthProfileTokensList$.

13. If $securityLevelToken$ is specified:
   
   13.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters
   
   $\begin{itemize}
   \item in $securityLevelToken$ - security level token
   \end{itemize}$
Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send GetAuthenticationProfilesResponse message.

Note: If the number of items in authProfileCompleteList is less than cap.MaxLimit, then all authProfileCompleteList.Token items shall be used for the step 5.

Note: The following fields are compared at step 11.4:

• AuthenticationProfile:
  • token
  • Name
  • Description
  • DefaultSecurityLevelToken

• AuthenticationPolicy list
  • ScheduleToken
  • SecurityLevelConstraint list
    • ActiveRegularSchedule
    • ActiveSpecialDaySchedule
  • AuthenticationMode list
  • SecurityLevelToken

5.3.2 GET AUTHENTICATION PROFILE LIST - LIMIT

Test Case ID: AUTH_BEHAVIOR-3-1-2


Feature Under Test: GetAuthenticationProfileList
WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify Get Authentication Profile List using Limit.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client creates number of authentication profiles by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out authProfileInfoCompleteList - complete list of authentication profiles information
   - out createdAuthProfileTokensList - list of created authentication profiles tokens
   - out securityLevelToken - created security level token (if any)
   - out cap - Authentication Behavior Service capabilities

4. ONVIF client invokes GetAuthenticationProfileList with parameters
   - Limit := 1
   - StartReference skipped

5. The DUT responds with GetAuthenticationProfileListResponse message with parameters
   - NextStartReference =: nextStartReference
   - AuthenticationProfile list =: authProfilesList1

6. If authProfilesList1 contains more AuthenticationProfile items than 1, FAIL the test, restore
   the DUT state, and skip other steps.

7. If cap.MaxLimit is equal to 1, go to step 16.

8. ONVIF client invokes GetAuthenticationProfileList with parameters
   - Limit := cap.MaxLimit
   - StartReference skipped
9. The DUT responds with `GetAuthenticationProfileListResponse` message with parameters
   - `NextStartReference := nextStartReference`
   - `AuthenticationProfile list := authProfilesList2`
10. If `authProfilesList2` contains more AuthenticationProfile items than `cap.MaxLimit`, FAIL the test, restore the DUT state, and skip other steps.
11. If `cap.MaxLimit` is equal to 2, go to step 16.
12. Set `limit := [number between 1 and cap.MaxLimit]`
13. ONVIF client invokes `GetAuthenticationProfileList` with parameters
   - `Limit := limit`
   - `StartReference skipped`
14. The DUT responds with `GetAuthenticationProfileListResponse` message with parameters
   - `NextStartReference := nextStartReference`
   - `AuthenticationProfile list := authProfilesList3`
15. If `authProfilesList3` contains more AuthenticationProfile items than `limit`, FAIL the test, restore the DUT state, and skip other steps.
16. Remove all authentication profiles with tokens from `createdAuthProfileTokensList`.
17. If `securityLevelToken` is specified:
   17.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters
       - `in securityLevelToken` - security level token

**Test Result:**

PASS –
   - The DUT passed all assertions.

FAIL –
   - The DUT did not send `GetAuthenticationProfileListResponse` message.
5.3.3 GET AUTHENTICATION PROFILE LIST - START REFERENCE AND LIMIT

Test Case ID: AUTH_BEHAVIOR-3-1-3


Feature Under Test: GetAuthenticationProfileList

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify Get Authentication Profile List using StartReference and Limit.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client creates number of authentication profiles by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out authProfileInfoCompleteList - complete list of authentication profiles information
   - out createdAuthProfileTokensList - list of created authentication profiles tokens
   - out securityLevelToken - created security level token (if any)
   - out cap - Authentication Behavior Service capabilities

4. ONVIF client invokes GetAuthenticationProfileList with parameters
   - Limit := cap.MaxLimit
   - StartReference skipped

5. The DUT responds with GetAuthenticationProfileListResponse message with parameters
   - NextStartReference =: nextStartReference
   - AuthenticationProfile list =: authProfileCompleteList1
6. If `authProfileCompleteList1` contains more AuthenticationProfile items than `cap.MaxLimit`, FAIL the test, restore the DUT state, and skip other steps.

7. Until `nextStartReference` is not null, repeat the following steps:

7.1. ONVIF client invokes `GetAuthenticationProfileList` with parameters

   • Limit := `cap.MaxLimit`
   • StartReference := `nextStartReference`

7.2. The DUT responds with `GetAuthenticationProfileListResponse` message with parameters

   • NextStartReference := `nextStartReference`
   • AuthenticationProfile list := `authProfilesListPart`

7.3. If `authProfilesListPart` contains more AuthenticationProfile items than `cap.MaxLimit`, FAIL the test, restore the DUT state, and skip other steps.

7.4. Set `authProfileCompleteList1` := `authProfileCompleteList1` + `authProfilesListPart`.

8. If `authProfileCompleteList1` contains at least two AuthenticationProfile item with equal token, FAIL the test, restore the DUT state, and skip other steps.

9. If `cap.MaxLimit` is equal to 1, do the following steps:

9.1. ONVIF Client compares Authentication Profile List and Authentication Profile Info List by following the procedure mentioned in Annex A.7 with the following input and output parameters

   • in `authProfileCompleteList1` - list of authentication profiles information
   • in `authProfileInfoCompleteList` - list of authentication profiles

9.2. Skip other steps.

10. ONVIF client invokes `GetAuthenticationProfileList` with parameters

    • Limit := 1
    • StartReference skipped

11. The DUT responds with `GetAuthenticationProfileListResponse` message with parameters

    • NextStartReference := `nextStartReference`
ONVIF Authentication Behavior Device Test Specification Version 18.12

- AuthenticationProfile list =: authProfileCompleteList2

12. If authProfileCompleteList2 contains more AuthenticationProfile items than 1, FAIL the test, restore the DUT state, and skip other steps.

13. Until nextStartReference is not null, repeat the following steps:

13.1. ONVIF client invokes GetAuthenticationProfileList with parameters

   - Limit := 1
   - StartReference := nextStartReference

13.2. The DUT responds with GetAuthenticationProfileListResponse message with parameters

   - NextStartReference =: nextStartReference
   - AuthenticationProfile list =: authProfilesListPart

13.3. If authProfilesListPart contains more AuthenticationProfile items than 1, FAIL the test, restore the DUT state, and skip other steps.

13.4. Set authProfileCompleteList2 := authProfileCompleteList2 + authProfilesListPart

14. If authProfileCompleteList2 contains at least two AuthenticationProfile item with equal token, FAIL the test, restore the DUT state, and skip other steps.

15. If authProfileCompleteList2 does not contain all authentication profiles from authProfileCompleteList1, FAIL the test, restore the DUT state, and skip other steps.

16. If authProfileCompleteList2 contains authentication profiles other than authentication profiles from authProfileCompleteList1, FAIL the test, restore the DUT state, and skip other steps.

17. If cap.MaxLimit is equal to 2 do the following steps:

17.1. ONVIF Client compares Authentication Profile List and Authentication Profile Info List by following the procedure mentioned in Annex A.7 with the following input and output parameters

   - in authProfileCompleteList2 - list of authentication profiles information
   - in authProfileInfoCompleteList - list of authentication profiles

17.2. Skip other steps.
18. Set \( \text{limit} := \) [number between 1 and \( \text{cap}.\text{MaxLimit} \)].

19. ONVIF client invokes \texttt{GetAuthenticationProfileList} with parameters

\begin{itemize}
\item \texttt{Limit := limit}
\item \texttt{StartReference skipped}
\end{itemize}

20. The DUT responds with \texttt{GetAuthenticationProfileListResponse} message with parameters

\begin{itemize}
\item \texttt{NextStartReference =: nextStartReference}
\item \texttt{AuthenticationProfile list =: authProfileCompleteList3}
\end{itemize}

21. If \( \text{authProfileCompleteList3} \) contains more AuthenticationProfile items than \( \text{limit} \), FAIL the test, restore the DUT state, and skip other steps.

22. Until \( \text{nextStartReference} \) is not null, repeat the following steps:

22.1. ONVIF client invokes \texttt{GetAuthenticationProfileList} with parameters

\begin{itemize}
\item \texttt{Limit := limit}
\item \texttt{StartReference := nextStartReference}
\end{itemize}

22.2. The DUT responds with \texttt{GetAuthenticationProfileListResponse} message with parameters

\begin{itemize}
\item \texttt{NextStartReference =: nextStartReference}
\item \texttt{AuthenticationProfile list =: authProfilesListPart}
\end{itemize}

22.3. If \( \text{authProfilesListPart} \) contains more AuthenticationProfile items than \( \text{limit} \), FAIL the test, restore the DUT state, and skip other steps.

22.4. Set \( \text{authProfileCompleteList3} := \text{authProfileCompleteList3} + \text{authProfilesListPart} \)

23. If \( \text{authProfileCompleteList3} \) contains at least two AuthenticationProfile item with equal token, FAIL the test, restore the DUT state, and skip other steps.

24. If \( \text{authProfileCompleteList3} \) does not contain all authentication profiles from \( \text{authProfileCompleteList1} \), FAIL the test, restore the DUT state, and skip other steps.

25. If \( \text{authProfileCompleteList3} \) contains authentication profiles other than authentication profiles from \( \text{authProfileCompleteList1} \), FAIL the test, restore the DUT state, and skip other steps.
26. ONVIF Client compares Authentication Profile List and Authentication Profile Info List by following the procedure mentioned in Annex A.7 with the following input and output parameters

- in `authProfileCompleteList3` - list of authentication profiles information
- in `authProfileInfoCompleteList` - list of authentication profiles

27. Remove all authentication profiles with tokens from `createdAuthProfileTokensList`.

28. If `securityLevelToken` is specified:

28.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

- in `securityLevelToken` - security level token

Test Result:

PASS –

- The DUT passed all assertions.

FAIL –

- The DUT did not send `GetAuthenticationProfileListResponse` message.

5.3.4 GET AUTHENTICATION PROFILE LIST - NO LIMIT

Test Case ID: AUTH_BEHAVIOR-3-1-4


Feature Under Test: GetAuthenticationProfileList

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify Get Authentication Profile List without using Limit.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

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

3. ONVIF Client creates number of authentication profiles by following the procedure mentioned in Annex A.4 with the following input and output parameters
   
   - out authProfileInfoCompleteList - complete list of authentication profiles information
   - out createdAuthProfileTokensList - list of created authentication profiles tokens
   - out securityLevelToken - created security level token (if any)
   - out cap - Authentication Behavior Service capabilities

4. ONVIF client invokes `GetAuthenticationProfileList` with parameters
   
   - Limit skipped
   - StartReference skipped

5. The DUT responds with `GetAuthenticationProfileListResponse` message with parameters
   
   - NextStartReference =: `nextStartReference`
   - AuthenticationProfile list =: `authProfileCompleteList`

6. If `authProfileCompleteList` contains more AuthenticationProfile items than `cap.MaxLimit`, FAIL the test, restore the DUT state, and skip other steps.

7. Until `nextStartReference` is not null, repeat the following steps:
   
   7.1. ONVIF client invokes `GetAuthenticationProfileList` with parameters
       
       - Limit skipped
       - StartReference := `nextStartReference`

   7.2. The DUT responds with `GetAuthenticationProfileListResponse` message with parameters
       
       - NextStartReference =: `nextStartReference`
       - AuthenticationProfile list =: `authProfilesListPart`

   7.3. If `authProfilesListPart` contains more AuthenticationProfile items than `cap.MaxLimit`, FAIL the test, restore the DUT state, and skip other steps.

   7.4. Set `authProfileCompleteList := authProfileCompleteList + authProfilesListPart`
8. If `authProfileCompleteList` contains at least two AuthenticationProfile item with equal token, FAIL the test, restore the DUT state, and skip other steps.

9. ONVIF Client compares Authentication Profile List and Authentication Profile Info List by following the procedure mentioned in Annex A.7 with the following input and output parameters

   - in `authProfileCompleteList` - list of authentication profiles information
   - in `authProfileInfoCompleteList` - list of authentication profiles

10. Remove all authentication profiles with tokens from `createdAuthProfileTokensList`.

11. If `securityLevelToken` is specified:

   11.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

       - in `securityLevelToken` - security level token

**Test Result:**

**PASS** –

- The DUT passed all assertions.

**FAIL** –

- The DUT did not send `GetAuthenticationProfileListResponse` message.

### 5.3.5 GET AUTHENTICATION PROFILES WITH INVALID TOKEN

**Test Case ID:** AUTH_BEHAVIOR-3-1-5

**Specification Coverage:** AuthenticationProfile (ONVIF Authentication Behavior Service Specification), GetAuthenticationProfiles command (ONVIF Authentication Behavior Service Specification)

**Feature Under Test:** GetAuthenticationProfiles

**WSDL Reference:** authenticationbehavior.wsdl

**Test Purpose:** To verify Get Authentication Profile with invalid token.

**Pre-Requisite:** Authentication Behavior Service is received from the DUT.
Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client creates number of authentication profiles by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out authProfileInfoCompleteList - complete list of authentication profiles information
   - out createdAuthProfileTokensList - list of created authentication profiles tokens
   - out securityLevelToken - created security level token (if any)
   - out cap - Authentication Behavior Service capabilities

4. Set invalidToken := value not equal to any authProfileInfoCompleteList.token.

5. ONVIF client invokes GetAuthenticationProfiles with parameters
   - Token list := invalidToken

6. The DUT responds with GetAuthenticationProfilesResponse message with parameters
   - AuthenticationProfile list =: authProfilesList

7. If authProfilesList is not empty, FAIL the test, restore the DUT state, and skip other steps.

8. If cap.MaxLimit is less than 2, go to step 14.

9. ONVIF client invokes GetAuthenticationProfileInfo with parameters
   - Token[0] := invalidToken
   - Token[1] := authProfileInfoCompleteList[0].token

10. The DUT responds with GetAuthenticationProfileInfoResponse message with parameters
    - AuthenticationProfileInfo list =: authProfilesList

11. If authProfilesList is empty, FAIL the test, restore the DUT state, and skip other steps.

12. If authProfilesList contains more than one item, FAIL the test, restore the DUT state, and skip other steps.
13. If `authProfilesList[0].token` is not equal to `authProfileInfoCompleteList[0].token`, FAIL the test, restore the DUT state, and skip other steps.

14. Remove all authentication profiles with tokens from `createdAuthProfileTokensList`.

15. If `securityLevelToken` is specified:
   
   15.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters
   
   • in `securityLevelToken` - security level token

Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send `GetAuthenticationProfilesResponse` message.

5.3.6 GET AUTHENTICATION PROFILE - TOO MANY ITEMS

Test Case ID: AUTH_BEHAVIOR-3-1-6


Feature Under Test: GetAuthenticationProfiles

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify Get Authentication Profile in case there are more items than MaxLimit in request.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client creates number of authentication profiles by following the procedure mentioned in Annex A.4 with the following input and output parameters
• out `authProfileInfoCompleteList` - complete list of authentication profiles information

• out `createdAuthProfileTokensList` - list of created authentication profiles tokens

• out `securityLevelToken` - created security level token (if any)

• out `cap` - Authentication Behavior Service capabilities

4. If `authProfileCompleteList`.token items number is less than `cap.MaxLimit` or equal to `cap.MaxLimit`, go to step 8.

5. Set `tokenList` := [subset of `authProfileInfoCompleteList`.token values with items number equal to `cap.MaxLimit + 1`].

6. ONVIF client invokes `GetAuthenticationProfiles` with parameters

   • Token list := `tokenList`


8. Remove all authentication profiles with tokens from `createdAuthProfileTokensList`.

9. If `securityLevelToken` is specified:

   9.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

      • in `securityLevelToken` - security level token

Test Result:

PASS –

  • The DUT passed all assertions.

FAIL –

  • The DUT did not send `env:Sender/ter:InvalidArgs/ter:TooManyItems` SOAP 1.2 fault

5.4 Authentication Profile Management

5.4.1 CREATE AUTHENTICATION PROFILE WITHOUT AUTHENTICATION POLICIES

Test Case ID: AUTH_BEHAVIOR-4-1-1
**Specification Coverage:** AuthenticationProfileInfo (ONVIF Authentication Behavior Service Specification), AuthenticationProfile (ONVIF Authentication Behavior Service Specification), CreateAuthenticationProfile command (ONVIF Authentication Behavior Service Specification)

**Feature Under Test:** CreateAuthenticationProfile

**WSDL Reference:** authenticationbehavior.wsdl, event.wsdl

**Test Purpose:** To verify creation of authentication profile without any authentication policies and generating of appropriate notifications.

**Pre-Requisite:** Authentication Behavior Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional Authentication Profile.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves a complete list of authentication profile info by following the procedure mentioned in Annex A.1 with the following input and output parameters
   - out authProfileInfoInitialList - complete list of authentication profiles information
4. ONVIF Client find existing or create new security level by following the procedure mentioned in Annex A.5 with the following input and output parameters
   - out securityLevelToken - security level token
   - out newSecurityLevel - flag if new security level was created
5. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - in "tns1:Configuration/AuthenticationProfile/Changed" - Notification Topic
   - out s - Subscription reference
   - out currentTime - current time for the DUT
   - out terminationTime - Subscription termination time
6. ONVIF client invokes CreateAuthenticationProfile with parameters
• AuthenticationProfile.token := ""
• AuthenticationProfile.Description := "Test Description"
• AuthenticationProfile.Name := "Test Name"
• AuthenticationProfile.DefaultSecurityLevelToken := securityLevelToken
• AuthenticationProfile.AuthenticationPolicy is skipped

7. The DUT responds with CreateAuthenticationProfileResponse message with parameters
   • Token =: authProfileToken

8. ONVIF Client retrieves and checks tns1:Configuration/AuthenticationProfile/Changed event for the specified Authentication Profile token by following the procedure mentioned in Annex A.10 with the following input and output parameters
   • in s - Subscription reference
   • in currentTime - current time for the DUT
   • in terminationTime - subscription termination time
   • in authProfileToken - Authentication Profile token

9. ONVIF Client retrieves a authentication profile by following the procedure mentioned in Annex A.11 with the following input and output parameters
   • in authProfileToken - authentication profile token
   • out authProfilesList - authentication profile list

10. If authProfilesList[0] item does not have equal field values to values from step 6, FAIL the test, restore the DUT state, and skip other steps.

11. ONVIF Client retrieves a authentication profile information by following the procedure mentioned in Annex A.12 with the following input and output parameters
    • in authProfileToken - authentication profile token
    • out authProfileInfoList - authentication profile information list

12. If authProfileInfoList[0] item does not have equal field values to values from step 6, FAIL the test, restore the DUT state, and skip other steps.

13. ONVIF Client retrieves a complete list of authentication profile info by following the procedure mentioned in Annex A.1 with the following input and output parameters
14. If `authProfileInfoCompleteList` does not have AuthenticationProfileInfo[token = `authProfileToken`] item with equal field values to values from step 6, FAIL the test, restore the DUT state, and skip other steps.

15. ONVIF Client retrieves a complete list of authentication profiles by following the procedure mentioned in Annex A.3 with the following input and output parameters

- out `authProfileCompleteList` - complete list of authentication profiles

16. If `authProfileCompleteList` does not have AuthenticationProfile[token = `authProfileToken`] item with equal field values to values from step 6, FAIL the test, restore the DUT state, and skip other steps.

17. For each AuthenticationProfileInfo.token (token) from `authProfileInfoInitialList` do the following:

17.1. If `authProfileCompleteList` does not have AuthenticationProfile[token = `token`] item, FAIL the test, restore the DUT state, and skip other steps.

18. ONVIF Client deletes a authentication profile by following the procedure mentioned in Annex A.13 with the following input and output parameters

- in `authProfileToken` - authentication profile token

19. If `newSecurityLevel` = true:

19.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

- in `securityLevelToken` - security level token

Test Result:

PASS –

- The DUT passed all assertions.

FAIL –

- The DUT did not send `CreateAuthenticationProfileResponse` message.

Note: The following fields are compared at steps 10 and 14:

- AuthenticationProfile:
ONVIF Authentication Behavior Device Test Specification Version 18.12

- token
- Name
- Description
- DefaultSecurityLevelToken
- AuthenticationPolicy list
  - ScheduleToken
  - SecurityLevelConstraint list
    - ActiveRegularSchedule
    - ActiveSpecialDaySchedule
  - AuthenticationMode list
  - SecurityLevelToken

**Note:** The following fields are compared at step 12 and 16:

- AuthenticationProfileInfo:
  - token
  - Name
  - Description

### 5.4.2 CREATE AUTHENTICATION PROFILE WITH AUTHENTICATION POLICY

**Test Case ID:** AUTH_BEHAVIOR-4-1-2

**Specification Coverage:** AuthenticationProfileInfo (ONVIF Authentication Behavior Service Specification), AuthenticationProfile (ONVIF Authentication Behavior Service Specification), CreateAuthenticationProfile command (ONVIF Authentication Behavior Service Specification)

**Feature Under Test:** CreateAuthenticationProfile

**WSDL Reference:** authenticationbehavior.wsdl, event.wsdl

**Test Purpose:** To verify creation of authentication profile and generating of appropriate notifications.
Pre-Requisite: Authentication Behavior Service is received from the DUT. Schedule Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional Authentication Profile.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of authentication profile info by following the procedure mentioned in Annex A.1 with the following input and output parameters

   • out authProfileInfoInitialList - complete list of authentication profiles information

4. ONVIF Client find existing or create new security level by following the procedure mentioned in Annex A.5 with the following input and output parameters

   • out securityLevelToken - security level token
   • out newSecurityLevel - flag if new security level was created

5. ONVIF Client find existing or create new schedule by following the procedure mentioned in Annex A.14 with the following input and output parameters

   • out scheduleToken - schedule level token
   • out newSchedule - flag if new schedule was created

6. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters

   • in "tns1:Configuration/AuthenticationProfile/Changed" - Notification Topic
   • out s - Subscription reference
   • out currentTime - current time for the DUT
   • out terminationTime - Subscription termination time

7. ONVIF client invokes CreateAuthenticationProfile with parameters

   • AuthenticationProfile.token := ""
   • AuthenticationProfile.Description := "Test Description"
   • AuthenticationProfile.Name := "Test Name"
ONVIF Authentication Behavior Device Test Specification Version 18.12

8. The DUT responds with `CreateAuthenticationProfileResponse` message with parameters

   - Token = `authProfileToken`

9. ONVIF Client retrieves and checks `tns1:Configuration/AuthenticationProfile/Changed` event for the specified Authentication Profile token by following the procedure mentioned in Annex A.10 with the following input and output parameters

   - in `subscriptionReference`
   - in `currentTime` - current time for the DUT
   - in `terminationTime` - subscription termination time
   - in `authProfileToken` - Authentication Profile token

10. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.9 with the following input and output parameters

   - in `subscriptionReference`

11. ONVIF Client retrieves a authentication profile by following the procedure mentioned in Annex A.11 with the following input and output parameters

   - in `authProfileToken` - authentication profile token
   - out `authProfilesList` - authentication profile list

12. If `authProfilesList[0]` item does not have equal field values to values from step 7, FAIL the test, restore the DUT state, and skip other steps.
13. ONVIF Client retrieves a authentication profile information by following the procedure mentioned in Annex A.12 with the following input and output parameters

- in authProfileToken - authentication profile token
- out authProfileInfoList - authentication profile information list

14. If authProfileInfoList[0] item does not have equal field values to values from step 7, FAIL the test, restore the DUT state, and skip other steps.

15. ONVIF Client retrieves a complete list of authentication profile info by following the procedure mentioned in Annex A.1 with the following input and output parameters

- out authProfileInfoCompleteList - complete list of authentication profiles information

16. If authProfileInfoCompleteList does not have AuthenticationProfileInfo[token = authProfileToken] item with equal field values to values from step 7, FAIL the test, restore the DUT state, and skip other steps.

17. ONVIF Client retrieves a complete list of authentication profiles by following the procedure mentioned in Annex A.3 with the following input and output parameters

- out authProfileCompleteList - complete list of authentication profiles

18. If authProfileCompleteList does not have AuthenticationProfile[token = authProfileToken] item with equal field values to values from step 7, FAIL the test, restore the DUT state, and skip other steps.

19. For each AuthenticationProfileInfo.token (token) from authProfileInfoInitialList do the following:

19.1. If authProfileCompleteList does not have AuthenticationProfile[token = token] item, FAIL the test, restore the DUT state, and skip other steps.

20. ONVIF Client deletes a authentication profile by following the procedure mentioned in Annex A.13 with the following input and output parameters

- in authProfileToken - authentication profile token

21. If newSchedule = true:

21.1. ONVIF Client deletes schedule by following the procedure mentioned in Annex A.23 with the following input and output parameters

- in scheduleToken - schedule token

22. If newSecurityLevel = true:
22.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

- in `securityLevelToken` - security level token

**Test Result:**

**PASS** –

- The DUT passed all assertions.

**FAIL** –

- The DUT did not send `CreateAuthenticationProfileResponse` message.

**Note:** The following fields are compared at steps **12** and **16**:

- AuthenticationProfile:
  - token
  - Name
  - Description
  - DefaultSecurityLevelToken
  - AuthenticationPolicy list
    - ScheduleToken
    - SecurityLevelConstraint list
      - ActiveRegularSchedule
      - ActiveSpecialDaySchedule
    - AuthenticationMode list
      - SecurityLevelToken

**Note:** The following fields are compared at step **14** and **18**:

- AuthenticationProfileInfo:
  - token
  - Name
5.4.3 MODIFY AUTHENTICATION PROFILE

**Test Case ID:** AUTH_BEHAVIOR-4-1-3

**Specification Coverage:** AuthenticationProfileInfo (ONVIF Authentication Behavior Service Specification), AuthenticationProfile (ONVIF Authentication Behavior Service Specification), ModifyAuthenticationProfile command (ONVIF Authentication Behavior Service Specification)

**Feature Under Test:** ModifyAuthenticationProfile

**WSDL Reference:** authenticationbehavior.wsdl, event.wsdl

**Test Purpose:** To verify modifying of authentication profile and generating of appropriate notifications.

**Pre-Requisite:** Authentication Behavior Service is received from the DUT. Schedule Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional Authentication Profile.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of authentication profile info by following the procedure mentioned in Annex A.1 with the following input and output parameters

   - out authProfileInfoInitialList - complete list of authentication profiles information

4. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.2 with the following input and output parameters

   - out cap - Authentication Behavior Service capabilities

5. ONVIF Client gets the schedule service capabilities by following the procedure mentioned in Annex A.18 with the following input and output parameters

   - out capSchedule - Schedule Service capabilities

6. ONVIF Client creates Authentication Profile by following the procedure mentioned in Annex A.16 with the following input and output parameters
• out authProfileToken - authentication profile token
• out authProfile - authentication profile
• out newSecurityLevel - flag if new security level was created
• out newSchedule - flag if new schedule was created


8. If cap.MaxSecurityLevels > 1:

8.1. ONVIF Client creates security level by following the procedure mentioned in Annex A.17 with the following input and output parameters
• out securityLevelToken2 - security level token

8.2. Set newSecurityLevel2 := true.


10. If capSchedule.MaxSchedules > 1:

10.1. ONVIF Client creates schedule by following the procedure mentioned in Annex A.20 with the following input and output parameters
• out scheduleToken2 - schedule token

10.2. Set newSchedule2 := true.

11. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters
• in "tns1:Configuration/AuthenticationProfile/Changed" - Notification Topic
• out s - Subscription reference
• out currentTime - current time for the DUT
• out terminationTime - Subscription termination time

12. ONVIF client invokes ModifyAuthenticationProfile with parameters
• AuthenticationProfile.token := authProfileToken
• AuthenticationProfile.Description := "Test Description2"
• AuthenticationProfile.Name := "Test Name2"
• AuthenticationProfile.DefaultSecurityLevelToken := if newSecurityLevel2 = true, then securityLevelToken2, else securityLevelToken

• AuthenticationProfile.AuthenticationPolicy[0].ScheduleToken := if newSchedule2 = true, then scheduleToken2, else scheduleToken

• AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].ActiveRegularSchedule := false

• AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].ActiveSpecialDaySchedule := false

• AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].AuthenticationMode[0] := other than current TODO (clarification)

• AuthenticationProfile.AuthenticationProfile[0].SecurityLevelConstraint[0].SecurityLevelToken := if newSecurityLevel2 = true, then securityLevelToken2, else securityLevelToken

13. The DUT responds with ModifyAuthenticationProfileResponse message.

14. ONVIF Client retrieves and checks tns1:Configuration/AuthenticationProfile/Changed event for the specified Authentication Profile token by following the procedure mentioned in Annex A.10 with the following input and output parameters

   • in s - Subscription reference
   
   • in currentTime - current time for the DUT
   
   • in terminationTime - subscription termination time
   
   • in authProfileToken - Authentication Profile token

15. ONVIF Client retrieves an authentication profile by following the procedure mentioned in Annex A.11 with the following input and output parameters

   • in authProfileToken - authentication profile token

   • out authProfilesList - authentication profile list

16. If authProfilesList[0] item does not have equal field values to values from step 12, FAIL the test, restore the DUT state, and skip other steps.

17. ONVIF Client retrieves an authentication profile information by following the procedure mentioned in Annex A.12 with the following input and output parameters

   • in authProfileToken - authentication profile token
18. If authProfileInfoList[0] item does not have equal field values to values from step 12, FAIL the test, restore the DUT state, and skip other steps.

19. ONVIF client invokes ModifyAuthenticationProfile with parameters

- AuthenticationProfile.token := authProfileToken
- AuthenticationProfile.Description := "Test Description3"
- AuthenticationProfile.Name := "Test Name3"
- AuthenticationProfile.DefaultSecurityLevelToken := if newSecurityLevel2 = true, then securityLevelToken2, else securityLevelToken
- AuthenticationProfile.AuthenticationPolicy is skipped

20. The DUT responds with ModifyAuthenticationProfileResponse message.

21. ONVIF Client retrieves and checks tns1:Configuration/AuthenticationProfile/Changed event for the specified Authentication Profile token by following the procedure mentioned in Annex A.10 with the following input and output parameters

- in s - Subscription reference
- in currentTime - current time for the DUT
- in terminationTime - subscription termination time
- in authProfileToken - Authentication Profile token

22. ONVIF Client retrieves a authentication profile by following the procedure mentioned in Annex A.11 with the following input and output parameters

- in authProfileToken - authentication profile token
- out authProfilesList - authentication profile list

23. If authProfilesList[0] item does not have equal field values to values from step 19, FAIL the test, restore the DUT state, and skip other steps.

24. ONVIF Client retrieves a authentication profile information by following the procedure mentioned in Annex A.12 with the following input and output parameters

- in authProfileToken - authentication profile token
25. If authProfileInfoList[0] item does not have equal field values to values from step 19, FAIL the test, restore the DUT state, and skip other steps.

26. If cap.MaxPoliciesPerAuthenticationProfile > 1:

26.1. ONVIF client invokes ModifyAuthenticationProfile with parameters

- AuthenticationProfile.token := authProfileToken
- AuthenticationProfile.Description := "Test Description4"
- AuthenticationProfile.Name := "Test Name4"
- AuthenticationProfile.DefaultSecurityLevelToken := securityLevelToken
- AuthenticationProfile.AuthenticationPolicy[0].ScheduleToken := scheduleToken
- AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].ActiveRegularSchedule := false
- AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].ActiveSpecialDaySchedule := false
- AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].AuthenticationMode[0] := other than current TODO (clarification)
- AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].SecurityLevelToken := securityLevelToken
- AuthenticationProfile.AuthenticationPolicy[1].ScheduleToken := if newSchedule2 = true, then scheduleToken2, else scheduleToken
- AuthenticationProfile.AuthenticationPolicy[1].SecurityLevelConstraint[0].ActiveSpecialDaySchedule := false
- AuthenticationProfile.AuthenticationPolicy[1].SecurityLevelConstraint[0].AuthenticationMode[0] := other than current TODO (clarification)
- AuthenticationProfile.AuthenticationPolicy[1].SecurityLevelConstraint[0].SecurityLevelToken := if newSecurityLevel2 = true, then securityLevelToken2, else securityLevelToken
26.2. The DUT responds with **ModifyAuthenticationProfileResponse** message.

26.3. ONVIF Client retrieves and checks **tns1:Configuration/AuthenticationProfile/Changed** event for the specified Authentication Profile token by following the procedure mentioned in **Annex A.10** with the following input and output parameters

- in **s** - Subscription reference
- in **currentTime** - current time for the DUT
- in **terminationTime** - subscription termination time
- in **authProfileToken** - Authentication Profile token

26.4. ONVIF Client retrieves a authentication profile by following the procedure mentioned in **Annex A.11** with the following input and output parameters

- in **authProfileToken** - authentication profile token
- out **authProfilesList** - authentication profile list

26.5. If **authProfilesList[0]** item does not have equal field values to values from step 26.1, FAIL the test, restore the DUT state, and skip other steps.

26.6. ONVIF Client retrieves a authentication profile information by following the procedure mentioned in **Annex A.12** with the following input and output parameters

- in **authProfileToken** - authentication profile token
- out **authProfileInfoList** - authentication profile information list

26.7. If **authProfileInfoList[0]** item does not have equal field values to values from step 26.1, FAIL the test, restore the DUT state, and skip other steps.

27. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in **Annex A.9** with the following input and output parameters

- in **s** - Subscription reference

28. ONVIF Client retrieves a complete list of authentication profile by following the procedure mentioned in **Annex A.3** with the following input and output parameters

- out **authProfileUpdatedList** - complete list of authentication profiles information

29. If **authProfileUpdatedList** does not have AuthenticationProfile[token = **authProfileToken**] item, FAIL the test, restore the DUT state, and skip other steps.
30. For each AuthenticationProfile.token (token) from authProfileInitialList do the following:

30.1. If authProfileUpdatedList does not have AuthenticationProfile[token = token] item, FAIL the test, restore the DUT state, and skip other steps.

31. ONVIF Client deletes a authentication profile by following the procedure mentioned in Annex A.13 with the following input and output parameters

   • in authProfileToken - authentication profile token

32. If newSchedule = true:

32.1. ONVIF Client deletes schedule by following the procedure mentioned in Annex A.23 with the following input and output parameters

   • in scheduleToken - schedule token

33. If newSecurityLevel = true:

33.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

   • in securityLevelToken - security level token

34. If newSchedule2 = true:

34.1. ONVIF Client deletes schedule by following the procedure mentioned in Annex A.23 with the following input and output parameters

   • in scheduleToken2 - schedule token

35. If newSecurityLevel2 = true:

35.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

   • in securityLevelToken2 - security level token

Test Result:

PASS –

   • The DUT passed all assertions.

FAIL –

   • The DUT did not send ModifyAuthenticationProfileResponse message.

Note: The following fields are compared at steps 16, 23, and 26.5:
• AuthenticationProfile:
  • token
  • Name
  • Description
  • DefaultSecurityLevelToken
  • AuthenticationPolicy list
    • ScheduleToken
    • SecurityLevelConstraint list
      • ActiveRegularSchedule
      • ActiveSpecialDaySchedule
    • AuthenticationMode list
    • SecurityLevelToken

**Note:** The following fields are compared at step 18, 25, and 26.7:

• AuthenticationProfileInfo:
  • token
  • Name
  • Description

### 5.4.4 DELETE AUTHENTICATION PROFILE

**Test Case ID:** AUTH_BEHAVIOR-4-1-4

**Specification Coverage:** AuthenticationProfileInfo (ONVIF Authentication Behavior Service Specification), AuthenticationProfile (ONVIF Authentication Behavior Service Specification), DeleteAuthenticationProfile command (ONVIF Authentication Behavior Service Specification)

**Feature Under Test:** DeleteAuthenticationProfile

**WSDL Reference:** authenticationbehavior.wsdl, event.wsdl

**Test Purpose:** To verify deleting of authentication profile and generating of appropriate notifications.
Pre-Requisite: Authentication Behavior Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional Authentication Profile.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves a complete list of authentication profiles by following the procedure mentioned in Annex A.3 with the following input and output parameters
   - out authProfileInitialList - complete list of authentication profiles
4. ONVIF Client creates Authentication Profile by following the procedure mentioned in Annex A.16 with the following input and output parameters
   - out authProfileToken - authentication profile token
   - out authProfile - authentication profile
   - out newSecurityLevel - flag if new security level was created
   - out newSchedule - flag if new schedule was created
5. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - in "tns1:Configuration/AuthenticationProfile/Removed" - Notification Topic
   - out s - Subscription reference
   - out currentTime - current time for the DUT
   - out terminationTime - Subscription termination time
6. ONVIF client invokes DeleteAuthenticationProfile with parameters
   - AuthenticationProfile.token := authProfileToken
7. The DUT responds with DeleteAuthenticationProfileResponse message.
8. ONVIF Client retrieves and checks tns1:Configuration/AuthenticationProfile/Removed event for the specified Authentication Profile token by following the procedure mentioned in Annex A.21 with the following input and output parameters
• in `s` - Subscription reference

• in `currentTime` - current time for the DUT

• in `terminationTime` - subscription termination time

• in `authProfileToken` - Authentication Profile token

9. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.9 with the following input and output parameters

• in `s` - Subscription reference

10. ONVIF Client retrieves a authentication profile by following the procedure mentioned in Annex A.11 with the following input and output parameters

• in `authProfileToken` - authentication profile token

• out `authProfilesList` - authentication profile list

11. If `authProfilesList` is not empty, FAIL the test, restore the DUT state, and skip other steps.

12. ONVIF Client retrieves a authentication profile information by following the procedure mentioned in Annex A.12 with the following input and output parameters

• in `authProfileToken` - authentication profile token

• out `authProfileInfoList` - authentication profile information list

13. If `authProfileInfoList` is not empty, FAIL the test, restore the DUT state, and skip other steps.

14. ONVIF Client retrieves a complete list of authentication profile info by following the procedure mentioned in Annex A.1 with the following input and output parameters

• out `authProfileInfoList` - complete list of authentication profiles information

15. If `authProfileInfoList` contains AuthenticationProfileInfo.[token = `authProfileToken`] item, FAIL the test, restore the DUT state, and skip other steps.

16. ONVIF Client retrieves a complete list of authentication profiles by following the procedure mentioned in Annex A.3 with the following input and output parameters

• out `authProfileList` - complete list of authentication profiles

17. If `authProfileList` contains AuthenticationProfile.[token = `authProfileToken`] item, FAIL the test, restore the DUT state, and skip other steps.

18. For each AuthenticationProfile.token (token) from `authProfileInitialList` do the following:
18.1. If `authProfileList` does not have `AuthenticationProfile[token = token]` item, FAIL the test, restore the DUT state, and skip other steps.

19. If `newSchedule` = true:

19.1. ONVIF Client deletes schedule by following the procedure mentioned in Annex A.23 with the following input and output parameters

   • in `scheduleToken` - schedule token

20. If `newSecurityLevel` = true:

20.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

   • in `securityLevelToken` - security level token

**Test Result:**

**PASS** –

• The DUT passed all assertions.

**FAIL** –

• The DUT did not send `DeleteAuthenticationProfileResponse` message.

### 5.4.5 SET NEW AUTHENTICATION PROFILE WITHOUT AUTHENTICATION POLICIES

**Test Case ID:** AUTH_BEHAVIOR-4-1-5

**Specification Coverage:** `AuthenticationProfileInfo` (ONVIF Authentication Behavior Service Specification), `AuthenticationProfile` (ONVIF Authentication Behavior Service Specification), `SetAuthenticationProfile` command (ONVIF Authentication Behavior Service Specification)

**Feature Under Test:** `SetAuthenticationProfile`

**WSDL Reference:** `authenticationbehavior.wsdl`, `event.wsdl`

**Test Purpose:** To verify creation of authentication profile without any authentication policies and generating of appropriate notifications using `SetAuthenticationProfile` command.

**Pre-Requisite:** Authentication Behavior Service is received from the DUT. Event Service was received from the DUT. Token supplying is supported by the DUT as indicated by
ClientSuppliedTokenSupported capability. The DUT shall have enough free storage capacity for one additional Authentication Profile.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves a complete list of authentication profile info by following the procedure mentioned in Annex A.1 with the following input and output parameters
   - out authProfileInfoInitialList - complete list of authentication profiles information
4. ONVIF Client find existing or create new security level by following the procedure mentioned in Annex A.5 with the following input and output parameters
   - out securityLevelToken - security level token
   - out newSecurityLevel - flag if new security level was created
5. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - in "tns1:Configuration/AuthenticationProfile/Changed" - Notification Topic
   - out s - Subscription reference
   - out currentTime - current time for the DUT
   - out terminationTime - Subscription termination time
6. Set authProfileToken := token that differs from tokens listed in authProfileInfoInitialList.
7. ONVIF client invokes SetAuthenticationProfile with parameters
   - AuthenticationProfile.token := authProfileToken
   - AuthenticationProfile.Description := "Test Description"
   - AuthenticationProfile.Name := "Test Name"
   - AuthenticationProfile.DefaultSecurityLevelToken := securityLevelToken
   - AuthenticationProfile.AuthenticationPolicy[0].ScheduleToken := scheduleToken
   - AuthenticationProfile.AuthenticationPolicy is skipped
8. The DUT responds with **SetAuthenticationProfileResponse** message.

9. ONVIF Client retrieves a authentication profile by following the procedure mentioned in Annex A.11 with the following input and output parameters

   - in `authProfileToken` - authentication profile token
   - out `authProfilesList` - authentication profile list

10. If `authProfilesList[0]` item does not have equal field values to values from step 7, FAIL the test, restore the DUT state, and skip other steps.

11. ONVIF Client retrieves a authentication profile information by following the procedure mentioned in Annex A.12 with the following input and output parameters

   - in `authProfileToken` - authentication profile token
   - out `authProfileInfoList` - authentication profile information list

12. If `authProfileInfoList[0]` item does not have equal field values to values from step 7, FAIL the test, restore the DUT state, and skip other steps.

13. ONVIF Client retrieves a complete list of authentication profile info by following the procedure mentioned in Annex A.1 with the following input and output parameters

   - out `authProfileInfoCompleteList` - complete list of authentication profiles information

14. If `authProfileInfoCompleteList` does not have `AuthenticationProfileInfo[token = authProfileToken]` item with equal field values to values from step 7, FAIL the test, restore the DUT state, and skip other steps.

15. ONVIF Client retrieves a complete list of authentication profiles by following the procedure mentioned in Annex A.3 with the following input and output parameters

   - out `authProfileCompleteList` - complete list of authentication profiles

16. If `authProfileCompleteList` does not have `AuthenticationProfile[token = authProfileToken]` item with equal field values to values from step 7, FAIL the test, restore the DUT state, and skip other steps.

17. For each `AuthenticationProfileInfo.token (token)` from `authProfileInfoInitialList` do the following:

   17.1. If `authProfileCompleteList` does not have `AuthenticationProfile[token = token]` item, FAIL the test, restore the DUT state, and skip other steps.

18. ONVIF Client deletes a authentication profile by following the procedure mentioned in Annex A.13 with the following input and output parameters
• in authProfileToken - authentication profile token

19. If newSecurityLevel = true:

19.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

• in securityLevelToken - security level token

Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send SetAuthenticationProfileResponse message.

Note: The following fields are compared at steps 10 and 14:

• AuthenticationProfile:
  • token
  • Name
  • Description
  • DefaultSecurityLevelToken
  • AuthenticationPolicy list
  • ScheduleToken
  • SecurityLevelConstraint list
  • ActiveRegularSchedule
  • ActiveSpecialDaySchedule
  • AuthenticationMode list
  • SecurityLevelToken

Note: The following fields are compared at step 12 and 16:

• AuthenticationProfile:
5.4.6 SET NEW AUTHENTICATION PROFILE WITH AUTHENTICATION POLICY

Test Case ID: AUTH_BEHAVIOR-4-1-6


Feature Under Test: SetAuthenticationProfile

WSDL Reference: authenticationbehavior.wsdl, event.wsdl

Test Purpose: To verify creation of authentication profile and generating of appropriate notifications using SetAuthenticationProfile command.

Pre-Requisite: Authentication Behavior Service is received from the DUT. Schedule Service is received from the DUT. Event Service was received from the DUT. Token supplying is supported by the DUT as indicated by ClientSuppliedTokenSupported capability. The DUT shall have enough free storage capacity for one additional Authentication Profile.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves a complete list of authentication profile info by following the procedure mentioned in Annex A.1 with the following input and output parameters
   - out authProfileInfoInitialList - complete list of authentication profiles information
4. ONVIF Client find existing or create new security level by following the procedure mentioned in Annex A.5 with the following input and output parameters
   - out securityLevelToken - security level token
• out newSecurityLevel - flag if new security level was created

5. ONVIF Client find existing or create new schedule by following the procedure mentioned in Annex A.14 with the following input and output parameters

• out scheduleToken - schedule level token

• out newSchedule - flag if new schedule was created

6. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters

• in "tns1:Configuration/AuthenticationProfile/Changed" - Notification Topic

• out s - Subscription reference

• out currentTime - current time for the DUT

• out terminationTime - Subscription termination time

7. Set authProfileToken := token that differs from tokens listed in authProfileInfoInitialList.

8. ONVIF client invokes SetAuthenticationProfile with parameters

• AuthenticationProfile.token := authProfileToken

• AuthenticationProfile.Description := "Test Description"

• AuthenticationProfile.Name := "Test Name"

• AuthenticationProfile.DefaultSecurityLevelToken := securityLevelToken

• AuthenticationProfile.AuthenticationPolicy[0].ScheduleToken := scheduleToken

• AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].ActiveRegularSchedule := true

• AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].ActiveSpecialDaySchedule := true

• AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].AuthenticationMode[0] TODO (clarification)

• AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].SecurityLevelToken := securityLevelToken

9. The DUT responds with SetAuthenticationProfileResponse message.
10. ONVIF Client retrieves and checks `tns1:Configuration/AuthenticationProfile/Changed` event for the specified Authentication Profile token by following the procedure mentioned in Annex A.10 with the following input and output parameters:

- `in s` - Subscription reference
- `in currentTime` - current time for the DUT
- `in terminationTime` - subscription termination time
- `in authProfileToken` - Authentication Profile token

11. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.9 with the following input and output parameters:

- `in s` - Subscription reference

12. ONVIF Client retrieves a authentication profile by following the procedure mentioned in Annex A.11 with the following input and output parameters:

- `in authProfileToken` - authentication profile token
- `out authProfilesList` - authentication profile list

13. If `authProfilesList[0]` item does not have equal field values to values from step 8, FAIL the test, restore the DUT state, and skip other steps.

14. ONVIF Client retrieves a authentication profile information by following the procedure mentioned in Annex A.12 with the following input and output parameters:

- `in authProfileToken` - authentication profile token
- `out authProfileInfoList` - authentication profile information list

15. If `authProfileInfoList[0]` item does not have equal field values to values from step 8, FAIL the test, restore the DUT state, and skip other steps.

16. ONVIF Client retrieves a complete list of authentication profile info by following the procedure mentioned in Annex A.1 with the following input and output parameters:

- `out authProfileInfoCompleteList` - complete list of authentication profiles information

17. If `authProfileInfoCompleteList` does not have `AuthenticationProfileInfo[token = authProfileToken]` item with equal field values to values from step 8, FAIL the test, restore the DUT state, and skip other steps.

18. ONVIF Client retrieves a complete list of authentication profiles by following the procedure mentioned in Annex A.3 with the following input and output parameters.
• out authProfileCompleteList - complete list of authentication profiles

19. If authProfileCompleteList does not have AuthenticationProfile[token = authProfileToken] item with equal field values to values from step 8, FAIL the test, restore the DUT state, and skip other steps.

20. For each AuthenticationProfileInfo.token (token) from authProfileInfoInitialList do the following:

20.1. If authProfileCompleteList does not have AuthenticationProfile[token = token] item, FAIL the test, restore the DUT state, and skip other steps.

21. ONVIF Client deletes a authentication profile by following the procedure mentioned in Annex A.13 with the following input and output parameters

• in authProfileToken - authentication profile token

22. If newSchedule = true:

22.1. ONVIF Client deletes schedule by following the procedure mentioned in Annex A.23 with the following input and output parameters

• in scheduleToken - schedule token

23. If newSecurityLevel = true:

23.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

• in securityLevelToken - security level token

Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send SetAuthenticationProfileResponse message.

Note: The following fields are compared at steps 13 and 17:

• AuthenticationProfile:

• token

• Name
ONVIF Authentication Behavior Device Test Specification Version 18.12

- Description
- DefaultSecurityLevelToken
- AuthenticationPolicy list
  - ScheduleToken
  - SecurityLevelConstraint list
    - ActiveRegularSchedule
    - ActiveSpecialDaySchedule
    - AuthenticationMode list
  - SecurityLevelToken

**Note:** The following fields are compared at step 15 and 19:

- AuthenticationProfileInfo:
  - token
  - Name
  - Description

### 5.4.7 SET AUTHENTICATION PROFILE

**Test Case ID:** AUTH_BEHAVIOR-4-1-7

**Specification Coverage:** AuthenticationProfileInfo (ONVIF Authentication Behavior Service Specification), AuthenticationProfile (ONVIF Authentication Behavior Service Specification), SetAuthenticationProfile command (ONVIF Authentication Behavior Service Specification)

**Feature Under Test:** SetAuthenticationProfile

**WSDL Reference:** authenticationbehavior.wsdl, event.wsdl

**Test Purpose:** To verify modifying of authentication profile and generating of appropriate notifications using SetAuthenticationProfile command.

**Pre-Requirement:** Authentication Behavior Service is received from the DUT. Schedule Service is received from the DUT. Event Service was received from the DUT. Token supplying is supported by the DUT as indicated by ClientSuppliedTokenSupported capability. The DUT shall have enough free storage capacity for one additional Authentication Profile.
**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of authentication profile info by following the procedure mentioned in Annex A.1 with the following input and output parameters
   - out `authProfileInfoInitialList` - complete list of authentication profiles information

4. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.2 with the following input and output parameters
   - out `cap` - Authentication Behavior Service capabilities

5. ONVIF Client gets the schedule service capabilities by following the procedure mentioned in Annex A.18 with the following input and output parameters
   - out `capSchedule` - Schedule Service capabilities

6. ONVIF Client creates Authentication Profile by following the procedure mentioned in Annex A.16 with the following input and output parameters
   - out `authProfileToken` - authentication profile token
   - out `authProfile` - authentication profile
   - out `newSecurityLevel` - flag if new security level was created
   - out `newSchedule` - flag if new schedule was created


8. If `cap.MaxSecurityLevels > 1`:
   8.1. ONVIF Client creates security level by following the procedure mentioned in Annex A.17 with the following input and output parameters
       - out `securityLevelToken2` - security level token
   8.2. Set `newSecurityLevel2` := true.


10. If `capSchedule.MaxSchedules > 1`:
10.1. ONVIF Client creates schedule by following the procedure mentioned in Annex A.20 with the following input and output parameters

- out $scheduleToken2$ - schedule token

10.2. Set $newSchedule2 := true$.

11. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters

- in "$tns1:Configuration/AuthenticationProfile/Changed" - Notification Topic
- out $s$ - Subscription reference
- out $currentTime$ - current time for the DUT
- out $terminationTime$ - Subscription termination time

12. ONVIF client invokes $SetAuthenticationProfile$ with parameters

- AuthenticationProfile.token := $authProfileToken$
- AuthenticationProfile.Description := "Test Description2"
- AuthenticationProfile.Name := "Test Name2"
- AuthenticationProfile.DefaultSecurityLevelToken := if $newSecurityLevel2 = true$, then $securityLevelToken2$, else $securityLevelToken$
- AuthenticationProfile.AuthenticationPolicy[0].ScheduleToken := if $newSchedule2 = true$, then $scheduleToken2$, else $scheduleToken$
- AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].ActiveRegularSchedule := false
- AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].ActiveSpecialDaySchedule := false
- AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].AuthenticationMode[0] := other than current TODO (clarification)
- AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].SecurityLevelToken := if $newSecurityLevel2 = true$, then $securityLevelToken2$, else $securityLevelToken$

13. The DUT responds with $SetAuthenticationProfileResponse$ message.
14. ONVIF Client retrieves and checks `tns1:Configuration/AuthenticationProfile/Changed` event for the specified Authentication Profile token by following the procedure mentioned in Annex A.10 with the following input and output parameters

- in `s` - Subscription reference
- in `currentTime` - current time for the DUT
- in `terminationTime` - subscription termination time
- in `authProfileToken` - Authentication Profile token

15. ONVIF Client retrieves a authentication profile by following the procedure mentioned in Annex A.11 with the following input and output parameters

- in `authProfileToken` - authentication profile token
- out `authProfilesList` - authentication profile list

16. If `authProfilesList[0]` item does not have equal field values to values from step 12, FAIL the test, restore the DUT state, and skip other steps.

17. ONVIF Client retrieves a authentication profile information by following the procedure mentioned in Annex A.12 with the following input and output parameters

- in `authProfileToken` - authentication profile token
- out `authProfileInfoList` - authentication profile information list

18. If `authProfileInfoList[0]` item does not have equal field values to values from step 12, FAIL the test, restore the DUT state, and skip other steps.

19. ONVIF client invokes `SetAuthenticationProfile` with parameters

- `AuthenticationProfile.token := authProfileToken`
- `AuthenticationProfile.Description := "Test Description3"`
- `AuthenticationProfile.Name := "Test Name3"`
- `AuthenticationProfile.DefaultSecurityLevelToken := if newSecurityLevel2 = true, then securityLevelToken2, else securityLevelToken`
- `AuthenticationProfile.AuthenticationPolicy is skipped`

20. The DUT responds with `SetAuthenticationProfileResponse` message.
21. ONVIF Client retrieves and checks `tns1:Configuration/AuthenticationProfile/Changed` event for the specified Authentication Profile token by following the procedure mentioned in Annex A.10 with the following input and output parameters

- in `s` - Subscription reference
- in `currentTime` - current time for the DUT
- in `terminationTime` - subscription termination time
- in `authProfileToken` - Authentication Profile token

22. ONVIF Client retrieves a authentication profile by following the procedure mentioned in Annex A.11 with the following input and output parameters

- in `authProfileToken` - authentication profile token
- out `authProfilesList` - authentication profile list

23. If `authProfilesList[0]` item does not have equal field values to values from step 19, FAIL the test, restore the DUT state, and skip other steps.

24. ONVIF Client retrieves a authentication profile information by following the procedure mentioned in Annex A.12 with the following input and output parameters

- in `authProfileToken` - authentication profile token
- out `authProfileInfoList` - authentication profile information list

25. If `authProfileInfoList[0]` item does not have equal field values to values from step 19, FAIL the test, restore the DUT state, and skip other steps.

26. If `cap.MaxPoliciesPerAuthenticationProfile > 1`:

26.1. ONVIF client invokes `ModifyAuthenticationProfile` with parameters

- `AuthenticationProfile.token := authProfileToken`
- `AuthenticationProfile.Description := "Test Description4"`
- `AuthenticationProfile.Name := "Test Name4"`
- `AuthenticationProfile.DefaultSecurityLevelToken := securityLevelToken`
- `AuthenticationProfile.AuthenticationPolicy[0].ScheduleToken := scheduleToken`
- `AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].ActiveRegularSchedule := false`
• AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].ActiveSpecialDaySchedule := false

• AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].AuthenticationMode[0] otherwise other than current TODO (clarification)

• AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].SecurityLevelToken := securityLevelToken

• AuthenticationProfile.AuthenticationPolicy[1].ScheduleToken := if newSchedule2 = true, then scheduleToken2, else scheduleToken

• AuthenticationProfile.AuthenticationPolicy[1].SecurityLevelConstraint[0].ActiveRegularSchedule := false

• AuthenticationProfile.AuthenticationPolicy[1].SecurityLevelConstraint[0].ActiveSpecialDaySchedule := false

• AuthenticationProfile.AuthenticationPolicy[1].SecurityLevelConstraint[0].AuthenticationMode[0] otherwise other than current TODO (clarification)

• AuthenticationProfile.AuthenticationPolicy[1].SecurityLevelConstraint[0].SecurityLevelToken := if newSecurityLevel2 = true, then securityLevelToken2, else securityLevelToken

26.2. The DUT responds with ModifyAuthenticationProfileResponse message.

26.3. ONVIF Client retrieves and checks tns1:Configuration/AuthenticationProfile/Changed event for the specified Authentication Profile token by following the procedure mentioned in Annex A.10 with the following input and output parameters

• in s - Subscription reference

• in currentTime - current time for the DUT

• in terminationTime - subscription termination time

• in authProfileToken - Authentication Profile token

26.4. ONVIF Client retrieves a authentication profile by following the procedure mentioned in Annex A.11 with the following input and output parameters

• in authProfileToken - authentication profile token

• out authProfilesList - authentication profile list
26.5. If \( authProfilesList[0] \) item does not have equal field values to values from step 26.1, FAIL the test, restore the DUT state, and skip other steps.

26.6. ONVIF Client retrieves a authentication profile information by following the procedure mentioned in Annex A.12 with the following input and output parameters

- in \( authProfileToken \) - authentication profile token
- out \( authProfileInfoList \) - authentication profile information list

26.7. If \( authProfileInfoList[0] \) item does not have equal field values to values from step 26.1, FAIL the test, restore the DUT state, and skip other steps.

27. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.9 with the following input and output parameters

- in \( s \) - Subscription reference

28. ONVIF Client retrieves a complete list of authentication profile by following the procedure mentioned in Annex A.3 with the following input and output parameters

- out \( authProfileUpdatedList \) - complete list of authentication profiles information

29. If \( authProfileUpdatedList \) does not have AuthenticationProfile\[token = authProfileToken\] item, FAIL the test, restore the DUT state, and skip other steps.

30. For each AuthenticationProfile.token (\( token \)) from \( authProfileInitialList \) do the following:

   30.1. If \( authProfileUpdatedList \) does not have AuthenticationProfile\[token = token\] item, FAIL the test, restore the DUT state, and skip other steps.

31. ONVIF Client deletes a authentication profile by following the procedure mentioned in Annex A.13 with the following input and output parameters

- in \( authProfileToken \) - authentication profile token

32. If \( newSchedule = true \):

   32.1. ONVIF Client deletes schedule by following the procedure mentioned in Annex A.23 with the following input and output parameters

   - in \( scheduleToken \) - schedule token

33. If \( newSecurityLevel = true \):

   33.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters
34. If newSchedule2 = true:

34.1. ONVIF Client deletes schedule by following the procedure mentioned in Annex A.23 with the following input and output parameters

• in scheduleToken2 - schedule token

35. If newSecurityLevel2 = true:

35.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

• in securityLevelToken2 - security level token

Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send SetAuthenticationProfileResponse message.

Note: The following fields are compared at steps 16, 23, and 26.5:

• AuthenticationProfile:
  • token
  • Name
  • Description
  • DefaultSecurityLevelToken
  • AuthenticationPolicy list
  • ScheduleToken
  • SecurityLevelConstraint list
  • ActiveRegularSchedule
  • ActiveSpecialDaySchedule
  • AuthenticationMode list
Note: The following fields are compared at step 18, 25, and 26.7:

- AuthenticationProfileInfo:
  - token
  - Name
  - Description

5.4.8 CREATE AUTHENTICATION PROFILE - NOT EMPTY TOKEN

Test Case ID: AUTH_BEHAVIOR-4-1-8

Specification Coverage: CreateAuthenticationProfile command (ONVIF Authentication Behavior Service Specification)

Feature Under Test: CreateAuthenticationProfile

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify create authentication profile with not empty token.

Pre-Requisite: Authentication Behavior Service is received from the DUT. The DUT shall have enough free storage capacity for one additional AuthenticationProfile.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client find existing or create new security level by following the procedure mentioned in Annex A.5 with the following input and output parameters
   - out securityLevelToken - security level token
   - out newSecurityLevel - flag if new security level was created
4. ONVIF client invokes CreateAuthenticationProfile with parameters
• AuthenticationProfile.token := "Token"
• AuthenticationProfile.Name := "Test Name"
• AuthenticationProfile.Description is skipped
• AuthenticationProfile.DefaultSecurityLevelToken := securityLevelToken
• AuthenticationProfile.AuthenticationPolicy is skipped
• AuthenticationProfile.Extension is skipped

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

6. If newSecurityLevel = true:
   6.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters
   • in securityLevelToken - security level token

Test Result:

PASS –
• The DUT passed all assertions.

FAIL –
• The DUT did not send env:Sender/ter:InvalidArgVal SOAP 1.2 fault.

5.4.9 CREATE AUTHENTICATION PROFILE - CAPABILITY VIOLATED (MAX POLICIES PER AUTHENTICATION PROFILE)

Test Case ID: AUTH_BEHAVIOR-4-1-9

Specification Coverage: CreateAuthenticationProfile command (ONVIF Authentication Behavior Service Specification)

Feature Under Test: CreateAuthenticationProfile

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify creation of authentication profile with maximum number of authentication policies per authentication profile.
**Pre-Requisite:** Authentication Behavior Service is received from the DUT. The DUT shall have enough free storage capacity for one additional Authentication Profile.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.2 with the following input and output parameters
   - out cap - Authentication Behavior Service capabilities

4. If cap.MaxPoliciesPerAuthenticationProfile value is more than 50, skip other steps.

5. ONVIF Client find existing or create new security level by following the procedure mentioned in Annex A.5 with the following input and output parameters
   - out securityLevelToken - security level token
   - out newSecurityLevel - flag if new security level was created

6. ONVIF Client find existing or create new schedule by following the procedure mentioned in Annex A.14 with the following input and output parameters
   - out scheduleToken - schedule level token
   - out newSchedule - flag if new schedule was created

7. If cap.MaxPoliciesPerAuthenticationProfile is equal to one, go to step 12.

8. Set authenticationPolicy :=
   - ScheduleToken := scheduleToken
   - SecurityLevelConstraint[0].ActiveRegularSchedule := true
   - SecurityLevelConstraint[0].ActiveSpecialDaySchedule := true
   - SecurityLevelConstraint[0].AuthenticationMode[0] := TODO (clarification)
   - SecurityLevelConstraint[0].SecurityLevelToken := securityLevelToken

9. ONVIF client invokes **CreateAuthenticationProfile** with parameters
   - AuthenticationProfile.token := ""

---

www.onvif.org
ONVIF Authentication Behavior Device Test Specification Version 18.12

- AuthenticationProfile.Description := "Test Description"
- AuthenticationProfile.Name := "Test Name"
- AuthenticationProfile.DefaultSecurityLevelToken := securityLevelToken
- AuthenticationProfile.AuthenticationPolicy list := authenticationPolicy duplicated cap.MaxPoliciesPerAuthenticationProfile number of times

10. The DUT responds with **CreateAuthenticationProfileResponse** message with parameters

- Token =: authProfileToken

11. ONVIF Client deletes a authentication profile by following the procedure mentioned in Annex A.13 with the following input and output parameters

- in authProfileToken - authentication profile token

12. ONVIF client invokes **CreateAuthenticationProfile** with parameters

- AuthenticationProfile.token := ""
- AuthenticationProfile.Description := "Test Description"
- AuthenticationProfile.Name := "Test Name"
- AuthenticationProfile.DefaultSecurityLevelToken := securityLevelToken
- AuthenticationProfile.AuthenticationPolicy list := authenticationPolicy duplicated cap.MaxPoliciesPerAuthenticationProfile + 1 number of times


14. If newSchedule = true:

14.1. ONVIF Client deletes schedule by following the procedure mentioned in Annex A.23 with the following input and output parameters

- in scheduleToken - schedule token

15. If newSecurityLevel = true:

15.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

- in securityLevelToken - security level token
Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send `CreateAuthenticationProfileResponse` mesage.

• The DUT did not send `env:Sender/ter:CapabilityViolated/ter:MaxPoliciesPerAuthenticationProfile` SOAP 1.2 fault.

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

5.4.10 MODIFY AUTHENTICATION PROFILE - INVALID TOKEN

Test Case ID: AUTH_BEHAVIOR-4-1-10

Specification Coverage: ModifyAuthenticationProfile command (ONVIF Authentication Behavior Service Specification)

Feature Under Test: ModifyAuthenticationProfile

WSDL Reference: `authenticationbehavior.wsdl`

Test Purpose: To verify modifying of authentication profile with invalid token.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of authentication profile info by following the procedure mentioned in Annex A.1 with the following input and output parameters

   • `out authProfileInfoList` - complete list of authentication profiles information

4. Set `invalidToken` := value not equal to any `authProfileInfoList.token`

5. ONVIF Client find existing or create new security level by following the procedure mentioned in Annex A.5 with the following input and output parameters
6. ONVIF client invokes `ModifyAuthenticationProfile` with parameters

- AuthenticationProfile.token := `invalidToken`
- AuthenticationProfile.Name := "Test Name"
- AuthenticationProfile.Description is skipped
- AuthenticationProfile.DefaultSecurityLevelToken := `securityLevelToken`
- AuthenticationProfile.AuthenticationPolicy is skipped
- AuthenticationProfile.Extension is skipped


**Test Result:**

**PASS –**

- The DUT passed all assertions.

**FAIL –**

- The DUT did not send `env:Sender/ter:InvalidArgVal/ter:NotFound` SOAP 1.2 fault

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

5.4.11 MODIFY AUTHENTICATION PROFILE - CAPABILITY VIOLATED (MAX POLICIES PER AUTHENTICATION PROFILE)

**Test Case ID:** AUTH_BEHAVIOR-4-1-11

**Specification Coverage:** ModifyAuthenticationProfile command (ONVIF Authentication Behavior Service Specification)

**Feature Under Test:** ModifyAuthenticationProfile

**WSDL Reference:** authenticationbehavior.wsdl
**Test Purpose:** To verify modification of authentication profile with maximum number of authentication policies per authentication profile.

**Pre-Requisite:** Authentication Behavior Service is received from the DUT. The DUT shall have enough free storage capacity for one additional AuthenticationProfile.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.2 with the following input and output parameters
   - out `cap` - Authentication Behavior Service capabilities
4. If `cap.MaxPoliciesPerAuthenticationProfile` value is more than 50, skip other steps.
5. ONVIF Client creates Authentication Profile by following the procedure mentioned in Annex A.16 with the following input and output parameters
   - out `authProfileToken` - authentication profile token
   - out `authProfile` - authentication profile
   - out `newSecurityLevel` - flag if new security level was created
   - out `newSchedule` - flag if new schedule was created
6. If `cap.MaxPoliciesPerAuthenticationProfile` is equal to one, go to step 10.
7. Set `authenticationPolicy` :=
   - `ScheduleToken` := `scheduleToken`
   - `SecurityLevelConstraint[0].ActiveRegularSchedule` := true
   - `SecurityLevelConstraint[0].ActiveSpecialDaySchedule` := true
   - `SecurityLevelConstraint[0].AuthenticationMode[0]` := TODO (clarification)
   - `SecurityLevelConstraint[0].SecurityLevelToken` := `securityLevelToken`
8. ONVIF client invokes **ModifyAuthenticationProfile** with parameters
   - `AuthenticationProfile.token` := `authProfileToken`
• AuthenticationProfile.Description is skipped

• AuthenticationProfile.Name := "Test Name"

• AuthenticationProfile.DefaultSecurityLevelToken := securityLevelToken

• AuthenticationProfile.AuthenticationPolicy list := authenticationPolicy duplicated cap.MaxPoliciesPerAuthenticationProfile number of times

9. The DUT responds with ModifyAuthenticationProfileResponse message.

10. ONVIF client invokes ModifyAuthenticationProfile with parameters

   • AuthenticationProfile.token := authProfileToken

   • AuthenticationProfile.Description is skipped

   • AuthenticationProfile.Name := "Test Name"

   • AuthenticationProfile.DefaultSecurityLevelToken := securityLevelToken

   • AuthenticationProfile.AuthenticationPolicy list := authenticationPolicy duplicated cap.MaxPoliciesPerAuthenticationProfile + 1 number of times


12. ONVIF Client deletes a authentication profile by following the procedure mentioned in Annex A.13 with the following input and output parameters

   • in authProfileToken - authentication profile token

13. If newSchedule = true:

   13.1. ONVIF Client deletes schedule by following the procedure mentioned in Annex A.23 with the following input and output parameters

       • in scheduleToken - schedule token

14. If newSecurityLevel = true:

   14.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

       • in securityLevelToken - security level token

Test Result:
PASS –
• The DUT passed all assertions.

FAIL –
• The DUT did not send ModifyAuthenticationProfileResponse message.
• The DUT did not send env:Sender/ter:CapabilityViolated/ter:MaxPoliciesPerAuthenticationProfile SOAP 1.2 fault.

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

5.4.12 SET AUTHENTICATION PROFILE - CAPABILITY VIOLATED (MAX POLICIES PER AUTHENTICATION PROFILE)

Test Case ID: AUTH_BEHAVIOR-4-1-12

Specification Coverage: SetAuthenticationProfile command (ONVIF Authentication Behavior Service Specification)

Feature Under Test: SetAuthenticationProfile

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify set of authentication profile with maximum number of authentication policies per authentication profile.

Pre-Requisite: Authentication Behavior Service is received from the DUT. Token supplying is supported by the DUT as indicated by ClientSuppliedTokenSupported capability. The DUT shall have enough free storage capacity for one additional AuthenticationProfile.

Test Configuration: ONVIF Client and DUT

Test Sequence:
1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.2 with the following input and output parameters
   • out cap - Authentication Behavior Service capabilities
4. If `cap.MaxPoliciesPerAuthenticationProfile` value is more than 50, skip other steps.

5. ONVIF Client creates Authentication Profile by following the procedure mentioned in Annex A.16 with the following input and output parameters
   - `out authProfileToken` - authentication profile token
   - `out authProfile` - authentication profile
   - `out newSecurityLevel` - flag if new security level was created
   - `out newSchedule` - flag if new schedule was created

6. If `cap.MaxPoliciesPerAuthenticationProfile` is equal to one, go to step 10.

7. Set `authenticationPolicy` :=
   - `ScheduleToken := scheduleToken`
   - `SecurityLevelConstraint[0].ActiveRegularSchedule := true`
   - `SecurityLevelConstraint[0].ActiveSpecialDaySchedule := true`
   - `SecurityLevelConstraint[0].AuthenticationMode[0] := TODO (clarification)`
   - `SecurityLevelConstraint[0].SecurityLevelToken := securityLevelToken`

8. ONVIF client invokes `SetAuthenticationProfile` with parameters
   - `AuthenticationProfile.token := authProfileToken`
   - `AuthenticationProfile.Description` is skipped
   - `AuthenticationProfile.Name := "Test Name"`
   - `AuthenticationProfile.DefaultSecurityLevelToken := securityLevelToken`
   - `AuthenticationProfile.AuthenticationPolicy list := authenticationPolicy duplicated cap.MaxPoliciesPerAuthenticationProfile number of times`

9. The DUT responds with `SetAuthenticationProfileResponse` message.

10. ONVIF client invokes `SetAuthenticationProfile` with parameters
    - `AuthenticationProfile.token := authProfileToken`
    - `AuthenticationProfile.Description` is skipped
    - `AuthenticationProfile.Name := "Test Name"`
• AuthenticationProfile.DefaultSecurityLevelToken := securityLevelToken

• AuthenticationProfile.AuthenticationPolicy list := authenticationPolicy duplicated cap.MaxPoliciesPerAuthenticationProfile + 1 number of times


12. ONVIF Client deletes a authentication profile by following the procedure mentioned in Annex A.13 with the following input and output parameters

• in authProfileToken - authentication profile token

13. If newSchedule = true:

13.1. ONVIF Client deletes schedule by following the procedure mentioned in Annex A.23 with the following input and output parameters

• in scheduleToken - schedule token

14. If newSecurityLevel = true:

14.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

• in securityLevelToken - security level token

Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send SetAuthenticationProfileResponse mesage.

• The DUT did not send env:Sender/ter:CapabilityViolated/ter:MaxPoliciesPerAuthenticationProfile SOAP 1.2 fault.

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

5.4.13 SET AUTHENTICATION PROFILE - EMPTY TOKEN

Test Case ID: AUTH_BEHAVIOR-4-1-13
**Specification Coverage:** SetAuthenticationProfile command (ONVIF Authentication Behavior Service Specification)

**Feature Under Test:** SetAuthenticationProfile

**WSDL Reference:** authenticationbehavior.wsdl

**Test Purpose:** To verify set of authentication profile with empty token.

**Pre-Requisite:** Authentication Behavior Service is received from the DUT. Token supplying is supported by the DUT as indicated by ClientSuppliedTokenSupported capability. The DUT shall have enough free storage capacity for one additional AuthenticationProfile.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client find existing or create new security level by following the procedure mentioned in Annex A.5 with the following input and output parameters

   - out securityLevelToken - security level token
   - out newSecurityLevel - flag if new security level was created

4. ONVIF client invokes **SetAuthenticationProfile** with parameters

   - AuthenticationProfile.token := ""
   - AuthenticationProfile.Description is skipped
   - AuthenticationProfile.Name := "Test Name"
   - AuthenticationProfile.DefaultSecurityLevelToken := securityLevelToken
   - AuthenticationProfile.AuthenticationPolicy is skipped

5. The DUT returns **env:Sender/ter:InvalidArgs** SOAP 1.2 fault.

6. If newSecurityLevel = true:

   6.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

       - in securityLevelToken - security level token
Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send env:Sender/ter:InvalidArgs SOAP 1.2 fault.

5.4.14 DELETE AUTHENTICATION PROFILE - INVALID TOKEN

Test Case ID: AUTH_BEHAVIOR-4-1-14

Specification Coverage: DeleteAuthenticationProfile command (ONVIF Authentication Behavior Service Specification)

Feature Under Test: DeleteAuthenticationProfile

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify deleting of authentication profile with invalid token.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of authentication profile info by following the procedure mentioned in Annex A.1 with the following input and output parameters

   • out authProfileInfoList - complete list of authentication profiles information

4. Set invalidToken := value not equal to any authProfileInfoList.token

5. ONVIF Client invokes DeleteAuthenticationProfile with parameters

   • Token := invalidToken

Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send `env:Sender/ter:InvalidArgVal/ter:NotFound` SOAP 1.2 fault.

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

5.4.15 DELETE AUTHENTICATION PROFILE - NO TOKEN

Test Case ID: AUTH_BEHAVIOR-4-1-15

Specification Coverage: DeleteAuthenticationProfile command (ONVIF Authentication Behavior Service Specification)

Feature Under Test: DeleteAuthenticationProfile

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify deleting of authentication profile without token.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client invokes `DeleteAuthenticationProfile` with parameters

   • Token := ""


Test Result:

PASS –

• The DUT passed all assertions.
FAIL –

• The DUT did not send `env:Sender/ter:InvalidArgVal` SOAP 1.2 fault.

5.5 Security Level Info

5.5.1 GET SECURITY LEVEL INFO

Test Case ID: AUTH_BEHAVIOR-5-1-1


Feature Under Test: GetSecurityLevelInfo

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify Get Security Level Info.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client creates number of security levels by following the procedure mentioned in Annex A.24 with the following input and output parameters

   • `out securityLevelInfoCompleteList` - complete list of security levels information
   • `out createdSecurityLevelTokensList` - list of created security levels tokens
   • `out cap` - Authentication Behavior Service capabilities

4. Set `tokenList := [subset of securityLevelInfoCompleteList.token values with items number equal to cap.MaxLimit]`

5. ONVIF client invokes `GetSecurityLevelInfo` with parameters

   • Token list := `tokenList`
6. The DUT responds with `GetSecurityLevelInfoResponse` message with parameters
   - `SecurityLevelInfo list := securityLevelInfoList1`

7. If `securityLevelInfoList1` does not contain `SecurityLevelInfo` item for each token from `tokenList`, FAIL the test, restore the DUT state, and skip other steps.

8. If `securityLevelInfoList1` contains at least two `SecurityLevelInfo` items with equal token, FAIL the test, restore the DUT state, and skip other steps.

9. If `securityLevelInfoList1` contains other `SecurityLevelInfo` items than listed in `tokenList`, FAIL the test, restore the DUT state, and skip other steps.

10. For each `SecurityLevelInfo.token` `token` from `securityLevelInfoCompleteList` repeat the following steps:
   10.1. ONVIF client invokes `GetSecurityLevelInfo` with parameters
       - `Token[0] := token`
   10.2. The DUT responds with `GetSecurityLevelInfoResponse` message with parameters
       - `SecurityLevelInfo list := securityLevelInfoList2`
   10.3. If `securityLevelInfoList2` does not contain only one `SecurityLevelInfo` item with token equal to `token`, FAIL the test, restore the DUT state, and skip other steps.
   10.4. If `securityLevelInfoList2[0]` item is not equal to `securityLevelInfoCompleteList[token = token]` item, FAIL the test, restore the DUT state, and skip other steps.

11. Remove all security levels with tokens from `createdSecurityLevelTokensList`.

**Test Result:**

**PASS** –
   - The DUT passed all assertions.

**FAIL** –
   - The DUT did not send `GetSecurityLevelInfoResponse` message.

**Note:** If number of items in `securityLevelInfoCompleteList` is less than `cap.MaxLimit`, then all `securityLevelInfoCompleteList.Token` items shall be used for the step 4.

**Note:** The following fields are compared at step 10.4:
   - `SecurityLevelInfo:`
5.5.2 GET SECURITY LEVEL INFO LIST - LIMIT

Test Case ID: AUTH_BEHAVIOR-5-1-2


Feature Under Test: GetSecurityLevelInfoList

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify Get Security Level Info List using Limit.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client creates number of security levels by following the procedure mentioned in Annex A.24 with the following input and output parameters
   - out securityLevelInfoCompleteList - complete list of security levels information
   - out createdSecurityLevelTokensList - list of created security levels tokens
   - out cap - Authentication Behavior Service capabilities

4. ONVIF client invokes GetSecurityLevelInfoList with parameters
   - Limit := 1
   - StartReference skipped
5. The DUT responds with **GetSecurityLevelInfoListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - SecurityLevelInfo list =: *securityLevelInfoList1*

6. If *securityLevelInfoList1* contains more SecurityLevelInfo items than 1, FAIL the test, restore the DUT state, and skip other steps.

7. If *cap.MaxLimit* is equal to 1, go to step 16.

8. ONVIF client invokes **GetSecurityLevelInfoList** with parameters
   - Limit := *cap.MaxLimit*
   - StartReference skipped

9. The DUT responds with **GetSecurityLevelInfoListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - SecurityLevelInfo list =: *securityLevelInfoList2*

10. If *securityLevelInfoList2* contains more SecurityLevelInfo items than *cap.MaxLimit*, FAIL the test, restore the DUT state, and skip other steps.

11. If *cap.MaxLimit* is equal to 2, go to step 16.

12. Set *limit* := [number between 1 and *cap.MaxLimit*].

13. ONVIF client invokes **GetSecurityLevelInfoList** with parameters
   - Limit := *limit*
   - StartReference skipped

14. The DUT responds with **GetSecurityLevelInfoListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - SecurityLevelInfo list =: *securityLevelInfoList3*

15. If *securityLevelInfoList3* contains more SecurityLevelInfo items than *limit*, FAIL the test, restore the DUT state, and skip other steps.

16. Remove all security levels with tokens from *createdSecurityLevelTokensList*.

**Test Result:**

---

www.onvif.org
PASS –

- The DUT passed all assertions.

FAIL –

- The DUT did not send `GetSecurityLevelInfoListResponse` message.

5.5.3 GET SECURITY LEVEL INFO LIST - START REFERENCE AND LIMIT

Test Case ID: AUTH_BEHAVIOR-5-1-3


Feature Under Test: GetSecurityLevelInfoList

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify Get Security Level Info List using StartReference and Limit.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client creates number of security levels by following the procedure mentioned in Annex A.24 with the following input and output parameters

   - `out securityLevelCompleteList` - complete list of security levels information
   - `out createdSecurityLevelTokensList` - list of created security levels tokens
   - `out cap` - Authentication Behavior Service capabilities

4. ONVIF client invokes `GetSecurityLevelInfoList` with parameters

   - `Limit := cap.MaxLimit`
5. The DUT responds with \texttt{GetSecurityLevelInfoListResponse} message with parameters
   \begin{itemize}
   \item NextStartReference \(=\ nextStartReference\)
   \item SecurityLevelInfo list \(=\ securityLevelInfoCompleteList1\)
   \end{itemize}

6. If \texttt{securityLevelInfoCompleteList1} contains more SecurityLevelInfo items than \texttt{cap.MaxLimit}, FAIL the test, restore the DUT state, and skip other steps.

7. Until \texttt{nextStartReference} is not null, repeat the following steps:
   \begin{enumerate}
   \item ONVIF client invokes \texttt{GetSecurityLevelInfoList} with parameters
     \begin{itemize}
     \item Limit \(=\ cap.MaxLimit\)
     \item StartReference \(=\ nextStartReference\)
     \end{itemize}
   \item The DUT responds with \texttt{GetSecurityLevelInfoListResponse} message with parameters
     \begin{itemize}
     \item NextStartReference \(=\ nextStartReference\)
     \item SecurityLevelInfo list \(=\ securityLevelInfoListPart\)
     \end{itemize}
   \item If \texttt{securityLevelInfoListPart} contains more SecurityLevelInfo items than \texttt{cap.MaxLimit}, FAIL the test, restore the DUT state, and skip other steps.
   \item Set \texttt{securityLevelInfoCompleteList1} \(=\ securityLevelInfoCompleteList1 + securityLevelInfoListPart\)
   \end{enumerate}

8. If \texttt{securityLevelInfoCompleteList1} contains at least two SecurityLevelInfo item with equal token, FAIL the test, restore the DUT state, and skip other steps.

9. If \texttt{cap.MaxLimit} is equal to 1, go to step 26.

10. ONVIF client invokes \texttt{GetSecurityLevelInfoList} with parameters
    \begin{itemize}
    \item Limit \(=\ 1\)
    \item StartReference skipped
    \end{itemize}

11. The DUT responds with \texttt{GetSecurityLevelInfoListResponse} message with parameters
    \begin{itemize}
    \item NextStartReference \(=\ nextStartReference\)
    \item SecurityLevelInfo list \(=\ securityLevelInfoCompleteList2\)
    \end{itemize}
12. If `securityLevelInfoCompleteList2` contains more SecurityLevelInfo items than 1, FAIL the test, restore the DUT state, and skip other steps.

13. Until `nextStartReference` is not null, repeat the following steps:

   13.1. ONVIF client invokes `GetSecurityLevelInfoList` with parameters
       
       • Limit := 1

       • StartReference := `nextStartReference`

   13.2. The DUT responds with `GetSecurityLevelInfoListResponse` message with parameters
       
       • NextStartReference =: `nextStartReference`

       • SecurityLevelInfo list =: `securityLevelInfoListPart`

   13.3. If `securityLevelInfoListPart` contains more SecurityLevelInfo items than 1, FAIL the test, restore the DUT state, and skip other steps.

   13.4. Set `securityLevelInfoCompleteList2` := `securityLevelInfoCompleteList2` +

       `securityLevelInfoListPart`

14. If `securityLevelInfoCompleteList2` contains at least two SecurityLevelInfo item with equal token, FAIL the test, restore the DUT state, and skip other steps.

15. If `securityLevelInfoCompleteList2` does not contain all security levels from `securityLevelInfoCompleteList1`, FAIL the test, restore the DUT state, and skip other steps.

16. If `securityLevelInfoCompleteList2` contains security levels other than security levels from `securityLevelInfoCompleteList1`, FAIL the test, restore the DUT state, and skip other steps.

17. If `cap.MaxLimit` is equal to 2, go to step 26.

18. Set `limit` := [number between 1 and `cap.MaxLimit`]

19. ONVIF client invokes `GetSecurityLevelInfoList` with parameters

    • Limit := `limit`

    • StartReference skipped

20. The DUT responds with `GetSecurityLevelInfoListResponse` message with parameters

    • NextStartReference =: `nextStartReference`

    • SecurityLevelInfo list =: `securityLevelInfoCompleteList3`
21. If `securityLevelInfoCompleteList3` contains more SecurityLevelInfo items than `limit`, FAIL the test, restore the DUT state, and skip other steps.

22. Until `nextStartReference` is not null, repeat the following steps:

   22.1. ONVIF client invokes `GetSecurityLevelInfoList` with parameters
   
   • Limit := `limit`
   • StartReference := `nextStartReference`

   22.2. The DUT responds with `GetSecurityLevelInfoListResponse` message with parameters
   
   • NextStartReference =: `nextStartReference`
   • SecurityLevelInfo list =: `securityLevelInfoListPart`

   22.3. If `securityLevelInfoListPart` contains more SecurityLevelInfo items than `limit`, FAIL the test, restore the DUT state, and skip other steps.

   22.4. Set `securityLevelInfoCompleteList3` := `securityLevelInfoCompleteList3` + `securityLevelInfoListPart`

23. If `securityLevelInfoCompleteList3` contains at least two SecurityLevelInfo item with equal token, FAIL the test, restore the DUT state, and skip other steps.

24. If `securityLevelInfoCompleteList3` does not contain all security levels from `securityLevelInfoCompleteList1`, FAIL the test, restore the DUT state, and skip other steps.

25. If `securityLevelInfoCompleteList3` contains security levels other than security levels from `securityLevelInfoCompleteList1`, FAIL the test, restore the DUT state, and skip other steps.

26. Remove all security levels with tokens from `createdSecurityLevelTokensList`.

**Test Result:**

**PASS** –

• The DUT passed all assertions.

**FAIL** –

• The DUT did not send `GetSecurityLevelInfoListResponse` message.

### 5.5.4 GET SECURITY LEVEL INFO LIST - NO LIMIT

**Test Case ID:** AUTH_BEHAVIOR-5-1-4

Feature Under Test: GetSecurityLevelInfoList

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify Get Security Level Info List without using Limit.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client creates number of security levels by following the procedure mentioned in Annex A.24 with the following input and output parameters

   • out securityLevelInfoCompleteList - complete list of security levels information
   • out createdSecurityLevelTokensList - list of created security levels tokens
   • out cap - Authentication Behavior Service capabilities

4. ONVIF client invokes GetSecurityLevelInfoList with parameters

   • Limit skipped
   • StartReference skipped

5. The DUT responds with GetSecurityLevelInfoListResponse message with parameters

   • NextStartReference =: nextStartReference
   • SecurityLevelInfo list =: securityLevelInfoCompleteList

6. If securityLevelInfoCompleteList contains more SecurityLevelInfo items than cap.MaxLimit, FAIL the test, restore the DUT state, and skip other steps.

7. Until nextStartReference is not null, repeat the following steps:

    7.1. ONVIF client invokes GetSecurityLevelInfoList with parameters

        • Limit skipped
7.2. The DUT responds with \texttt{GetSecurityLevelInfoListResponse} message with parameters

\begin{itemize}
    \item \texttt{NextStartReference := nextStartReference}
    \item \texttt{SecurityLevelInfo list := securityLevelInfoListPart}
\end{itemize}

7.3. If \texttt{securityLevelInfoListPart} contains more SecurityLevelInfo items than \texttt{cap.MaxLimit}, FAIL the test, restore the DUT state, and skip other steps.

7.4. Set \texttt{securityLevelInfoCompleteList := securityLevelInfoCompleteList + securityLevelInfoListPart}

8. If \texttt{securityLevelInfoCompleteList} contains at least two SecurityLevelInfo item with equal token, FAIL the test, restore the DUT state, and skip other steps.

9. If \texttt{securityLevelInfoCompleteList} contains more SecurityLevelInfo items than \texttt{cap.MaxSecurityLevels}, FAIL the test, restore the DUT state, and skip other steps.

10. Remove all security levels with tokens from \texttt{createdSecurityLevelTokensList}.

\textbf{Test Result:}

\textbf{PASS –}

\begin{itemize}
    \item The DUT passed all assertions.
\end{itemize}

\textbf{FAIL –}

\begin{itemize}
    \item The DUT did not send \texttt{GetSecurityLevelInfoListResponse} message.
\end{itemize}

\section*{5.5.5 \texttt{GET SECURITY LEVEL INFO WITH INVALID TOKEN}}

\textbf{Test Case ID:} \texttt{AUTH\_BEHAVIOR-5-1-5}


\textbf{Feature Under Test:} GetSecurityLevelInfo

\textbf{WSDL Reference:} authenticationbehavior.wsdl

\textbf{Test Purpose:} To verify Get Security Level Info with invalid token.
Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates number of security levels by following the procedure mentioned in Annex A.24 with the following input and output parameters
   - out securityLevelInfoCompleteList - complete list of security levels information
   - out createdSecurityLevelTokensList - list of created security levels tokens
   - out cap - Authentication Behavior Service capabilities
4. Set invalidToken := value not equal to any securityLevelInfoCompleteList.token
5. ONVIF client invokes GetSecurityLevelInfo with parameters
   - Token list := invalidToken
6. The DUT responds with GetSecurityLevelInfoResponse message with parameters
   - SecurityLevelInfo list =: securityLevelInfoList
7. If securityLevelInfoList is not empty, FAIL the test, restore the DUT state, and skip other steps.
8. If cap.MaxLimit is less than 2, go to step 14.
9. ONVIF client invokes GetSecurityLevelInfo with parameters
   - Token[0]:= invalidToken
   - Token[1]:= securityLevelInfoCompleteList[0].token
10. The DUT responds with GetSecurityLevelInfoResponse message with parameters
    - SecurityLevelInfo list =: securityLevelInfoList
11. If securityLevelInfoList is empty, FAIL the test, restore the DUT state, and skip other steps.
12. If securityLevelInfoList contains more than one item, FAIL the test, restore the DUT state, and skip other steps.
13. If `securityLevelInfoList[0].token` is not equal to `securityLevelInfoCompleteList[0].token`, FAIL the test, restore the DUT state, and skip other steps.

14. Remove all security levels with tokens from `createdSecurityLevelTokensList`.

Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send `GetSecurityLevelInfoResponse` message.

5.5.6 GET SECURITY LEVEL INFO - TOO MANY ITEMS

Test Case ID: AUTH_BEHAVIOR-5-1-6


Feature Under Test: GetSecurityLevelInfo

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify Get Security Level Info in case there are more items than MaxLimit in request.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client creates number of security levels by following the procedure mentioned in Annex A.24 with the following input and output parameters

   • `out securityLevelInfoCompleteList` - complete list of security levels information
   
   • `out createdSecurityLevelTokensList` - list of created security levels tokens
4. If $\text{securityLevelInfoCompleteList}.\text{token}$ items number is less than $\text{cap}.\text{MaxLimit}$ or equal to $\text{cap}.\text{MaxLimit}$, go to step 8.

5. Set $\text{tokenList} := [\text{subset of } \text{securityLevelInfoCompleteList}.\text{token} \text{ values with items number equal to } \text{cap}.\text{MaxLimit} + 1]$

6. ONVIF client invokes $\text{GetSecurityLevelInfo}$ with parameters

   • Token list := $\text{tokenList}$


8. Remove all security levels with toknens from $\text{createdSecurityLevelTokensList}$.

Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send $\text{env:Sender/ter:InvalidArgs/ter:TooManyItems}$ SOAP 1.2 fault.

5.6 Security Level

5.6.1 GET SECURITY LEVELS

Test Case ID: AUTH_BEHAVIOR-6-1-1


Feature Under Test: GetSecurityLevels

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify Get Security Level.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT
Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client creates number of security levels by following the procedure mentioned in Annex A.24 with the following input and output parameters

   - out securityLevelInfoCompleteList - complete list of security levels information
   - out createdSecurityLevelTokensList - list of created security levels tokens
   - out cap - Authentication Behavior Service capabilities

4. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.25 with the following input and output parameters

   - out securityLevelCompleteList - complete list of security levels information

5. Set tokenList := [subset of securityLevelCompleteList.token values with items number equal to cap.MaxLimit].

6. ONVIF client invokes **GetSecurityLevels** with parameters

   - Token list := tokenList

7. The DUT responds with **GetSecurityLevelsResponse** message with parameters

   - SecurityLevel list := securityLevelsList1

8. If securityLevelsList1 does not contain Security Level item for each token from tokenList, FAIL the test, restore the DUT state, and skip other steps.

9. If securityLevelsList1 contains at least two Security Level items with equal token, FAIL the test, restore the DUT state, and skip other steps.

10. If securityLevelsList1 contains other Security Level items than listed in tokenList, FAIL the test, restore the DUT state, and skip other steps.

11. For each SecurityLevel.token token from securityLevelCompleteList repeat the following steps:

   11.1. ONVIF client invokes **GetSecurityLevels** with parameters

       - Token[0] := token

   11.2. The DUT responds with **GetSecurityLevelsResponse** message with parameters
• SecurityLevel list =: securityLevelsList2

11.3. If securityLevelsList2 does not contain only one SecurityLevel item with token equal to token, FAIL the test, restore the DUT state, and skip other steps.

11.4. If securityLevelsList2[0] item does not have equal field values to securityLevelCompleteList[token = token] item, FAIL the test, restore the DUT state, and skip other steps.

12. Remove all security levels with tokens from createdSecurityLevelTokensList.

Test Result:

PASS –
• The DUT passed all assertions.

FAIL –
• The DUT did not send GetSecurityLevelsResponse message.

Note: If number of items in securityLevelCompleteList is less than cap.MaxLimit, then all securityLevelCompleteList.Token items shall be used for the step 5.

Note: The following fields are compared at step 11.4:

• SecurityLevel:
  • token
  • Name
  • Priority
  • Description
  • RecognitionGroup list
    • RecognitionMethod list
      • RecognitionType
      • Order

5.6.2 GET SECURITY LEVEL LIST - LIMIT

Test Case ID: AUTH_BEHAVIOR-6-1-2
Specification Coverage: SecurityLevel (ONVIF Authentication Behavior Service Specification),
GetSecurityLevelList command (ONVIF Authentication Behavior Service Specification)

Feature Under Test: GetSecurityLevelList

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify Get Security Level List using Limit.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client creates number of security levels by following the procedure mentioned in
   Annex A.24 with the following input and output parameters
   - out securityLevelInfoCompleteList - complete list of security levels information
   - out createdSecurityLevelTokensList - list of created security levels tokens
   - out cap - Authentication Behavior Service capabilities

4. ONVIF client invokes GetSecurityLevelList with parameters
   - Limit := 1
   - StartReference skipped

5. The DUT responds with GetSecurityLevelListResponse message with parameters
   - NextStartReference =: nextStartReference
   - SecurityLevel list =: securityLevelsList1

6. If securityLevelsList1 contains more SecurityLevel items than 1, FAIL the test, restore the
   DUT state, and skip other steps.

7. If cap.MaxLimit is equal to 1, go to step 16.

8. ONVIF client invokes GetSecurityLevelList with parameters
   - Limit := cap.MaxLimit
9. The DUT responds with `GetSecurityLevelListResponse` message with parameters
   - `NextStartReference` = `nextStartReference`
   - `SecurityLevel list` = `securityLevelsList2`

10. If `securityLevelsList2` contains more SecurityLevel items than `cap.MaxLimit`, FAIL the test, restore the DUT state, and skip other steps.

11. If `cap.MaxLimit` is equal to 2, go to step 16.

12. Set `limit` := [number between 1 and `cap.MaxLimit`]

13. ONVIF client invokes `GetSecurityLevelList` with parameters
   - `Limit` := `limit`
   - `StartReference` skipped

14. The DUT responds with `GetSecurityLevelListResponse` message with parameters
   - `NextStartReference` = `nextStartReference`
   - `SecurityLevel list` = `securityLevelsList3`

15. If `securityLevelsList3` contains more SecurityLevel items than `limit`, FAIL the test, restore the DUT state, and skip other steps.

16. Remove all security levels with tokens from `createdSecurityLevelTokensList`.

Test Result:

PASS –
- The DUT passed all assertions.

FAIL –
- The DUT did not send `GetSecurityLevelListResponse` message.

5.6.3 GET SECURITY LEVEL LIST - START REFERENCE
AND LIMIT

Test Case ID: AUTH_BEHAVIOR-6-1-3

**Feature Under Test:** GetSecurityLevelList

**WSDL Reference:** authenticationbehavior.wsdl

**Test Purpose:** To verify Get Security Level List using StartReference and Limit.

**Pre-Requisite:** Authentication Behavior Service is received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates number of security levels by following the procedure mentioned in Annex A.24 with the following input and output parameters
   - out `securityLevelInfoCompleteList` - complete list of security levels information
   - out `createdSecurityLevelTokensList` - list of created security levels tokens
   - out `cap` - Authentication Behavior Service capabilities
4. ONVIF client invokes `GetSecurityLevelList` with parameters
   - Limit := `cap`.MaxLimit
   - StartReference skipped
5. The DUT responds with `GetSecurityLevelListResponse` message with parameters
   - NextStartReference =: `nextStartReference`
   - SecurityLevel list =: `securityLevelCompleteList1`
6. If `securityLevelCompleteList1` contains more SecurityLevel items than `cap.MaxLimit`, FAIL the test, restore the DUT state, and skip other steps.
7. Until `nextStartReference` is not null, repeat the following steps:
   7.1. ONVIF client invokes `GetSecurityLevelList` with parameters
- Limit := cap.MaxLimit
- StartReference := nextStartReference

7.2. The DUT responds with **GetSecurityLevelListResponse** message with parameters

- NextStartReference :=: nextStartReference
- SecurityLevel list :=: securityLevelsListPart

7.3. If securityLevelsListPart contains more SecurityLevel items than cap.MaxLimit, FAIL the test, restore the DUT state, and skip other steps.

7.4. Set securityLevelCompleteList1 := securityLevelCompleteList1 + securityLevelsListPart.

8. If securityLevelCompleteList1 contains at least two SecurityLevel item with equal token, FAIL the test, restore the DUT state, and skip other steps.

9. If cap.MaxLimit is equal to 1, do the following steps:

9.1. ONVIF Client compares Security Level List and Security Level Info List by following the procedure mentioned in Annex A.26 with the following input and output parameters

- in securityLevelCompleteList1 - list of security levels information
- in securityLevelInfoCompleteList - list of security levels

9.2. Skip other steps.

10. ONVIF client invokes **GetSecurityLevelList** with parameters

- Limit := 1
- StartReference skipped

11. The DUT responds with **GetSecurityLevelListResponse** message with parameters

- NextStartReference :=: nextStartReference
- SecurityLevel list :=: securityLevelCompleteList2

12. If securityLevelCompleteList2 contains more SecurityLevel items than 1, FAIL the test, restore the DUT state, and skip other steps.

13. Until nextStartReference is not null, repeat the following steps:

13.1. ONVIF client invokes **GetSecurityLevelList** with parameters
• Limit := 1

• StartReference := nextStartReference

13.2. The DUT responds with GetSecurityLevelListResponse message with parameters

• NextStartReference =: nextStartReference

• SecurityLevel list =: securityLevelsListPart

13.3. If securityLevelsListPart contains more SecurityLevel items than 1, FAIL the test, restore the DUT state, and skip other steps.

13.4. Set securityLevelCompleteList2 := securityLevelCompleteList2 + securityLevelsListPart

14. If securityLevelCompleteList2 contains at least two SecurityLevel item with equal token, FAIL the test, restore the DUT state, and skip other steps.

15. If securityLevelCompleteList2 does not contain all security levels from securityLevelCompleteList1, FAIL the test, restore the DUT state, and skip other steps.

16. If securityLevelCompleteList2 contains security levels other than security levels from securityLevelCompleteList1, FAIL the test, restore the DUT state, and skip other steps.

17. If cap.MaxLimit is equal to 2 do the following steps:

17.1. ONVIF Client compares Security Level List and Security Level Info List by following the procedure mentioned in Annex A.26 with the following input and output parameters

• in securityLevelCompleteList2 - list of security levels information

• in securityLevelInfoCompleteList - list of security levels

17.2. Skip other steps.

18. Set limit := [number between 1 and cap.MaxLimit].

19. ONVIF client invokes GetSecurityLevelList with parameters

• Limit := limit

• StartReference skipped

20. The DUT responds with GetSecurityLevelListResponse message with parameters

• NextStartReference =: nextStartReference
• SecurityLevel list =: \textit{securityLevelCompleteList3}

21. If \( \textit{securityLevelCompleteList3} \) contains more SecurityLevel items than \textit{limit}, FAIL the test, restore the DUT state, and skip other steps.

22. Until \textit{nextStartReference} is not null, repeat the following steps:

\begin{enumerate}
\item ONVIF client invokes \textbf{GetSecurityLevelList} with parameters
\begin{itemize}
\item Limit := \textit{limit}
\item StartReference := \textit{nextStartReference}
\end{itemize}
\item The DUT responds with \textbf{GetSecurityLevelListResponse} message with parameters
\begin{itemize}
\item NextStartReference =: \textit{nextStartReference}
\item SecurityLevel list =: \textit{securityLevelsListPart}
\end{itemize}
\item If \textit{securityLevelsListPart} contains more SecurityLevel items than \textit{limit}, FAIL the test, restore the DUT state, and skip other steps.
\item Set \( \textit{securityLevelCompleteList3} := \textit{securityLevelCompleteList3} + \textit{securityLevelsListPart} \)
\end{enumerate}

23. If \textit{securityLevelCompleteList3} contains at least two SecurityLevel item with equal token, FAIL the test, restore the DUT state, and skip other steps.

24. If \textit{securityLevelCompleteList3} does not contain all security levels from \textit{securityLevelCompleteList1}, FAIL the test, restore the DUT state, and skip other steps.

25. If \textit{securityLevelCompleteList3} contains security levels other than security levels from \textit{securityLevelCompleteList1}, FAIL the test, restore the DUT state, and skip other steps.

26. ONVIF Client compares Security Level List and Security Level Info List by following the procedure mentioned in \textbf{Annex A.26} with the following input and output parameters

\begin{itemize}
\item in \textit{securityLevelCompleteList3} - list of security levels information
\item in \textit{securityLevelInfoCompleteList} - list of security levels
\end{itemize}

27. Remove all security levels with tokens from \textit{createdSecurityLevelTokensList}.

\textbf{Test Result:}

\textbf{PASS –}
• The DUT passed all assertions.

FAIL –

• The DUT did not send `GetSecurityLevelListResponse` message.

5.6.4 GET SECURITY LEVEL LIST - NO LIMIT

Test Case ID: AUTH_BEHAVIOR-6-1-4


Feature Under Test: GetSecurityLevelList

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify Get Security Level List without using Limit.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client creates number of security levels by following the procedure mentioned in Annex A.24 with the following input and output parameters
   • out `securityLevelInfoCompleteList` - complete list of security levels information
   • out `createdSecurityLevelTokensList` - list of created security levels tokens
   • out `cap` - Authentication Behavior Service capabilities

4. ONVIF client invokes `GetSecurityLevelList` with parameters
   • Limit skipped
   • StartReference skipped

5. The DUT responds with `GetSecurityLevelListResponse` message with parameters
• NextStartReference =: nextStartReference

• SecurityLevel list =: securityLevelCompleteList

6. If securityLevelCompleteList contains more SecurityLevel items than cap.MaxLimit, FAIL the test, restore the DUT state, and skip other steps.

7. Until nextStartReference is not null, repeat the following steps:

7.1. ONVIF client invokes GetSecurityLevelList with parameters

   • Limit skipped

   • StartReference := nextStartReference

7.2. The DUT responds with GetSecurityLevelListResponse message with parameters

   • NextStartReference =: nextStartReference

   • SecurityLevel list =: securityLevelsListPart

7.3. If securityLevelsListPart contains more SecurityLevel items than cap.MaxLimit, FAIL the test, restore the DUT state, and skip other steps.

7.4. Set securityLevelCompleteList := securityLevelCompleteList + securityLevelsListPart

8. If securityLevelCompleteList contains at least two SecurityLevel item with equal token, FAIL the test, restore the DUT state, and skip other steps.

9. ONVIF Client compares Security Level List and Security Level Info List by following the procedure mentioned in Annex A.26 with the following input and output parameters

   • in securityLevelCompleteList - list of security levels information

   • in securityLevelInfoCompleteList - list of security levels

10. Remove all security levels with tockens from createdSecurityLevelTokensList.

Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send GetSecurityLevelListResponse message.
5.6.5 GET SECURITY LEVELS WITH INVALID TOKEN

Test Case ID: AUTH_BEHAVIOR-6-1-5


Feature Under Test: GetSecurityLevels

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify Get Security Level with invalid token.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client creates number of security levels by following the procedure mentioned in Appendix A.24 with the following input and output parameters
   - out securityLevelInfoCompleteList - complete list of security levels information
   - out createdSecurityLevelTokensList - list of created security levels tokens
   - out cap - Authentication Behavior Service capabilities
4. Set invalidToken := value not equal to any securityLevelInfoCompleteList.token.
5. ONVIF client invokes GetSecurityLevels with parameters
   - Token list := invalidToken
6. The DUT responds with GetSecurityLevelsResponse message with parameters
   - SecurityLevel list := securityLevelsList
7. If securityLevelsList is not empty, FAIL the test, restore the DUT state, and skip other steps.
8. If cap.MaxLimit is less than 2, go to step 14.
9. ONVIF client invokes GetSecurityLevelInfo with parameters
   - Token[0] := invalidToken
• Token[1] := securityLevelInfoCompleteList[0].token

10. The DUT responds with **GetSecurityLevelInfoResponse** message with parameters

• SecurityLevelInfo list =: securityLevelsList

11. If securityLevelsList is empty, FAIL the test, restore the DUT state, and skip other steps.

12. If securityLevelsList contains more than one item, FAIL the test, restore the DUT state, and skip other steps.

13. If securityLevelsList[0].token is not equal to securityLevelInfoCompleteList[0].token, FAIL the test, restore the DUT state, and skip other steps.

14. Remove all security levels with tokens from createdSecurityLevelTokensList.

**Test Result:**

PASS –    
• The DUT passed all assertions.

FAIL –    
• The DUT did not send **GetSecurityLevelsResponse** message.

### 5.6.6 GET SECURITY LEVEL - TOO MANY ITEMS

**Test Case ID:** AUTH_BEHAVIOR-6-1-6

**Specification Coverage:** SecurityLevel (ONVIF Authentication Behavior Service Specification), GetSecurityLevels command (ONVIF Authentication Behavior Service Specification)

**Feature Under Test:** GetSecurityLevels

**WSDL Reference:** authenticationbehavior.wsdl

**Test Purpose:** To verify Get Security Level in case there are more items than MaxLimit in request.

**Pre-Requisite:** Authentication Behavior Service is received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**

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

3. ONVIF Client creates number of security levels by following the procedure mentioned in Annex A.24 with the following input and output parameters
   
   - out securityLevelInfoCompleteList - complete list of security levels information
   - out createdSecurityLevelTokensList - list of created security levels tokens
   - out cap - Authentication Behavior Service capabilities

4. If securityLevelCompleteList.token items number is less than cap.MaxLimit or equal to cap.MaxLimit, go to step 8.

5. Set tokenList := [subset of securityLevelInfoCompleteList.token values with items number equal to cap.MaxLimit + 1].

6. ONVIF client invokes GetSecurityLevels with parameters
   
   - Token list := tokenList


8. Remove all security levels with tokens from createdSecurityLevelTokensList.

Test Result:

PASS –

   - The DUT passed all assertions.

FAIL –

   - The DUT did not send env:Sender/ter:InvalidArgs/ter:TooManyItems SOAP 1.2 fault

5.7 Security Level Management

5.7.1 CREATE SECURITY LEVEL WITHOUT RECOGNITION GROUPS

Test Case ID: AUTH_BEHAVIOR-7-1-1

**Feature Under Test:** CreateSecurityLevel

**WSDL Reference:** authenticationbehavior.wsdl, event.wsdl

**Test Purpose:** To verify creation of security level without any recognition groups and generating of appropriate notifications.

**Pre-Requisite:** Authentication Behavior Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional Security Level.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - out securityLevelInfoInitialList - complete list of security levels information
4. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - in "tns1:Configuration/SecurityLevel/Changed" - Notification Topic
   - out s - Subscription reference
   - out currentTime - current time for the DUT
   - out terminationTime - Subscription termination time
5. ONVIF client invokes CreateSecurityLevel with parameters
   - SecurityLevel.token := ""
   - SecurityLevel.Name := "Test Name"
   - SecurityLevel.Priority := other then specified for SecurityLevelInfo items in securityLevelInfoInitialList
   - SecurityLevel.Description := "Test Description"
   - SecurityLevel.RecognitionGroup is skipped
6. The DUT responds with CreateSecurityLevelResponse message with parameters
7. ONVIF Client retrieves and checks `tns1:Configuration/SecurityLevel/Changed` event for the specified Authentication Profile token by following the procedure mentioned in Annex A.27 with the following input and output parameters

- in **s** - Subscription reference
- in **currentTime** - current time for the DUT
- in **terminationTime** - subscription termination time
- in **securityLevelToken** - Security Level token

8. ONVIF Client retrieves a security level by following the procedure mentioned in Annex A.30 with the following input and output parameters

- in **securityLevelToken** - security level token
- out **securityLevelsList** - security level list

9. If `securityLevelsList[0]` item does not have equal field values to values from step 5, FAIL the test, restore the DUT state, and skip other steps.

10. ONVIF Client retrieves a security level information by following the procedure mentioned in Annex A.29 with the following input and output parameters

- in **securityLevelToken** - security level token
- out **securityLevelInfoList** - security level information list

11. If `securityLevelInfoList[0]` item does not have equal field values to values from step 5, FAIL the test, restore the DUT state, and skip other steps.

12. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters

- out **securityLevelInfoCompleteList** - complete list of security levels information

13. If `securityLevelInfoCompleteList` does not have SecurityLevelInfo[token = **securityLevelToken**] item with equal field values to values from step 5, FAIL the test, restore the DUT state, and skip other steps.

14. ONVIF Client retrieves a complete list of security levels by following the procedure mentioned in Annex A.25 with the following input and output parameters

- out **securityLevelCompleteList** - complete list of security levels

---

www.onvif.org
15. If `securityLevelCompleteList` does not have `SecurityLevel[token = securityLevelToken]` item with equal field values to values from step 5, FAIL the test, restore the DUT state, and skip other steps.

16. For each `SecurityLevelInfo.token` (token) from `securityLevelInfoInitialList` do the following:

   16.1. If `securityLevelCompleteList` does not have `SecurityLevel[token = token]` item, FAIL the test, restore the DUT state, and skip other steps.

17. ONVIF Client deletes a security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

   - in `securityLevelToken` - security level token

Test Result:

PASS –

   • The DUT passed all assertions.

FAIL –

   • The DUT did not send `CreateSecurityLevelResponse` message.

**Note:** The following fields are compared at steps 9 and 13:

- SecurityLevel:
  - token
  - Name
  - Priority
  - Description
  - RecognitionGroup list
  - RecognitionMethod list
  - RecognitionType
  - Order

**Note:** The following fields are compared at step 11 and 15:

- SecurityLevelInfo:
  - token
5.7.2 CREATE SECURITY LEVEL WITHOUT RECOGNITION METHODS

Test Case ID: AUTH_BEHAVIOR-7-1-2


Feature Under Test: CreateSecurityLevel

WSDL Reference: authenticationbehavior.wsdl, event.wsdl

Test Purpose: To verify creation of security level without any recognition methods and generating of appropriate notifications.

Pre-Requisite: Authentication Behavior Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional Security Level.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - out securityLevelInfoInitialList - complete list of security levels information

4. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - in "tns1:Configuration/SecurityLevel/Changed" - Notification Topic
   - out s - Subscription reference
• out *currentTime* - current time for the DUT

• out *terminationTime* - Subscription termination time

5. ONVIF client invokes **createSecurityLevel** with parameters

   • SecurityLevel.token := ""

   • SecurityLevel.Name := "Test Name"

   • SecurityLevel.Priority := other than specified for SecurityLevelInfo items in securityLevelInfoInitialList

   • SecurityLevel.Description := "Test Description"

   • SecurityLevel.RecognitionGroup[0]

   • SecurityLevel.RecognitionGroup[0].RecognitionMethod is skipped

6. The DUT responds with **CreateSecurityLevelResponse** message with parameters

   • Token =: securityLevelToken

7. ONVIF Client retrieves and checks **tns1:Configuration/SecurityLevel/Changed** event for the specified Authentication Profile token by following the procedure mentioned in Annex A.27 with the following input and output parameters

   • in s - Subscription reference

   • in *currentTime* - current time for the DUT

   • in *terminationTime* - subscription termination time

   • in *securityLevelToken* - Security Level token

8. ONVIF Client retrieves a security level by following the procedure mentioned in Annex A.30 with the following input and output parameters

   • in *securityLevelToken* - security level token

   • out *securityLevelsList* - security level list

9. If *securityLevelsList[0]* item does not have equal field values to values from step 5, FAIL the test, restore the DUT state, and skip other steps.

10. ONVIF Client retrieves a security level information by following the procedure mentioned in Annex A.29 with the following input and output parameters
• in `securityLevelToken` - security level token

• out `securityLevelInfoList` - security level information list

11. If `securityLevelInfoList[0]` item does not have equal field values to values from step 5, FAIL the test, restore the DUT state, and skip other steps.

12. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters

• out `securityLevelInfoCompleteList` - complete list of security levels information

13. If `securityLevelInfoCompleteList` does not have `SecurityLevelInfo[token = securityLevelToken]` item with equal field values to values from step 5, FAIL the test, restore the DUT state, and skip other steps.

14. ONVIF Client retrieves a complete list of security levels by following the procedure mentioned in Annex A.25 with the following input and output parameters

• out `securityLevelCompleteList` - complete list of security levels

15. If `securityLevelCompleteList` does not have `SecurityLevel[token = securityLevelToken]` item with equal field values to values from step 5, FAIL the test, restore the DUT state, and skip other steps.

16. For each `SecurityLevelInfo.token (token)` from `securityLevelInfoInitialList` do the following:

  16.1. If `securityLevelCompleteList` does not have `SecurityLevel[token = token]` item, FAIL the test, restore the DUT state, and skip other steps.

17. ONVIF Client deletes a security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

• in `securityLevelToken` - security level token

Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send `CreateSecurityLevelResponse` message.

Note: The following fields are compared at steps 9 and 13:

• SecurityLevel:
• token
• Name
• Priority
• Description
• RecognitionGroup list
  • RecognitionMethod list
    • RecognitionType
    • Order

**Note:** The following fields are compared at step 11 and 15:

• SecurityLevelInfo:
  • token
  • Name
  • Priority
  • Description

### 5.7.3 CREATE SECURITY LEVEL WITH RECOGNITION METHODS

**Test Case ID:** AUTH_BEHAVIOR-7-1-3


**Feature Under Test:** CreateSecurityLevel

**WSDL Reference:** authenticationbehavior.wsdl, event.wsdl

**Test Purpose:** To verify creation of security level and generating of appropriate notifications.

**Pre-Requisite:** Authentication Behavior Service is received from the DUT. Schedule Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional Security Level.
**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - Out `securityLevelInfoInitialList` - complete list of security levels information

4. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - In "tns1:Configuration/SecurityLevel/Changed" - Notification Topic
   - Out `s` - Subscription reference
   - Out `currentTime` - current time for the DUT
   - Out `terminationTime` - Subscription termination time

5. ONVIF client invokes **CreateSecurityLevel** with parameters
   - SecurityLevel.token := ""
   - SecurityLevel.Name := "Test Name"
   - SecurityLevel.Priority := other then specified for SecurityLevelInfo items in `securityLevelInfoInitialList`
   - SecurityLevel.Description := "Test Description"
   - SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].RecognitionType := TODO (clarification)
   - SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].Order := 1
   - SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].Extension is skipped
   - SecurityLevel.Extension is skipped

6. The DUT responds with **CreateSecurityLevelResponse** message with parameters
   - Token =: `securityLevelToken`
7. ONVIF Client retrieves and checks `tns1:Configuration/SecurityLevel/Changed` event for the specified Security Level token by following the procedure mentioned in Annex A.27 with the following input and output parameters

   • in `s` - Subscription reference
   • in `currentTime` - current time for the DUT
   • in `terminationTime` - subscription termination time
   • in `securityLevelToken` - Security Level token

8. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.9 with the following input and output parameters

   • in `s` - Subscription reference

9. ONVIF Client retrieves a security level by following the procedure mentioned in Annex A.30 with the following input and output parameters

   • in `securityLevelToken` - security level token
   • out `securityLevelsList` - security level list

10. If `securityLevelsList[0]` item does not have equal field values to values from step 5, FAIL the test, restore the DUT state, and skip other steps.

11. ONVIF Client retrieves a security level information by following the procedure mentioned in Annex A.29 with the following input and output parameters

   • in `securityLevelToken` - security level token
   • out `securityLevelInfoList` - security level information list

12. If `securityLevelInfoList[0]` item does not have equal field values to values from step 5, FAIL the test, restore the DUT state, and skip other steps.

13. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters

   • out `securityLevelInfoCompleteList` - complete list of security levels information

14. If `securityLevelInfoCompleteList` does not have SecurityLevelInfo[token = `securityLevelToken`] item with equal field values to values from step 5, FAIL the test, restore the DUT state, and skip other steps.

15. ONVIF Client retrieves a complete list of security levels by following the procedure mentioned in Annex A.25 with the following input and output parameters
• out securityLevelCompleteList - complete list of security levels

16. If securityLevelCompleteList does not have SecurityLevel[token = securityLevelToken] item with equal field values to values from step 5, FAIL the test, restore the DUT state, and skip other steps.

17. For each SecurityLevelInfo.token (token) from securityLevelInfoInitialList do the following:

17.1. If securityLevelCompleteList does not have SecurityLevel[token = token] item, FAIL the test, restore the DUT state, and skip other steps.

18. ONVIF Client deletes a security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

• in securityLevelToken - security level token

Test Result:

PASS –
• The DUT passed all assertions.

FAIL –
• The DUT did not send CreateSecurityLevelResponse message.

Note: The following fields are compared at steps 10 and 14:

• SecurityLevel:
  • token
  • Name
  • Priority
  • Description
  • RecognitionGroup list
  • RecognitionMethod list
  • RecognitionType
  • Order

Note: The following fields are compared at step 12 and 16:

• SecurityLevelInfo:
• token
• Name
• Priority
• Description

5.7.4 MODIFY SECURITY LEVEL

Test Case ID: AUTH_BEHAVIOR-7-1-4


Feature Under Test: ModifySecurityLevel

WSDL Reference: authenticationbehavior.wsdl, event.wsdl

Test Purpose: To verify modifying of security level and generating of appropriate notifications.

Pre-Requisite: Authentication Behavior Service is received from the DUT. Schedule Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional Security Level.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.2 with the following input and output parameters
   • out cap - Authentication Behavior Service capabilities
4. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters
   • out securityLevelInfoInitialList - complete list of security levels information
5. ONVIF Client creates Security Level by following the procedure mentioned in Annex A.17 with the following input and output parameters
   • out securityLevelToken - security level token
6. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters

- in "tns1:Configuration/SecurityLevel/Changed" - Notification Topic
- out s - Subscription reference
- out currentTime - current time for the DUT
- out terminationTime - Subscription termination time

7. ONVIF client invokes **ModifySecurityLevel** with parameters

- SecurityLevel.token := securityLevelToken
- SecurityLevel.Name := "Test Name2"
- SecurityLevel.Priority := other then specified for SecurityLevelInfo items in securityLevelInfoInitialList and other than securityLevel.Priority
- SecurityLevel.Description := "Test Description2"
- SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].RecognitionType := TODO (clarification)
- SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].Order := 2
- SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].Extension is skipped
- SecurityLevel.RecognitionGroup[0].Extension is skipped
- SecurityLevel.Extension is skipped

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

9. ONVIF Client retrieves and checks tns1:Configuration/SecurityLevel/Changed event for the specified Security Level token by following the procedure mentioned in Annex A.27 with the following input and output parameters

- in s - Subscription reference
- in currentTime - current time for the DUT
- in terminationTime - subscription termination time
- in securityLevelToken - Security Level token
10. ONVIF Client retrieves a security level by following the procedure mentioned in Annex A.30 with the following input and output parameters:

- in `securityLevelToken` - security level token
- out `securityLevelsList` - security level list

11. If `securityLevelsList[0]` item does not have equal field values to values from step 7, FAIL the test, restore the DUT state, and skip other steps.

12. ONVIF Client retrieves a security level information by following the procedure mentioned in Annex A.29 with the following input and output parameters:

- in `securityLevelToken` - security level token
- out `securityLevelInfoList` - security level information list

13. If `securityLevelInfoList[0]` item does not have equal field values to values from step 7, FAIL the test, restore the DUT state, and skip other steps.

14. ONVIF client invokes `ModifySecurityLevel` with parameters:

- `SecurityLevel.token := securityLevelToken`
- `SecurityLevel.Name := "Test Name2"`
- `SecurityLevel.Priority := other then specified for SecurityLevelInfo items in securityLevelInfoInitialList and other than securityLevel.Priority`
- `SecurityLevel.Description := "Test Description2"`
- `SecurityLevel.RecognitionGroup is skipped`
- `SecurityLevel.Extension is skipped`

15. The DUT responds with `ModifySecurityLevelResponse` message.

16. ONVIF Client retrieves and checks `tns1:Configuration/SecurityLevel/Changed` event for the specified Security Level token by following the procedure mentioned in Annex A.27 with the following input and output parameters:

- in `s` - Subscription reference
- in `currentTime` - current time for the DUT
- in `terminationTime` - subscription termination time
- in `securityLevelToken` - Security Level token
17. ONVIF Client retrieves a security level by following the procedure mentioned in Annex A.30 with the following input and output parameters

- in `securityLevelToken` - security level token
- out `securityLevelsList` - security level list

18. If `securityLevelsList[0]` item does not have equal field values to values from step 14, FAIL the test, restore the DUT state, and skip other steps.

19. ONVIF Client retrieves a security level information by following the procedure mentioned in Annex A.29 with the following input and output parameters

- in `securityLevelToken` - security level token
- out `securityLevelInfoList` - security level information list

20. If `securityLevelInfoList[0]` item does not have equal field values to values from step 14, FAIL the test, restore the DUT state, and skip other steps.

21. If `cap.MaxRecognitionGroupsPerSecurityLevel` > 1:

21.1. ONVIF client invokes `ModifySecurityLevel` with parameters

- SecurityLevel.token := `securityLevelToken`
- SecurityLevel.Name := "Test Name3"
- SecurityLevel.Priority := other than specified for SecurityLevelInfo items in `securityLevelInfoInitialList` and other than `securityLevel.Priority`
- SecurityLevel.Description := "Test Description3"
- SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].RecognitionType := TODO (clarification)
- SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].Order := 3
- SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].Extension is skipped
- SecurityLevel.RecognitionGroup[0].Extension is skipped
- SecurityLevel.RecognitionGroup[1].RecognitionMethod[0].RecognitionType := TODO (clarification)
- SecurityLevel.RecognitionGroup[1].RecognitionMethod[0].Order := 1
- SecurityLevel.RecognitionGroup[1].RecognitionMethod[0].Extension is skipped
• SecurityLevel.RecognitionGroup[1].Extension is skipped
• SecurityLevel.Extension is skipped

21.2. The DUT responds with ModifySecurityLevelResponse message.

21.3. ONVIF Client retrieves and checks tns1:Configuration/SecurityLevel/Changed event for the specified Security Level token by following the procedure mentioned in Annex A.27 with the following input and output parameters

• in s - Subscription reference
• in currentTime - current time for the DUT
• in terminationTime - subscription termination time
• in securityLevelToken - Security Level token

21.4. ONVIF Client retrieves a security level by following the procedure mentioned in Annex A.30 with the following input and output parameters

• in securityLevelToken - security level token
• out securityLevelsList - security level list

21.5. If securityLevelsList[0] item does not have equal field values to values from step 21.1, FAIL the test, restore the DUT state, and skip other steps.

21.6. ONVIF Client retrieves a security level information by following the procedure mentioned in Annex A.29 with the following input and output parameters

• in securityLevelToken - security level token
• out securityLevelInfoList - security level information list

21.7. If securityLevelInfoList[0] item does not have equal field values to values from step 21.1, FAIL the test, restore the DUT state, and skip other steps.

22. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.9 with the following input and output parameters

• in s - Subscription reference

23. ONVIF Client retrieves a complete list of security level by following the procedure mentioned in Annex A.25 with the following input and output parameters

• out securityLevelUpdatedList - complete list of security levels information
24. If `securityLevelUpdatedList` does not have `SecurityLevel[token = securityLevelToken]` item, FAIL the test, restore the DUT state, and skip other steps.

25. For each `SecurityLevel.token (token)` from `securityLevelInitialList` do the following:

   25.1. If `securityLevelUpdatedList` does not have `SecurityLevel[token = token]` item, FAIL the test, restore the DUT state, and skip other steps.

26. ONVIF Client deletes a security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

   - in `securityLevelToken` - security level token

Test Result:

PASS –

   • The DUT passed all assertions.

FAIL –

   • The DUT did not send `ModifySecurityLevelResponse` message.

Note: The following fields are compared at steps 11, 18, and 21.5:

   • SecurityLevel:
     • token
     • Name
     • Priority
     • Description
     • RecognitionGroup list
       • RecognitionMethod list
         • RecognitionType
         • Order

Note: The following fields are compared at step 13, 20, and 21.7:

   • SecurityLevelInfo:
     • token
     • Name
5.7.5 DELETE SECURITY LEVEL

Test Case ID: AUTH_BEHAVIOR-7-1-5


Feature Under Test: DeleteSecurityLevel

WSDL Reference: authenticationbehavior.wsdl, event.wsdl

Test Purpose: To verify deleting of security level and generating of appropriate notifications.

Pre-Requisite: Authentication Behavior Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional Security Level.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves a complete list of security levels by following the procedure mentioned in Annex A.25 with the following input and output parameters
   • out securityLevelInitialList - complete list of security levels
4. ONVIF Client creates Security Level by following the procedure mentioned in Annex A.17 with the following input and output parameters
   • out securityLevelToken - security level token
   • out securityLevel - security level
5. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters
   • in "tns1:Configuration/SecurityLevel/Removed" - Notification Topic
   • out s - Subscription reference
6. ONVIF client invokes **DeleteSecurityLevel** with parameters
   - SecurityLevel.token := securityLevelToken

7. The DUT responds with **DeleteSecurityLevelResponse** message.

8. ONVIF Client retrieves and checks **tns1:Configuration/SecurityLevel/Removed** event for the specified Security Level token by following the procedure mentioned in Annex A.28 with the following input and output parameters
   - in s - Subscription reference
   - in currentTime - current time for the DUT
   - in terminationTime - subscription termination time
   - in securityLevelToken - Security Level token

9. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.9 with the following input and output parameters
   - in s - Subscription reference

10. ONVIF Client retrieves a security level by following the procedure mentioned in Annex A.30 with the following input and output parameters
    - in securityLevelToken - security level token
    - out securityLevelsList - security level list

11. If **securityLevelsList** is not empty, FAIL the test, restore the DUT state, and skip other steps.

12. ONVIF Client retrieves a security level information by following the procedure mentioned in Annex A.29 with the following input and output parameters
    - in securityLevelToken - security level token
    - out securityLevelInfoList - security level information list

13. If **securityLevelInfoList** is not empty, FAIL the test, restore the DUT state, and skip other steps.

14. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters
• out securityLevelInfoList - complete list of security levels information

15. If securityLevelInfoList contains SecurityLevelInfo[token = securityLevelToken] item, FAIL the test, restore the DUT state, and skip other steps.

16. ONVIF Client retrieves a complete list of security levels by following the procedure mentioned in Annex A.25 with the following input and output parameters

• out securityLevelList - complete list of security levels

17. If securityLevelList contains SecurityLevel[token = securityLevelToken] item, FAIL the test, restore the DUT state, and skip other steps.

18. For each SecurityLevel.token (token) from securityLevelInitialList do the following:

18.1. If securityLevelList does not have SecurityLevel[token = token] item, FAIL the test, restore the DUT state, and skip other steps.

Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send DeleteSecurityLevelResponse message.

5.7.6 SET SECURITY LEVEL WITHOUT RECOGNITION GROUPS

Test Case ID: AUTH_BEHAVIOR-7-1-6


Feature Under Test: SetSecurityLevel

WSDL Reference: authenticationbehavior.wsdl, event.wsdl

Test Purpose: To verify creation of security level without any recognition groups and generating of appropriate notifications.

Pre-Requisite: Authentication Behavior Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional Security Level.
Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - out securityLevelInfoInitialList - complete list of security levels information

4. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - in "tns1:Configuration/SecurityLevel/Changed" - Notification Topic
   - out s - Subscription reference
   - out currentTime - current time for the DUT
   - out terminationTime - Subscription termination time

5. Set securityLevelToken := token that differs from tokens listed in securityLevelInfoInitialList.

6. ONVIF client invokes SetSecurityLevel with parameters
   - SecurityLevel.token := securityLevelToken
   - SecurityLevel.Name := "Test Name"
   - SecurityLevel.Priority := other then specified for SecurityLivelInfo items in securityLevelInfoInitialList
   - SecurityLevel.Description := "Test Description"
   - SecurityLevel.RecognitionGroup is skipped

7. The DUT responds with SetSecurityLevelResponse message with parameters
   - Token =: securityLevelToken

8. ONVIF Client retrieves and checks tns1:Configuration/SecurityLevel/Changed event for the specified Authentication Profile token by following the procedure mentioned in Annex A.27 with the following input and output parameters
   - in s - Subscription reference
• in current\textit{Time} - current time for the DUT

• in termination\textit{Time} - subscription termination time

• in security\textit{Level}Token - Security Level token

9. ONVIF Client retrieves a security level by following the procedure mentioned in Annex A.30 with the following input and output parameters

• in security\textit{Level}Token - security level token

• out security\textit{Levels}List - security level list

10. If security\textit{Levels}List[0] item does not have equal field values to values from step 6, FAIL the test, restore the DUT state, and skip other steps.

11. ONVIF Client retrieves a security level information by following the procedure mentioned in Annex A.29 with the following input and output parameters

• in security\textit{Level}Token - security level token

• out security\textit{Level}InfoList - security level information list

12. If security\textit{Level}InfoList[0] item does not have equal field values to values from step 6, FAIL the test, restore the DUT state, and skip other steps.

13. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters

• out security\textit{Level}InfoCompleteList - complete list of security levels information

14. If security\textit{Level}InfoCompleteList does not have Security\textit{Level}Info[token = security\textit{Level}Token] item with equal field values to values from step 6, FAIL the test, restore the DUT state, and skip other steps.

15. ONVIF Client retrieves a complete list of security levels by following the procedure mentioned in Annex A.25 with the following input and output parameters

• out security\textit{Level}CompleteList - complete list of security levels

16. If security\textit{Level}CompleteList does not have Security\textit{Level}[token = security\textit{Level}Token] item with equal field values to values from step 6, FAIL the test, restore the DUT state, and skip other steps.

17. For each Security\textit{Level}Info.token (token) from security\textit{Level}Info\textit{Initial}List do the following:
17.1. If `securityLevelCompleteList` does not have `SecurityLevel[token = token]` item, FAIL the test, restore the DUT state, and skip other steps.

18. ONVIF Client deletes a security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

- in `securityLevelToken` - security level token

Test Result:

PASS –

- The DUT passed all assertions.

FAIL –

- The DUT did not send `SetSecurityLevelResponse` message.

Note: The following fields are compared at steps 10 and 14:

- SecurityLevel:
  - token
  - Name
  - Priority
  - Description
  - RecognitionGroup list
    - RecognitionMethod list
      - RecognitionType
      - Order

Note: The following fields are compared at step 12 and 16:

- SecurityLevelInfo:
  - token
  - Name
  - Priority
5.7.7 SET SECURITY LEVEL WITHOUT RECOGNITION METHODS

Test Case ID: AUTH_BEHAVIOR-7-1-7


Feature Under Test: SetSecurityLevel

WSDL Reference: authenticationbehavior.wsdl, event.wsdl

Test Purpose: To verify creation of security level without any recognition methods and generating of appropriate notifications.

Pre-Requisite: Authentication Behavior Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional Security Level.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters
   • out securityLevelInfoInitialList - complete list of security levels information

4. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters
   • in "tns1:Configuration/SecurityLevel/Changed" - Notification Topic
   • out s - Subscription reference
   • out currentTime - current time for the DUT
   • out terminationTime - Subscription termination time
5. Set `securityLevelToken` := token that differs from tokens listed in `securityLevelInfoInitialList`.

6. ONVIF client invokes `SetSecurityLevel` with parameters

   - `SecurityLevel.token` := `securityLevelToken`

   - `SecurityLevel.Name` := "Test Name"

   - `SecurityLevel.Priority` := other then specified for `SecurityLevelInfo` items in `securityLevelInfoInitialList`

   - `SecurityLevel.Description` := "Test Description"

   - `SecurityLevel.RecognitionGroup[0]`

   - `SecurityLevel.RecognitionGroup[0].RecognitionMethod` is skipped

7. The DUT responds with `SetSecurityLevelResponse` message with parameters

   - `Token` := `securityLevelToken`

8. ONVIF Client retrieves and checks `tns1:Configuration/SecurityLevel/Changed` event for the specified Authentication Profile token by following the procedure mentioned in Annex A.27 with the following input and output parameters

   - in `s` - Subscription reference

   - in `currentTime` - current time for the DUT

   - in `terminationTime` - subscription termination time

   - in `securityLevelToken` - Security Level token

9. ONVIF Client retrieves a security level by following the procedure mentioned in Annex A.30 with the following input and output parameters

   - in `securityLevelToken` - security level token

   - out `securityLevelsList` - security level list

10. If `securityLevelsList[0]` item does not have equal field values to values from step 6, FAIL the test, restore the DUT state, and skip other steps.

11. ONVIF Client retrieves a security level information by following the procedure mentioned in Annex A.29 with the following input and output parameters

    - in `securityLevelToken` - security level token
• out securityLevelInfoList - security level information list

12. If securityLevelInfoList[0] item does not have equal field values to values from step 6, FAIL the test, restore the DUT state, and skip other steps.

13. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters

• out securityLevelInfoCompleteList - complete list of security levels information

14. If securityLevelInfoCompleteList does not have SecurityLevelInfo[token = securityLevelToken] item with equal field values to values from step 6, FAIL the test, restore the DUT state, and skip other steps.

15. ONVIF Client retrieves a complete list of security levels by following the procedure mentioned in Annex A.25 with the following input and output parameters

• out securityLevelCompleteList - complete list of security levels

16. If securityLevelCompleteList does not have SecurityLevel[token = securityLevelToken] item with equal field values to values from step 6, FAIL the test, restore the DUT state, and skip other steps.

17. For each SecurityLevelInfo.token (token) from securityLevelInfoInitialList do the following:

   17.1. If securityLevelCompleteList does not have SecurityLevel[token = token] item, FAIL the test, restore the DUT state, and skip other steps.

18. ONVIF Client deletes a security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

• in securityLevelToken - security level token

Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send SetSecurityLevelResponse message.

Note: The following fields are compared at steps 10 and 14:

• SecurityLevel:
  • token
5.7.8 SET SECURITY LEVEL WITH RECOGNITION METHODS

Test Case ID: AUTH_BEHAVIOR-7-1-8


Feature Under Test: SetSecurityLevel

WSDL Reference: authenticationbehavior.wsdl, event.wsdl

Test Purpose: To verify creation of security level and generating of appropriate notifications.

Pre-Requisite: Authentication Behavior Service is received from the DUT. Schedule Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional Security Level.

Test Configuration: ONVIF Client and DUT
Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - out `securityLevelInfoInitialList` - complete list of security levels information

4. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - in "tns1:Configuration/SecurityLevel/Changed" - Notification Topic
   - out `s` - Subscription reference
   - out `currentTime` - current time for the DUT
   - out `terminationTime` - Subscription termination time

5. Set `securityLevelToken := token that differs from tokens listed in securityLevelInfoInitialList`.

6. ONVIF client invokes `SetSecurityLevel` with parameters
   - SecurityLevel.token := `securityLevelToken`
   - SecurityLevel.Name := "Test Name"
   - SecurityLevel.Priority := other then specified for SecurityLevelInfo items in `securityLevelInfoInitialList`
   - SecurityLevel.Description := "Test Description"
   - SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].RecognitionType := TODO (clarification)
   - SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].Order := 1
   - SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].Extension is skipped
   - SecurityLevel.RecognitionGroup[0].Extension is skipped
   - SecurityLevel.Extension is skipped

7. The DUT responds with `SetSecurityLevelResponse` message with parameters
   - Token =: `securityLevelToken`
8. ONVIF Client retrieves and checks \texttt{tns1:Configuration/SecurityLevel/Changed} event for the specified Security Level token by following the procedure mentioned in Annex A.27 with the following input and output parameters
   - in \texttt{s} - Subscription reference
   - in \texttt{currentTime} - current time for the DUT
   - in \texttt{terminationTime} - subscription termination time
   - in \texttt{securityLevelToken} - Security Level token

9. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.9 with the following input and output parameters
   - in \texttt{s} - Subscription reference

10. ONVIF Client retrieves a security level by following the procedure mentioned in Annex A.30 with the following input and output parameters
    - in \texttt{securityLevelToken} - security level token
    - out \texttt{securityLevelsList} - security level list

11. If \texttt{securityLevelsList}[0] item does not have equal field values to values from step 6, FAIL the test, restore the DUT state, and skip other steps.

12. ONVIF Client retrieves a security level information by following the procedure mentioned in Annex A.29 with the following input and output parameters
    - in \texttt{securityLevelToken} - security level token
    - out \texttt{securityLevelInfoList} - security level information list

13. If \texttt{securityLevelInfoList}[0] item does not have equal field values to values from step 6, FAIL the test, restore the DUT state, and skip other steps.

14. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters
    - out \texttt{securityLevelInfoCompleteList} - complete list of security levels information

15. If \texttt{securityLevelInfoCompleteList} does not have SecurityLevelInfo[token = \texttt{securityLevelToken}] item with equal field values to values from step 6, FAIL the test, restore the DUT state, and skip other steps.

16. ONVIF Client retrieves a complete list of security levels by following the procedure mentioned in Annex A.25 with the following input and output parameters
17. If `securityLevelCompleteList` does not have `SecurityLevel[token = securityLevelToken]` item with equal field values to values from step 6, FAIL the test, restore the DUT state, and skip other steps.

18. For each `SecurityLevelInfo.token (token)` from `securityLevelInfoInitialList` do the following:

18.1. If `securityLevelCompleteList` does not have `SecurityLevel[token = token]` item, FAIL the test, restore the DUT state, and skip other steps.

19. ONVIF Client deletes a security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

- `in securityLevelToken` - security level token

Test Result:

PASS –

- The DUT passed all assertions.

FAIL –

- The DUT did not send `SetSecurityLevelResponse` message.

Note: The following fields are compared at steps 11 and 15:

- `SecurityLevel`:
  - token
  - Name
  - Priority
  - Description
  - `RecognitionGroup list`
  - `RecognitionMethod list`
  - `RecognitionType`
  - Order

Note: The following fields are compared at step 13 and 17:

- `SecurityLevelInfo`: 
5.7.9 SET SECURITY LEVEL

Test Case ID: AUTH_BEHAVIOR-7-1-9


Feature Under Test: SetSecurityLevel

WSDL Reference: authenticationbehavior.wsdl, event.wsdl

Test Purpose: To verify modifying of security level and generating of appropriate notifications.

Pre-Requisite: Authentication Behavior Service is received from the DUT. Schedule Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional Security Level.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.2 with the following input and output parameters
   - out cap - Authentication Behavior Service capabilities
4. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - out securityLevelInfoInitialList - complete list of security levels information
5. ONVIF Client creates Security Level by following the procedure mentioned in Annex A.17 with the following input and output parameters
   - out securityLevelToken - security level token
6. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters

- in "tns1:Configuration/SecurityLevel/Changed" - Notification Topic
- out s - Subscription reference
- out currentTime - current time for the DUT
- out terminationTime - Subscription termination time

7. ONVIF client invokes SetSecurityLevel with parameters

- SecurityLevel.token := securityLevelToken
- SecurityLevel.Name := "Test Name2"
- SecurityLevel.Priority := other than specified for SecurityLevelInfo items in securityLevelInfoInitialList and other than securityLevel.Priority
- SecurityLevel.Description := "Test Description2"
- SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].RecognitionType := TODO (clarification)
- SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].Order := 2
- SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].Extension is skipped
- SecurityLevel.RecognitionGroup[0].Extension is skipped
- SecurityLevel.Extension is skipped

8. The DUT responds with SetSecurityLevelResponse message.

9. ONVIF Client retrieves and checks tns1:Configuration/SecurityLevel/Changed event for the specified Security Level token by following the procedure mentioned in Annex A.27 with the following input and output parameters

- in s - Subscription reference
- in currentTime - current time for the DUT
- in terminationTime - subscription termination time
- in securityLevelToken - Security Level token
10. ONVIF Client retrieves a security level by following the procedure mentioned in Annex A.30 with the following input and output parameters
   - in `securityLevelToken` - security level token
   - out `securityLevelsList` - security level list

11. If `securityLevelsList[0]` item does not have equal field values to values from step 7, FAIL the test, restore the DUT state, and skip other steps.

12. ONVIF Client retrieves a security level information by following the procedure mentioned in Annex A.29 with the following input and output parameters
   - in `securityLevelToken` - security level token
   - out `securityLevelInfoList` - security level information list

13. If `securityLevelInfoList[0]` item does not have equal field values to values from step 7, FAIL the test, restore the DUT state, and skip other steps.

14. ONVIF client invokes `SetSecurityLevel` with parameters
   - `SecurityLevel.token := securityLevelToken`
   - `SecurityLevel.Name := "Test Name2"`
   - `SecurityLevel.Priority := other then specified for SecurityLiveInfo items in securityLevelInfoInitialList and other than securityLevel.Priority`
   - `SecurityLevel.Description := "Test Description2"`
   - `SecurityLevel.RecognitionGroup is skipped`
   - `SecurityLevel.Extension is skipped`

15. The DUT responds with `SetSecurityLevelResponse` message.

16. ONVIF Client retrieves and checks `tns1:Configuration/SecurityLevel/Changed` event for the specified Security Level token by following the procedure mentioned in Annex A.27 with the following input and output parameters
   - in `s` - Subscription reference
   - in `current Time` - current time for the DUT
   - in `terminationTime` - subscription termination time
   - in `securityLevelToken` - Security Level token
17. ONVIF Client retrieves a security level by following the procedure mentioned in Annex A.30 with the following input and output parameters

- in `securityLevelToken` - security level token
- out `securityLevelsList` - security level list

18. If `securityLevelsList[0]` item does not have equal field values to values from step 14, FAIL the test, restore the DUT state, and skip other steps.

19. ONVIF Client retrieves a security level information by following the procedure mentioned in Annex A.29 with the following input and output parameters

- in `securityLevelToken` - security level token
- out `securityLevelInfoList` - security level information list

20. If `securityLevelInfoList[0]` item does not have equal field values to values from step 14, FAIL the test, restore the DUT state, and skip other steps.

21. If `cap.MaxRecognitionGroupsPerSecurityLevel` > 1:

21.1. ONVIF client invokes `SetSecurityLevel` with parameters

- `SecurityLevel.token` := `securityLevelToken`
- `SecurityLevel.Name` := "Test Name3"
- `SecurityLevel.Priority` := other then specified for SecurityLevelInfo items in `securityLevelInfoInitialList` and other than `securityLevel.Priority`
- `SecurityLevel.Description` := "Test Description3"
- `SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].RecognitionType` := TODO (clarification)
- `SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].Order` := 3
- `SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].Extension` is skipped
- `SecurityLevel.RecognitionGroup[0].Extension` is skipped
- `SecurityLevel.RecognitionGroup[1].RecognitionMethod[0].RecognitionType` := TODO (clarification)
- `SecurityLevel.RecognitionGroup[1].RecognitionMethod[0].Order` := 1
- `SecurityLevel.RecognitionGroup[1].RecognitionMethod[0].Extension` is skipped
• SecurityLevel.RecognitionGroup[1].Extension is skipped

• SecurityLevel.Extension is skipped

21.2. The DUT responds with **SetSecurityLevelResponse** message.

21.3. ONVIF Client retrieves and checks **tns1:Configuration/SecurityLevel/Changed** event for the specified Security Level token by following the procedure mentioned in Annex A.27 with the following input and output parameters

• in **s** - Subscription reference

• in **currentTime** - current time for the DUT

• in **terminationTime** - subscription termination time

• in **securityLevelToken** - Security Level token

21.4. ONVIF Client retrieves a security level by following the procedure mentioned in Annex A.30 with the following input and output parameters

• in **securityLevelToken** - security level token

• out **securityLevelsList** - security level list

21.5. If **securityLevelsList[0]** item does not have equal field values to values from step 21.1, FAIL the test, restore the DUT state, and skip other steps.

21.6. ONVIF Client retrieves a security level information by following the procedure mentioned in Annex A.29 with the following input and output parameters

• in **securityLevelToken** - security level token

• out **securityLevelInfoList** - security level information list

21.7. If **securityLevelInfoList[0]** item does not have equal field values to values from step 21.1, FAIL the test, restore the DUT state, and skip other steps.

22. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.9 with the following input and output parameters

• in **s** - Subscription reference

23. ONVIF Client retrieves a complete list of security level by following the procedure mentioned in Annex A.25 with the following input and output parameters

• out **securityLevelUpdatedList** - complete list of security levels information
24. If `securityLevelUpdatedList` does not have `SecurityLevel[token = securityLevelToken]` item, FAIL the test, restore the DUT state, and skip other steps.

25. For each `SecurityLevel.token (token)` from `securityLevelInitialList` do the following:

   25.1. If `securityLevelUpdatedList` does not have `SecurityLevel[token = token]` item, FAIL the test, restore the DUT state, and skip other steps.

26. ONVIF Client deletes a security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

   • in `securityLevelToken` - security level token

Test Result:

PASS –

   • The DUT passed all assertions.

FAIL –

   • The DUT did not send `SetSecurityLevelResponse` message.

Note: The following fields are compared at steps 11, 18, and 21.5:

   • SecurityLevel:
     • token
     • Name
     • Priority
     • Description
     • RecognitionGroup list
     • RecognitionMethod list
     • RecognitionType
     • Order

Note: The following fields are compared at step 13, 20, and 21.7:

   • SecurityLevelInfo:
     • token
5.7.10 CREATE SECURITY LEVEL - NOT EMPTY TOKEN

Test Case ID: AUTH_BEHAVIOR-7-1-10

Specification Coverage: CreateSecurityLevel command (ONVIF Authentication Behavior Service Specification)

Feature Under Test: CreateSecurityLevel

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify create security level with not empty token.

Pre-Requisite: Authentication Behavior Service is received from the DUT. The DUT shall have enough free storage capacity for one additional SecurityLevel.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters

- out securityLevelInfoList - complete list of security levels information

4. ONVIF client invokes CreateSecurityLevel with parameters

- SecurityLevel.token := "Token"

- SecurityLevel.Name := "Test Name"

- SecurityLevel.Priority := other then specified for SecurityLivellInfo items in securityLevelInfoInitialList

- SecurityLevel.Description is skipped

- SecurityLevel.RecognitionGroup is skipped
• SecurityLevel.Extension is skipped

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

Test Result:

PASS –
• The DUT passed all assertions.

FAIL –
• The DUT did not send env:Sender/ter:InvalidArgVal SOAP 1.2 fault.

5.7.11 CREATE SECURITY LEVEL - CAPABILITY VIOLATED
(MAX RECOGNITION GROUPS PER SECURITY LEVEL)

Test Case ID: AUTH_BEHAVIOR-7-1-11

Specification Coverage: CreateSecurityLevel command (ONVIF Authentication Behavior Service Specification)

Feature Under Test: CreateSecurityLevel

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify creation of security level with maximum number of recognition groups per security level.

Pre-Requisite: Authentication Behavior Service is received from the DUT. The DUT shall have enough free storage capacity for one additional Security Level.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.2 with the following input and output parameters
   • out cap - Authentication Behavior Service capabilities

4. If cap.MaxRecognitionGroupsPerSecurityLevel value is more than 50, skip other steps.
5. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters
   • out securityLevelInfoList - complete list of security levels information

6. If cap.MaxRecognitionGroupsPerSecurityLevel is equal to one, go to step 11.

7. Set recognitionGroup :=
   • RecognitionMethod[0].RecognitionType := TODO (clarification)
   • RecognitionMethod[0].Order := 1
   • RecognitionMethod[0].Extension is skipped

8. ONVIF client invokes CreateSecurityLevel with parameters
   • SecurityLevel.token := ""
   • SecurityLevel.Name := "Test Name"
   • SecurityLevel.Priority := other then specified for SecurityLevelInfo items in securityLevelInfoInitialList
   • SecurityLevel.Description is skipped
   • SecurityLevel.RecognitionGroup list := recognitionGroup duplicated cap.MaxRecognitionGroupsPerSecurityLevel number of times
   • SecurityLevel.Extension is skipped

9. The DUT responds with CreateSecurityLevelResponse message with parameters
   • Token =: securityLevelToken

10. ONVIF Client deletes a security level by following the procedure mentioned in Annex A.22 with the following input and output parameters
    • in securityLevelToken - security level token

11. ONVIF client invokes CreateSecurityLevel with parameters
    • SecurityLevel.token := ""
    • SecurityLevel.Name := "Test Name"
    • SecurityLevel.Priority := other then specified for SecurityLevelInfo items in securityLevelInfoInitialList
• SecurityLevel.Description is skipped

• SecurityLevel.RecognitionGroup list := recognitionGroup duplicated cap.MaxRecognitionGroupsPerSecurityLevel + 1 number of times

• SecurityLevel.Extension is skipped


Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send CreateSecurityLevelResponse message.

• The DUT did not send env:Sender/ter:CapabilityViolated/ter:MaxRecognitionGroupsPerSecurityLevel SOAP 1.2 fault.

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

5.7.12 CREATE SECURITY LEVEL - CAPABILITY VIOLATED (MAX RECOGNITION METHODS PER RECOGNITION GROUP)

Test Case ID: AUTH_BEHAVIOR-7-1-12

Specification Coverage: CreateSecurityLevel command (ONVIF Authentication Behavior Service Specification)

Feature Under Test: CreateSecurityLevel

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify creation of security level with maximum number recognition methods per recognition group.

Pre-Requisite: Authentication Behavior Service is received from the DUT. The DUT shall have enough free storage capacity for one additional Security Level.

Test Configuration: ONVIF Client and DUT
Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.2 with the following input and output parameters
   - out cap - Authentication Behavior Service capabilities

4. If $\text{cap}.\text{MaxRecognitionMethodsPerRecognitionGroup}$ value is more than 50, skip other steps.

5. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - out securityLevelInfoList - complete list of security levels information

6. If $\text{cap}.\text{MaxRecognitionMethodsPerRecognitionGroup}$ is equal to one, go to step 11.

7. Set $\text{recognitionMethod} :=$
   - RecognitionType := TODO (clarification)
   - Order := 1
   - Extension is skipped

8. ONVIF client invokes CreateSecurityLevel with parameters
   - SecurityLevel.token := ""
   - SecurityLevel.Name := "Test Name"
   - SecurityLevel.Priority := other then specified for SecurityLevelInfo items in securityLevelInfoInitialList
   - SecurityLevel.Description is skipped
   - SecurityLevel.RecognitionGroup[0].RecognitionMethod list := $\text{recognitionMethod}$ duplicated $\text{cap}.\text{MaxRecognitionMethodsPerRecognitionGroup}$ number of times
   - SecurityLevel.RecognitionGroup[0].Extension is skipped
   - SecurityLevel.Extension is skipped

9. The DUT responds with CreateSecurityLevelResponse message with parameters

www.onvif.org
• Token =: securityLevelToken

10. ONVIF Client deletes a security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

• in securityLevelToken - security level token

11. ONVIF client invokes CreateSecurityLevel with parameters

• SecurityLevel.token := ""

• SecurityLevel.Name := "Test Name"

• SecurityLevel.Priority := other then specified for SecurityLevelInfo items in securityLevelInfoInitialList

• SecurityLevel.Description is skipped

• SecurityLevel.RecognitionGroup[0].RecognitionMethod list := recognitionMethod duplicated cap.MaxRecognitionMethodsPerRecognitionGroup + 1 number of times

• SecurityLevel.Extension is skipped

• SecurityLevel.RecognitionGroup[0].Extension is skipped


Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send CreateSecurityLevelResponse message.

• The DUT did not send env:Sender/ter:CapabilityViolated/ter:MaxRecognitionMethodsPerRecognitionGroup SOAP 1.2 fault.

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

5.7.13 CREATE SECURITY LEVEL - DUPLICATE PRIORITY

Test Case ID: AUTH_BEHAVIOR-7-1-13
**Specification Coverage**: CreateSecurityLevel command (ONVIF Authentication Behavior Service Specification)

**Feature Under Test**: CreateSecurityLevel

**WSDL Reference**: authenticationbehavior.wsdl

**Test Purpose**: To verify creation of security level with duplicated priority.

**Pre-Requisite**: Authentication Behavior Service is received from the DUT. The DUT shall have enough free storage capacity for one additional Security Level.

**Test Configuration**: ONVIF Client and DUT

**Test Sequence**:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.2 with the following input and output parameters
   - out cap - Authentication Behavior Service capabilities
4. If cap.MaxSecurityLevels = 1, skip other steps.
5. ONVIF Client find existing or create new security level by following the procedure mentioned in Annex A.5 with the following input and output parameters
   - out securityLevelToken - security level token
   - out securityLevel - security level
   - out newSecurityLevel - flag if new security level was created
6. ONVIF client invokes CreateSecurityLevel with parameters
   - SecurityLevel.token := ""
   - SecurityLevel.Name := "Test Name"
   - SecurityLevel.Description is skipped
   - SecurityLevel.RecognitionGroup is skipped
   - SecurityLevel.Extension is skipped

8. If `newSecurityLevel = true`:

   8.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

       • in `securityLevelToken` - security level token

Test Result:

PASS –

   • The DUT passed all assertions.

FAIL –

   • The DUT did not send `env:Sender/ter:InvalidArgVal/ter:DuplicatePriority` SOAP 1.2 fault.

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

**5.7.14 MODIFY SECURITY LEVEL - INVALID TOKEN**

Test Case ID: AUTH_BEHAVIOR-7-1-14

Specification Coverage: ModifySecurityLevel command (ONVIF Authentication Behavior Service Specification)

Feature Under Test: ModifySecurityLevel

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify modifying of security level with invalid token.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters
4. Set invalidToken := value not equal to any securityLevelInfoList.token

5. ONVIF client invokes ModifySecurityLevel with parameters

   • SecurityLevel.token := invalidToken
   • SecurityLevel.Name := "Test Name"
   • SecurityLevel.Priority := 0
   • SecurityLevel.Description is skipped
   • SecurityLevel.RecognitionGroup is skipped
   • SecurityLevel.Extension is skipped


Test Result:

PASS –

   • The DUT passed all assertions.

FAIL –

   • The DUT did not send env:Sender/ter:InvalidArgVal/ter:NotFound SOAP 1.2 fault

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

5.7.15 MODIFY SECURITY LEVEL - CAPABILITY VIOLATED (MAX RECOGNITION GROUPS PER SECURITY LEVEL)

Test Case ID: AUTH_BEHAVIOR-7-1-15

Specification Coverage: ModifySecurityLevel command (ONVIF Authentication Behavior Service Specification)

Feature Under Test: ModifySecurityLevel

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify modification of security level with maximum number of recognition groups per security level.
**Pre-Requisite:** Authentication Behavior Service is received from the DUT. The DUT shall have enough free storage capacity for one additional SecurityLevel.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.2 with the following input and output parameters
   - out cap - Authentication Behavior Service capabilities

4. If \( \text{cap}.\text{MaxRecognitionGroupsPerSecurityLevel} \) value is more than 50, skip other steps.

5. ONVIF Client creates Security Level by following the procedure mentioned in Annex A.17 with the following input and output parameters
   - out securityLevelToken - security level token
   - out securityLevel - security level

6. If \( \text{cap}.\text{MaxRecognitionGroupsPerSecurityLevel} \) is equal to one, go to step 10.

7. Set \( \text{recognitionGroup} := \)
   - RecognitionMethod[0].RecognitionType := TODO (clarification)
   - RecognitionMethod[0].Order := 1
   - RecognitionMethod[0].Extension is skipped

8. ONVIF client invokes **ModifySecurityLevel** with parameters
   - SecurityLevel.token := securityLevelToken
   - SecurityLevel.Name := "Test Name"
   - SecurityLevel.Description is skipped
   - SecurityLevel.RecognitionGroup list := recognitionGroup duplicated \( \text{cap}.\text{MaxRecognitionGroupsPerSecurityLevel} \) number of times
   - SecurityLevel.Extension is skipped

10. ONVIF client invokes `ModifySecurityLevel` with parameters

   • SecurityLevel.token := `securityLevelToken`
   • SecurityLevel.Name := "Test Name"
   • SecurityLevel.Priority := `securityLevel.Priority`
   • SecurityLevel.Description is skipped
   • SecurityLevel.RecognitionGroup list := `recognitionGroup` duplicated `cap.MaxRecognitionGroupsPerSecurityLevel + 1` number of times
   • SecurityLevel.Extension is skipped


12. ONVIF Client deletes a security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

   • in `securityLevelToken` - security level token

Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send `ModifySecurityLevelResponse` message.

• The DUT did not send `env:Sender/ter:CapabilityViolated/ter:MaxPoliciesPerSecurityLevel` SOAP 1.2 fault.

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

5.7.16 MODIFY SECURITY LEVEL - CAPABILITY VIOLATED (MAX RECOGNITION METHODS PER RECOGNITION GROUP)

Test Case ID: AUTH_BEHAVIOR-7-1-16
**Specification Coverage:** ModifySecurityLevel command (ONVIF Authentication Behavior Service Specification)

**Feature Under Test:** ModifySecurityLevel

**WSDL Reference:** authenticationbehavior.wsdl

**Test Purpose:** To verify modification of security level with maximum number recognition methods per recognition group.

**Pre-Requisite:** Authentication Behavior Service is received from the DUT. The DUT shall have enough free storage capacity for one additional SecurityLevel.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.2 with the following input and output parameters
   - out cap - Authentication Behavior Service capabilities

4. If cap.MaxRecognitionMethodsPerRecognitionGroup value is more than 50, skip other steps.

5. ONVIF Client creates Security Level by following the procedure mentioned in Annex A.17 with the following input and output parameters
   - out securityLevelToken - security level token
   - out securityLevel - security level

6. If cap.MaxRecognitionMethodsPerRecognitionGroup is equal to one, go to step 10.

7. Set recognitionMethod :
   - RecognitionType := TODO (clarification)
   - Order := 1
   - Extension is skipped

8. ONVIF client invokes **ModifySecurityLevel** with parameters
ONVIF Authentication Behavior Device Test Specification Version 18.12

• SecurityLevel.token := securityLevelToken

• SecurityLevel.Name := "Test Name"

• SecurityLevel.Priority := securityLevel.Priority

• SecurityLevel.Description is skipped

• SecurityLevel.RecognitionGroup[0].RecognitionMethod list := recognitionMethod duplicated cap.MaxRecognitionMethodsPerRecognitionGroup number of times

• SecurityLevel.RecognitionGroup[0].Extension is skipped

• SecurityLevel.Extension is skipped


10. ONVIF client invokes ModifySecurityLevel with parameters

• SecurityLevel.token := securityLevelToken

• SecurityLevel.Name := "Test Name"

• SecurityLevel.Priority := securityLevel.Priority

• SecurityLevel.Description is skipped

• SecurityLevel.RecognitionGroup[0].RecognitionMethod list := recognitionMethod duplicated cap.MaxRecognitionMethodsPerRecognitionGroup + 1 number of times

• SecurityLevel.RecognitionGroup[0].Extension is skipped

• SecurityLevel.Extension is skipped


12. ONVIF Client deletes a security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

• in securityLevelToken - security level token

Test Result:

PASS –

• The DUT passed all assertions.
FAIL –

• The DUT did not send ModifySecurityLevelResponse message.

• The DUT did not send env:Sender/ter:CapabilityViolated/ter:MaxRecognitionMethodsPerRecognitionGroup SOAP 1.2 fault.

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

5.7.17 MODIFY SECURITY LEVEL - DUPLICATE PRIORITY

Test Case ID: AUTH_BEHAVIOR-7-1-17

Specification Coverage: CreateSecurityLevel command (ONVIF Authentication Behavior Service Specification)

Feature Under Test: CreateSecurityLevel

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify modification of security level with duplicated priority.

Pre-Requisite: Authentication Behavior Service is received from the DUT. The DUT shall have enough free storage capacity for one additional Security Level.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

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

   • out cap - Authentication Behavior Service capabilities

4. If cap.MaxSecurityLevels = 1, skip other steps.

5. ONVIF Client find existing or create new security level by following the procedure mentioned in Annex A.5 with the following input and output parameters

   • out securityLevelToken1 - security level token
6. ONVIF Client creates Security Level by following the procedure mentioned in Annex A.17 with the following input and output parameters
   • out securityLevel1 - security level
   • out newSecurityLevel - flag if new security level was created

7. ONVIF client invokes ModifySecurityLevel with parameters
   • SecurityLevel.token := securityLevel2
   • SecurityLevel.Name := "Test Name"
   • SecurityLevel.Priority := securityLevel1.Priority
   • SecurityLevel.Description is skipped
   • SecurityLevel.RecognitionGroup is skipped
   • SecurityLevel.Extension is skipped


9. If newSecurityLevel = true:
   9.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters
       • in securityLevelToken1 - security level token

10. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters
    • in securityLevelToken2 - security level token

Test Result:

PASS –
   • The DUT passed all assertions.

FAIL –
   • The DUT did not send env:Sender/ter:InvalidArgVal/ter:DuplicatePriority SOAP 1.2 fault.
**Note:** If the DUT sends other SOAP 1.2 fault message than specified, log WARNING message, and PASS the test.

### 5.7.18 SET SECURITY LEVEL - CAPABILITY VIOLATED (MAX RECOGNITION GROUPS PER SECURITY LEVEL)

**Test Case ID:** AUTH_BEHAVIOR-7-1-18

**Specification Coverage:** SetSecurityLevel command (ONVIF Authentication Behavior Service Specification)

**Feature Under Test:** SetSecurityLevel

**WSDL Reference:** authenticationbehavior.wsdl

**Test Purpose:** To verify modification of security level with maximum number of recognition groups per security level using SetSecurityLevel command.

**Pre-Requisite:** Authentication Behavior Service is received from the DUT. Token supplying is supported by the DUT as indicated by ClientSuppliedTokenSupported capability. The DUT shall have enough free storage capacity for one additional SecurityLevel.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.2 with the following input and output parameters
   - `out cap` - Authentication Behavior Service capabilities
4. If `cap.MaxRecognitionGroupsPerSecurityLevel` value is more than 50, skip other steps.
5. ONVIF Client creates Security Level by following the procedure mentioned in Annex A.17 with the following input and output parameters
   - `out securityLevelToken` - security level token
   - `out securityLevel` - security level
6. If `cap.MaxRecognitionGroupsPerSecurityLevel` is equal to one, go to step 10.
7. **Set recognitionGroup**: 
   - RecognitionMethod[0].RecognitionType := TODO (clarification)
   - RecognitionMethod[0].Order := 1
   - RecognitionMethod[0].Extension is skipped

8. ONVIF client invokes **SetSecurityLevel** with parameters
   - SecurityLevel.token := securityLevelToken
   - SecurityLevel.Name := "Test Name"
   - SecurityLevel.Description is skipped
   - SecurityLevel.RecognitionGroup list := recognitionGroup duplicated maxRecognitionGroupsPerSecurityLevel number of times
   - SecurityLevel.Extension is skipped

9. The DUT responds with **SetSecurityLevelResponse** message.

10. ONVIF client invokes **SetSecurityLevel** with parameters
    - SecurityLevel.token := securityLevelToken
    - SecurityLevel.Name := "Test Name"
    - SecurityLevel.Description is skipped
    - SecurityLevel.RecognitionGroup list := recognitionGroup duplicated maxRecognitionGroupsPerSecurityLevel + 1 number of times
    - SecurityLevel.Extension is skipped


12. ONVIF Client deletes a security level by following the procedure mentioned in Annex A.22 with the following input and output parameters
    - in securityLevelToken - security level token

**Test Result:**
PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send `SetSecurityLevelResponse` message.

• The DUT did not send `env:Sender/ter:CapabilityViolated/ter:MaxPoliciesPerSecurityLevel` SOAP 1.2 fault.

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

5.7.19 SET SECURITY LEVEL - CAPABILITY VIOLATED (MAX RECOGNITION METHODS PER RECOGNITION GROUP)

Test Case ID: AUTH_BEHAVIOR-7-1-19


Feature Under Test: SetSecurityLevel

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify modification of security level with maximum number recognition methods per recognition group using SetSecurityLevel command.

Pre-Requisite: Authentication Behavior Service is received from the DUT. Token supplying is supported by the DUT as indicated by ClientSuppliedTokenSupported capability. The DUT shall have enough free storage capacity for one additional SecurityLevel.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.2 with the following input and output parameters
4. If `cap.MaxRecognitionMethodsPerRecognitionGroup` value is more than 50, skip other steps.

5. ONVIF Client creates Security Level by following the procedure mentioned in Annex A.17 with the following input and output parameters:
   - `securityLevelToken` - security level token
   - `securityLevel` - security level

6. If `cap.MaxRecognitionMethodsPerRecognitionGroup` is equal to one, go to step 10.

7. Set `recognitionMethod` :=
   - `RecognitionType` := TODO (clarification)
   - `Order` := 1
   - Extension is skipped

8. ONVIF client invokes `SetSecurityLevel` with parameters:
   - `SecurityLevel.token` := `securityLevelToken`
   - `SecurityLevel.Name` := "Test Name"
   - `SecurityLevel.Description` is skipped
   - `SecurityLevel.RecognitionGroup[0].RecognitionMethod list` := `recognitionMethod` duplicated `cap.MaxRecognitionMethodsPerRecognitionGroup` number of times
   - `SecurityLevel.RecognitionGroup[0].Extension` is skipped
   - `SecurityLevel.Extension` is skipped


10. ONVIF client invokes `SetSecurityLevel` with parameters:
    - `SecurityLevel.token` := `securityLevelToken`
    - `SecurityLevel.Name` := "Test Name"
• SecurityLevel.Description is skipped

• SecurityLevel.RecognitionGroup[0].RecognitionMethod list := recognitionMethod duplicated cap.MaxRecognitionMethodsPerRecognitionGroup + 1 number of times

• SecurityLevel.RecognitionGroup[0].Extension is skipped

• SecurityLevel.Extension is skipped


12. ONVIF Client deletes a security level by following the procedure mentioned in Annex A.22 with the following input and output parameters

• in securityLevelToken - security level token

Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send SetSecurityLevelResponse mesage.

• The DUT did not send env:Sender/ter:CapabilityViolated/ter:MaxRecognitionMethodsPerRecognitionGroup SOAP 1.2 fault.

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

5.7.20 SET SECURITY LEVEL - DUPLICATE PRIORITY

Test Case ID: AUTH_BEHAVIOR-7-1-20


Feature Under Test: SetSecurityLevel

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify creation of security level with duplicated priority using SetSecurityLevel command.
**Pre-Requisite:** Authentication Behavior Service is received from the DUT. Token supplying is supported by the DUT as indicated by ClientSuppliedTokenSupported capability. The DUT shall have enough free storage capacity for one additional Security Level.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.2 with the following input and output parameters
   - out cap - Authentication Behavior Service capabilities

4. If $\text{cap}.\text{MaxSecurityLevels} = 1$, skip other steps.

5. ONVIF Client find existing or create new security level by following the procedure mentioned in Annex A.5 with the following input and output parameters
   - out $\text{securityLevelToken}$ - security level token
   - out $\text{securityLevel}$ - security level
   - out $\text{newSecurityLevel}$ - flag if new security level was created

6. ONVIF client invokes **SetSecurityLevel** with parameters
   - SecurityLevel.token := string other than $\text{securityLevelToken}$
   - SecurityLevel.Name := "Test Name"
   - SecurityLevel.Priority := $\text{securityLevel}.\text{Priority}$
   - SecurityLevel.Description is skipped
   - SecurityLevel.RecognitionGroup is skipped
   - SecurityLevel.Extension is skipped


8. If $\text{newSecurityLevel} = \text{true}$:

   8.1. ONVIF Client deletes security level by following the procedure mentioned in Annex A.22 with the following input and output parameters
Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send `env:Sender/ter:InvalidArgVal/ter:DuplicatePriority` SOAP 1.2 fault.

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

5.7.21 SET SECURITY LEVEL - EMPTY TOKEN

Test Case ID: AUTH_BEHAVIOR-7-1-21


Feature Under Test: SetSecurityLevel

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify set of security level with empty token.

Pre-Requisite: Authentication Behavior Service is received from the DUT. Token supplying is supported by the DUT as indicated by ClientSuppliedTokenSupported capability. The DUT shall have enough free storage capacity for one additional SecurityLevel.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF client invokes `SetSecurityLevel` with parameters

   • `SecurityLevel.token := ""`
   
   • `SecurityLevel.Name := "Test Name"`
   
   • `SecurityLevel.Priority := 0`
• SecurityLevel.Description is skipped
• SecurityLevel.RecognitionGroup is skipped
• SecurityLevel.Extension is skipped


Test Result:

PASS –
• The DUT passed all assertions.

FAIL –
• The DUT did not send env:Sender/ter:InvalidArgs SOAP 1.2 fault.

5.7.22 DELETE SECURITY LEVEL - INVALID TOKEN

Test Case ID: AUTH_BEHAVIOR-7-1-22

Specification Coverage: DeleteSecurityLevel command (ONVIF Authentication Behavior Service Specification)

Feature Under Test: DeleteSecurityLevel

WSDL Reference: authenticationbehavior.wsdl

Test Purpose: To verify deleting of security level with invalid token.

Pre-Requisite: Authentication Behavior Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters
   • out securityLevelInfoList - complete list of security levels information

4. Set invalidToken := value not equal to any securityLevelInfoList.token
5. ONVIF Client invokes **DeleteSecurityLevel** with parameters

   - Token := *invalidToken*


**Test Result:**

**PASS** –

- The DUT passed all assertions.

**FAIL** –

- The DUT did not send **env:Sender/ter:InvalidArgVal/ter:NotFound** SOAP 1.2 fault.

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

### 5.7.23 DELETE SECURITY LEVEL - NO TOKEN

**Test Case ID:** AUTH_BEHAVIOR-7-1-23

**Specification Coverage:** DeleteSecurityLevel command (ONVIF Authentication Behavior Service Specification)

**Feature Under Test:** DeleteSecurityLevel

**WSDL Reference:** authenticationbehavior.wsdl

**Test Purpose:** To verify deleting of security level without token.

**Pre-Requisite:** Authentication Behavior Service is received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Sequence:**

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client invokes **DeleteSecurityLevel** with parameters

   - Token := ""  

4. The DUT returns **env:Sender/ter:InvalidArgVal** SOAP 1.2 fault.
Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send `env:Sender/ter:InvalidArgVal` SOAP 1.2 fault.

5.8 Authentication Behavior Events

5.8.1 AUTHENTICATION PROFILE CHANGED EVENT

Test Case ID: AUTH_BEHAVIOR-8-1-1


Feature Under Test: GetEventProperties

WSDL Reference: event.wsdl

Test Purpose: To verify tns1:Configuration/AuthenticationProfile/Changed event format.

Pre-Requisite: Authentication Behavior Service is supported by the DUT. Event Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetEventProperties.
4. The DUT responds with a GetEventPropertiesResponse message with parameters
   • TopicNamespaceLocation list
   • FixedTopicSet
   • TopicSet =: topicSet
- TopicExpressionDialect list
- MessageContentFilterDialect list
- MessageContentSchemaLocation list

5. If topicSet does not contain tns1:Configuration/AuthenticationProfile/Changed topic, FAIL the test, restore the DUT state, and skip other steps.

6. ONVIF Client verifies tns1:Configuration/AuthenticationProfile/Changed topic (authProfileChangedTopic) from topicSet:

6.1. If authProfileChangedTopic.MessageDescription.IsProperty equals to true, FAIL the test, restore the DUT state, and skip other steps.

6.2. If authProfileChangedTopic does not contain MessageDescription.Source.SimpleItemDescription item with Name = "AuthenticationProfileToken", FAIL the test, restore the DUT state, and skip other steps.

6.3. If authProfileChangedTopic.MessageDescription.Source.SimpleItemDescription with Name = "AuthenticationProfileToken" does not have Type = "pt:ReferenceToken", FAIL the test, restore the DUT state, and skip other steps.

Test Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send GetEventPropertiesResponse message.

5.8.2 AUTHENTICATION PROFILE REMOVED EVENT

Test Case ID: AUTH_BEHAVIOR-8-1-2


Feature Under Test: GetEventProperties

WSDL Reference: event.wsdl
Test Purpose: To verify tns1:Configuration/AuthenticationProfile/Removed event format.

Pre-Requisite: Authentication Behavior Service is supported by the DUT. Event Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetEventProperties.
4. The DUT responds with a GetEventPropertiesResponse message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: topicSet
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list
5. If topicSet does not contain tns1:Configuration/AuthenticationProfile/Removed topic, FAIL the test, restore the DUT state, and skip other steps.
6. ONVIF Client verifies tns1:Configuration/AuthenticationProfile/Removed topic (authProfileChangedTopic) from topicSet:
   6.1. If authProfileChangedTopic.MessageDescription.IsProperty equals to true, FAIL the test, restore the DUT state, and skip other steps.
   6.2. If authProfileChangedTopic does not contain MessageDescription.Source.SimpleItemDescription item with Name = "AuthenticationProfileToken", FAIL the test, restore the DUT state, and skip other steps.
   6.3. If authProfileChangedTopic.MessageDescription.Source.SimpleItemDescription with Name = "AuthenticationProfileToken" does not have Type = "pt:ReferenceToken", FAIL the test, restore the DUT state, and skip other steps.

Test Result:
PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send `GetEventPropertiesResponse` message.

5.8.3 AUTHENTICATION PROFILE CHANGED EVENT

Test Case ID: AUTH_BEHAVIOR-8-1-3


Feature Under Test: GetEventProperties

WSDL Reference: event.wsdl

Test Purpose: To verify tns1:Configuration/SecurityLevel/Changed event format.

Pre-Requisite: Authentication Behavior Service is supported by the DUT. Event Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes `GetEventProperties`.
4. The DUT responds with a `GetEventPropertiesResponse` message with parameters
   • TopicNamespaceLocation list
   • FixedTopicSet
   • TopicSet =: `topicSet`
   • TopicExpressionDialect list
   • MessageContentFilterDialect list
   • MessageContentSchemaLocation list
5. If topicSet does not contain tns1:Configuration/SecurityLevel/Changed topic, FAIL the test, restore the DUT state, and skip other steps.

6. ONVIF Client verifies tns1:Configuration/SecurityLevel/Changed topic (authProfileChangedTopic) from topicSet:

   6.1. If authProfileChangedTopic.MessageDescription.IsProperty equals to true, FAIL the test, restore the DUT state, and skip other steps.

   6.2. If authProfileChangedTopic does not contain MessageDescription.Source.SimpleItemDescription item with Name = "SecurityLevelToken", FAIL the test, restore the DUT state, and skip other steps.

   6.3. If authProfileChangedTopic.MessageDescription.Source.SimpleItemDescription with Name = "SecurityLevelToken" does not have Type = "pt:ReferenceToken", FAIL the test, restore the DUT state, and skip other steps.

Test Result:

PASS –

- The DUT passed all assertions.

FAIL –

- The DUT did not send GetEventPropertiesResponse message.

5.8.4 AUTHENTICATION PROFILE REMOVED EVENT

Test Case ID: AUTH_BEHAVIOR-8-1-4


Feature Under Test: GetEventProperties

WSDL Reference: event.wsdl

Test Purpose: To verify tns1:Configuration/SecurityLevel/Removed event format.

Pre-Requisite: Authentication Behavior Service is supported by the DUT. Event Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:
1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client invokes **GetEventProperties**.

4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet \( =: topicSet \)
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list

5. If \( topicSet \) does not contain \( \text{tns1:SecurityLevel/AuthenticationProfile/Removed} \) topic, FAIL the test, restore the DUT state, and skip other steps.

6. ONVIF Client verifies \( \text{tns1:Configuration/SecurityLevel/Removed topic} \) (\( authProfileChangedTopic \)) from topicSet:
   - If \( authProfileChangedTopic.MessageDescription.IsProperty \) equals to true, FAIL the test, restore the DUT state, and skip other steps.
   - If \( authProfileChangedTopic \) does not contain \( \text{MessageDescription.Source.SimpleItemDescription} \) item with Name = "SecurityLevelToken", FAIL the test, restore the DUT state, and skip other steps.
   - If \( authProfileChangedTopic.MessageDescription.Source.SimpleItemDescription \) with Name = "SecurityLevelToken" does not have Type = "pt:ReferenceToken", FAIL the test, restore the DUT state, and skip other steps.

**Test Result:**

**PASS** –
- The DUT passed all assertions.

**FAIL** –
- The DUT did not send **GetEventPropertiesResponse** message.
Annex A Helper Procedures and Additional Notes

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

A.1 Get Authentication Profiles Information List

Name: HelperGetAuthenticationProfileInfoList

Procedure Purpose: Helper procedure to get complete authentication profiles information list.

Pre-requisite: Authentication Behavior Service is received from the DUT.

Input: None.

Returns: The complete list of authentication profiles information (authProfileInfoCompleteList).

Procedure:

1. ONVIF client invokes GetAuthenticationProfileInfoList with parameters
   - Limit skipped
   - StartReference skipped

2. The DUT responds with GetAuthenticationProfileInfoListResponse message with parameters
   - NextStartReference =: nextStartReference
   - AuthenticationProfileInfo list =: authProfileInfoCompleteList

3. Until nextStartReference is not null, repeat the following steps:

   3.1. ONVIF client invokes GetAuthenticationProfileInfoList with parameters
       - Limit skipped
       - StartReference := nextStartReference

   3.2. The DUT responds with GetAuthenticationProfileInfoListResponse message with parameters
       - NextStartReference =: nextStartReference
       - AuthenticationProfileInfo list =: authProfileInfoListPart
3.3. Set \[ \text{authProfileInfoCompleteList} := \text{authProfileInfoCompleteList} + \text{authProfileInfoListPart}. \]

Procedure Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send \text{GetAuthenticationProfileInfoListResponse} message.

A.2 Get Service Capabilities

Name: HelperGetServiceCapabilities

Procedure Purpose: Helper procedure to get service capabilities.

Pre-requisite: Authentication Behavior Service is received from the DUT.

Input: None.

Returns: The service capabilities (\( \text{cap} \)).

Procedure:

1. ONVIF client invokes \text{GetServiceCapabilities}.

2. The DUT responds with a \text{GetServiceCapabilitiesResponse} message with parameters

   • Capabilities =: \( \text{cap} \)

Procedure Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send \text{GetServiceCapabilitiesResponse} message.

A.3 Get Authentication Profiles List

Name: HelperGetAuthenticationProfileList
**Procedure Purpose:** Helper procedure to get complete authentication profiles list with.

**Pre-requisite:** Authentication Behavior Service is received from the DUT.

**Input:** None.

**Returns:** The complete list of authentication profiles (`authProfileCompleteList`).

**Procedure:**

1. ONVIF client invokes `GetAuthenticationProfileList` with parameters
   - Limit skipped
   - StartReference skipped

2. The DUT responds with `GetAuthenticationProfileListResponse` message with parameters
   - `NextStartReference` := `nextStartReference`
   - `AuthenticationProfile List` := `authProfileCompleteList`

3. Until `nextStartReference` is not null, repeat the following steps:

   3.1. ONVIF client invokes `GetAuthenticationProfileList` with parameters
       - Limit skipped
       - `StartReference` := `nextStartReference`

   3.2. The DUT responds with `GetAuthenticationProfileListResponse` message with parameters
       - `NextStartReference` := `nextStartReference`
       - `Schedule list` := `authProfilesListPart`

   3.3. Set `authProfileCompleteList` := `authProfileCompleteList` + `authProfilesListPart`

**Procedure Result:**

**PASS** –
- The DUT passed all assertions.

**FAIL** –
- The DUT did not send `GetAuthenticationProfileListResponse` message.
A.4 Create Number of Authentication Profiles

Name: HelperCreateAuthenticationProfiles

Procedure Purpose: Helper procedure to create number of authentication profiles required for test cases.

Pre-requisite: Authentication Behavior Service is received from the DUT.

Input: None.

Returns: The complete list of authentication profiles information (authProfileInfoCompleteList). List of created authentication profiles tokens (createdAuthProfileTokensList). The service capabilities (cap). Created security level token (securityLevelToken).

Procedure:

1. ONVIF Client retrieves a complete list of authentication profile info by following the procedure mentioned in Annex A.1 with the following input and output parameters
   - out authProfileInfoInitialList - complete list of authentication profiles information

2. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.2 with the following input and output parameters
   - out cap - Authentication Behavior Service capabilities

3. Set requiredNumberOfAuthProfile := min {50; cap.MaxLimit; cap.MaxAuthenticationProfiles}.

4. Set authProfileInfoCompleteList := authProfileInfoInitialList.

5. If requiredNumberOfAuthProfile <= number of AuthenticationProfileInfo items in authProfileInfoInitialList, skip other steps of the procedure.

6. Set numberOfAuthProfilesToBeCreated := requiredNumberOfAuthProfile - number of AuthenticationProfileInfo items in authProfileInfoInitialList.

7. ONVIF Client find existing or create new security level by following the procedure mentioned in Annex A.5 with the following input and output parameters
   - out securityLevelToken - security level token
   - out newSecurityLevel - flag if new security level was created

8. ONVIF client invokes CreateAuthenticationProfile with parameters
   - AuthenticationProfile.token := ""
• AuthenticationProfile.Name := "Test Name"

• AuthenticationProfile.Description is skipped

• AuthenticationProfile.AuthenticationPolicy is skipped

• AuthenticationProfile.DefaultSecurityLevelToken := securityLevelToken

• AuthenticationProfile.Extension is skipped

9. The DUT responds with CreateAuthenticationProfileResponse message with parameters

   • Token = authProfileToken

10. Set authProfileInfoCompleteList := authProfileInfoInitialList + new AuthenticationProfileInfo
    (with token := authProfileToken; Name := "Test Name"; DefaultSecurityLevelToken := securityLevelToken).

11. Set createdAuthProfileTokensList := createdAuthProfileTokensList + authProfileToken.

12. Set numberOfAuthProfilesToBeCreated := numberOfAuthProfilesToBeCreated - 1

13. If numberOfAuthProfilesToBeCreated > 0, go to step 8

14. If newSecurityLevel = false:


Procedure Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send CreateAuthenticationProfileResponse message.

A.5 Find or Create Security Level

Name: HelperFindOrCreateSecurityLevel

Procedure Purpose: Helper procedure to find existing or create new security level.

Pre-requisite: Authentication Behavior Service is received from the DUT.

Input: None.
Returns: Security level token (securityLevelToken). Flag if new security level was created (newSecurityLevel). Security level (securityLevel).

Procedure:

1. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters
   • out securityLevelInfoList - complete list of security levels information
2. If securityLevelInfoList contains at least one SecurityLevelInfo:
   2.1. Set securityLevelToken := securityLevelInfoList[0].token.
   2.2. Set securityLevel := securityLevelInfoList[0].
   2.3. Set newSecurityLevel := false.
   2.4. Skip other steps of the procedure.
4. Set securityLevel :=
   • token := ""
   • Name := "Test Name"
   • Priority := 0
   • Description is skipped
   • RecognitionGroup is skipped
   • Extension is skipped
5. ONVIF client invokes CreateSecurityLevel with parameters
   • SecurityLevel := securityLevel
6. The DUT responds with CreateSecurityLevelResponse message with parameters
   • Token := securityLevelToken

Procedure Result:

PASS –
   • The DUT passed all assertions.
FAIL –

- The DUT did not send CreateSecurityLevelResponse message.

A.6 Get Security Levels Information List

Name: HelperGetSecurityLevelInfoList

Procedure Purpose: Helper procedure to get complete security levels information list.

Pre-requisite: Authentication Behavior Service is received from the DUT.

Input: None.

Returns: The complete list of security levels information (securityLevelCompleteList).

Procedure:

1. ONVIF client invokes GetSecurityLevelInfoList with parameters
   - Limit skipped
   - StartReference skipped

2. The DUT responds with GetSecurityLevelInfoListResponse message with parameters
   - NextStartReference =: nextStartReference
   - SecurityLevelInfo list =: securityLevelCompleteList

3. Until nextStartReference is not null, repeat the following steps:
   3.1. ONVIF client invokes GetSecurityLevelInfoList with parameters
       - Limit skipped
       - StartReference := nextStartReference
   3.2. The DUT responds with GetSecurityLevelInfoListResponse message with parameters
       - NextStartReference =: nextStartReference
       - SecurityLevelInfo list =: securityLevelCompleteListPart


Procedure Result:
PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send `GetSecurityLevelInfoListResponse` message.

A.7 Compare Authentication Profile List and Authentication Profile Info List

Name: HelperCompareAuthProfilesList

Procedure Purpose: Helper procedure to compare Authentication Profile List and Authentication Profile Info List.

Pre-requisite: Authentication Behavior Service is received from the DUT.

Input: The list of authentication profiles information (`authProfileInfoList`). The list of authentication profiles (`authProfilesList`).

Returns: None.

Procedure:

1. If `authProfilesList` does not contain all tokens from `authProfileInfoList`, FAIL the test, restore the DUT state, and skip other steps.

2. If `authProfilesList` contains tokens other than tokens from `authProfileInfoList`, FAIL the test, restore the DUT state, and skip other steps.

3. For each AuthenticationProfileInfo.token `token` from `authProfileInfoList` repeat the following steps:

   3.1. If `authProfilesList[token = token]` item does not have equal field values to `authProfileInfoList[token = token]` item, FAIL the test, restore the DUT state, and skip other steps.

Procedure Result:

PASS –

• The DUT passed all assertions.

FAIL –

• None.
Note: The following fields are compared at step 3.1:

- AuthenticationProfile/AuthenticationProfileInfo:
  - token
  - Name
  - Description

A.8 Create Pull Point Subscription

Name: HelperCreatePullPointSubscription

Procedure Purpose: Helper procedure to create PullPoint Subscription with specified Topic.

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

Input: Notification Topic (topic).

Returns: Subscription reference (s), current time for the DUT (ct), subscription termination time (tt).

Procedure:

1. ONVIF Client invokes CreatePullPointSubscription request with parameters
   - Filter.TopicExpression := topic
   - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"

2. The DUT responds with CreatePullPointSubscriptionResponse message with parameters
   - SubscriptionReference := s
   - CurrentTime := ct
   - TerminationTime := tt

Procedure Result:

PASS –
- DUT passes all assertions.

FAIL –
- DUT did not send CreatePullPointSubscriptionResponse message.
A.9  Delete Subscription

**Name:** HelperDeleteSubscription

**Procedure Purpose:** Helper procedure to delete subscription.

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

**Input:** Subscription reference (s)

**Returns:** None

**Procedure:**

1. ONVIF Client sends an **Unsubscribe** to the subscription endpoint s.
2. The DUT responds with **UnsubscribeResponse** message.

**Procedure Result:**

PASS –

- DUT passes all assertions.

FAIL –

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

A.10  Retrieve Authentication Profile Changed Event by PullPoint

**Name:** HelperPullAuthProfileChanged

**Procedure Purpose:** Helper procedure to retrieve and check tns1:Configuration/AuthenticationProfile/Changed event with PullMessages.

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

**Input:** Subscription reference (s), current time for the DUT (ct), Subscription termination time (tt) and Authentication Profile token (authProfileToken).

**Returns:** None

**Procedure:**

1. Until **operationDelay** timeout expires, repeat the following steps:

   1.1. ONVIF Client waits for time \( t := \min((tt-ct)/2, 1 \text{ second}) \)
1.2. ONVIF Client invokes **PullMessages** to the subscription endpoint with parameters

- Timeout := PT60S
- MessageLimit := 1

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

- CurrentTime =: \( ct \)
- TerminationTime =: \( tt \)
- NotificationMessage list =: \( notificationMessageList \)

1.4. If \( notificationMessageList \) is not empty and the AuthenticationProfileToken source simple item in \( notificationMessageList \) is equal to \( authProfileToken \), skip other steps and finish the procedure.

1.5. If \( timeout1 \) timeout expires for step 1 without Notification with Token source simple item equal to \( authProfileToken \), FAIL the test, restore the DUT state, and skip other steps.

**Procedure Result:**

**PASS** –

- DUT passes all assertions.

**FAIL** –

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

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

### A.11 Get Authentication Profile

**Name:** HelperGetAuthenticationProfile

**Procedure Purpose:** Helper procedure to get authentication profile.

**Pre-requisite:** Authentication Behavior Service is received from the DUT.

**Input:** Authentication Profile Token (\( authProfileToken \)).

**Returns:** Authentication Profile List (\( authProfileList \)).

**Procedure:**

1. ONVIF client invokes **GetAuthenticationProfiles** with parameters
2. The DUT responds with \texttt{GetAuthenticationProfilesResponse} message with parameters

\begin{itemize}
\item AuthenticationProfile list =: \texttt{authProfileList}
\end{itemize}

\textbf{Procedure Result:}

\begin{itemize}
\item \textbf{PASS} –
\begin{itemize}
\item The DUT passed all assertions.
\end{itemize}
\item \textbf{FAIL} –
\begin{itemize}
\item The DUT did not send \texttt{GetAuthenticationProfilesResponse} message.
\end{itemize}
\end{itemize}

\section*{A.12 Get Authentication Profile Info}

\textbf{Name:} HelperGetAuthenticationProfileInfo

\textbf{Procedure Purpose:} Helper procedure to get schedule info.

\textbf{Pre-requisite:} Authentication Behavior Service is received from the DUT.

\textbf{Input:} Authentication Profile Token (\texttt{authProfileToken}).

\textbf{Returns:} Authentication Profile Info List (\texttt{authProfileInfoList}).

\textbf{Procedure:}

\begin{enumerate}
\item ONVIF client invokes \texttt{GetAuthenticationProfileInfo} with parameters
\begin{itemize}
\item Token[0] := \texttt{authProfileToken}
\end{itemize}
\item The DUT responds with \texttt{GetAuthenticationProfileInfoResponse} message with parameters
\begin{itemize}
\item AuthenticationProfileInfo list =: \texttt{authProfileInfoList}
\end{itemize}
\end{enumerate}

\textbf{Procedure Result:}

\begin{itemize}
\item \textbf{PASS} –
\begin{itemize}
\item The DUT passed all assertions.
\end{itemize}
\item \textbf{FAIL} –
\begin{itemize}
\item The DUT did not send \texttt{GetAuthenticationProfileInfoResponse} message.
\end{itemize}
\end{itemize}
A.13 Delete Authentication Profile

**Name:** HelperDeleteAuthenticationProfile

**Procedure Purpose:** Helper procedure to delete authentication profile.

**Pre-requisite:** Authentication Behavior Service is received from the DUT.

**Input:** Authentication Profile Token (**authProfileToken**).

**Returns:** None.

**Procedure:**

1. ONVIF client invokes `DeleteAuthenticationProfile` with parameters
   
   • Token =: **authProfileToken**

2. The DUT responds with empty `DeleteAuthenticationProfileResponse` message

**Procedure Result:**

**PASS** –

• The DUT passed all assertions.

**FAIL** –

• The DUT did not send `DeleteAuthenticationProfileResponse` message.

A.14 Find or Create Schedule

**Name:** HelperFindOrCreateSchedule

**Procedure Purpose:** Helper procedure to find existing or create new schedule.

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

**Input:** None.

**Returns:** Schedule token (**scheduleToken**). Flag if new Schedule was created (**newSchedule**).

**Procedure:**

1. ONVIF Client retrieves a complete list of schedules info by following the procedure mentioned in Annex A.18 with the following input and output parameters

   • `out scheduleInfoList` - complete list of schedules information
2. If `scheduleInfoList` contains at least one `ScheduleInfo`:
   2.1. Set `scheduleToken := scheduleInfoList[0].token.`
   2.2. Set `newSchedule := false`.
   2.3. Skip other steps of the procedure.
4. ONVIF Client generates appropriate iCalendar value for the AuthenticationProfile.Standard field by following the procedure mentioned in Annex A.15 with the following input and output parameters
   - `out scheduleiCalendarValue` - iCalendarValue for the AuthenticationProfile.Standard field
5. ONVIF client invokes `CreateSchedule` with parameters
   - `Schedule.token := ""`
   - `Schedule.Description` is skipped
   - `Schedule.Name := "Test Name"`
   - `Schedule.Standard := scheduleiCalendarValue`
   - `Schedule.SpecialDays` is skipped
6. The DUT responds with `CreateScheduleResponse` message with parameters
   - `Token := scheduleToken`

**Procedure Result:**

**PASS** –
- The DUT passed all assertions.

**FAIL** –
- The DUT did not send `CreateScheduleResponse` message.

**A.15 Generate iCalendar Value for Schedule**

**Name:** HelperScheduleiCalendarGeneration

**Procedure Purpose:** Helper procedure to generate iCalendar value for Schedule.Standard field.

**Pre-requisite:** Schedule Service is received from the DUT.
Input: None.

Returns: iCalendar value for Standard field (scheduleiCalendarValue) that is compliant to [RFC 2445].

Procedure:

1. Set uid := new Globally Unique Identifier value.

2. Set scheduleiCalendarValue := "BEGIN:VCALENDAR
   BEGIN:VEVENT
   SUMMARY:Access on weekdays from 9 AM to 6 PM for employees
   DTSTART:1970<current month><current day>T090000
   DTEND:1970<current month><current day>T180000
   RRULE:FREQ=WEEKLY;BYDAY=MO,TU,WE,TH,FR
   UID:uid
   END:VEVENT
   END:VCALENDAR"

A.16 Create Authentication Profile

Name: HelperCreateAuthProfile

Procedure Purpose: Helper procedure to create authentication profile.

Pre-requisite: Authentication Behavior Service is received from the DUT.

Input: None.

Returns: Authentication profile token (authProfileToken). Authentication profile (authProfile). Flag if new Schedule was created (newSchedule). Flag if new security level was created (newSecurityLevel).

Procedure:

1. ONVIF Client find existing or create new security level by following the procedure mentioned in Annex A.5 with the following input and output parameters
   - out securityLevelToken - security level token
ONVIF Authentication Behavior Device Test Specification Version 18.12

1. out newSecurityLevel - flag if new security level was created

2. ONVIF Client find existing or create new schedule by following the procedure mentioned in Annex A.14 with the following input and output parameters
   - out scheduleToken - schedule level token
   - out newSchedule - flag if new schedule was created

3. Set authProfile :=
   - AuthenticationProfile.token := ""
   - AuthenticationProfile.Description := "Test Description"
   - AuthenticationProfile.Name := "Test Name"
   - AuthenticationProfile.DefaultSecurityLevelToken := securityLevelToken
   - AuthenticationProfile.AuthenticationPolicy[0].ScheduleToken := scheduleToken
   - AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].ActiveRegularSchedule := true
   - AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].ActiveSpecialDaySchedule := true
   - AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].AuthenticationMode[0] := TODO (clarification)
   - AuthenticationProfile.AuthenticationPolicy[0].SecurityLevelConstraint[0].SecurityLevelToken := securityLevelToken

4. ONVIF client invokes CreateAuthenticationProfile with parameters
   - AuthenticationProfile := authProfile

5. The DUT responds with CreateAuthenticationProfileResponse message with parameters
   - Token := authProfileToken

Procedure Result:

PASS –
   - The DUT passed all assertions.

FAIL –
ONVIF Authentication Behavior Device Test Specification Version 18.12

- The DUT did not send `CreateAuthenticationProfileResponse` message.

### A.17 Create Security Level

**Name:** HelperCreateSecurityLevel

**Procedure Purpose:** Helper procedure to create security level.

**Pre-requisite:** Authentication Behavior Service is received from the DUT.

**Input:** None.

**Returns:** Security level token (securityLevelToken). Security level (securityLevel).

**Procedure:**

1. ONVIF Client retrieves a complete list of security level info by following the procedure mentioned in Annex A.6 with the following input and output parameters

   - out securityLivellInfoList - complete list of security levels information

2. Set `securityLevel` :=

   - SecurityLevel.token := ""
   - SecurityLevel.Name := "Test Name"
   - SecurityLevel.Priority := other then specified for SecurityLivellInfo items in securityLivellInfoList
   - SecurityLevel.Description := "Test Description"
   - SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].RecognitionType = TODO (calrification)
   - SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].Order = 1
   - SecurityLevel.RecognitionGroup[0].RecognitionMethod[0].Extension is skipped
   - SecurityLevel.RecognitionGroup[0].Extension is skipped
   - SecurityLevel.Extension is skipped

3. ONVIF client invokes `CreateSecurityLevel` with parameters

   - SecurityLevel := `securityLevel`

4. The DUT responds with `CreateSecurityLevelResponse` message with parameters
• Token =: securityLevelToken

Procedure Result:

PASS –
  • The DUT passed all assertions.

FAIL –
  • The DUT did not send CreateSecurityLevelResponse message.

A.18 Get Schedules Information List

Name: HelperGetScheduleInfoList

Procedure Purpose: Helper procedure to get complete schedules information list.

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

Input: None.

Returns: The complete list of schedules information (scheduleInfoCompleteList).

Procedure:

1. ONVIF client invokes GetScheduleInfoList with parameters
   • Limit is skipped
   • StartReference is skipped

2. The DUT responds with GetScheduleInfoListResponse message with parameters
   • NextStartReference =: nextStartReference
   • ScheduleInfo list =: scheduleInfoCompleteList

3. Until nextStartReference is not null, repeat the following steps:
   3.1. ONVIF client invokes GetScheduleInfoList with parameters
       • Limit skipped
       • StartReference := nextStartReference
   3.2. The DUT responds with GetScheduleInfoListResponse message with parameters
       • NextStartReference =: nextStartReference
3.3. Set $scheduleInfoCompleteList := scheduleInfoCompleteList + scheduleInfoListPart$

Procedure Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send $GetScheduleInfoListResponse$ message.

A.19 Get Schedule Service Capabilities

Name: HelperGetScheduleServiceCapabilities

Procedure Purpose: Helper procedure to get service capabilities.

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

Input: None.

Returns: The service capabilities ($cap$).

Procedure:

1. ONVIF client invokes $GetServiceCapabilities$.

2. The DUT responds with a $GetServiceCapabilitiesResponse$ message with parameters

   • Capabilities $= cap$

Procedure Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send $GetServiceCapabilitiesResponse$ message.

A.20 Create Schedule

Name: HelperCreateSchedule
**Procedure Purpose:** Helper procedure to create schedule.

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

**Input:** None.

**Returns:** Schedule token (scheduleToken).

**Procedure:**

1. ONVIF Client generates appropriate iCalendar value for the AuthenticationProfile.Standard field by following the procedure mentioned in Annex A.15 with the following input and output parameters
   - out scheduleiCalendarValue - iCalendarValue for the AuthenticationProfile.Standard field
2. ONVIF client invokes `CreateSchedule` with parameters
   - Schedule.token := ""
   - Schedule.Description is skipped
   - Schedule.Name := "Test Name"
   - Schedule.Standard := `scheduleiCalendarValue`
   - Schedule.SpecialDays is skipped
3. The DUT responds with `CreateScheduleResponse` message with parameters
   - Token =: scheduleToken

**Procedure Result:**

**PASS** –

- The DUT passed all assertions.

**FAIL** –

- The DUT did not send `CreateScheduleResponse` message.

A.21 Retrieve Authentication Profile Removed Event by PullPoint

**Name:** HelperPullAuthProfileRemoved
Procedure Purpose: Helper procedure to retrieve and check tns1:Configuration/AuthenticationProfile/Removed event with PullMessages.

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

Input: Subscription reference (s), current time for the DUT (ct), Subscription termination time (tt) and Authentication Profile token (authProfileToken).

Returns: None

Procedure:

1. Until operationDelay timeout expires, repeat the following steps:

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

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

       - Timeout := PT60S
       - MessageLimit := 1

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

       - CurrentTime =: ct
       - TerminationTime =: tt
       - NotificationMessage list =: notificationMessageList

   1.4. If notificationMessageList is not empty and the AuthenticationProfileToken source simple item in notificationMessageList is equal to authProfileToken, skip other steps and finish the procedure.

   1.5. If timeout1 timeout expires for step 1 without Notification with Token source simple item equal to authProfileToken, FAIL the test, restore the DUT state, and skip other steps.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Note: operationDelay will be taken from Operation Delay field of ONVIF Device Test Tool.
A.22 Delete Security Level

Name: HelperDeleteSecurityLevel

Procedure Purpose: Helper procedure to delete security level.

Pre-requisite: Authentication Behavior Service is received from the DUT.

Input: Security Level Token (securityLevelToken).

Returns: None.

Procedure:

1. ONVIF client invokes DeleteSecurityLevel with parameters
   • Token =: securityLevelToken

2. The DUT responds with empty DeleteSecurityLevelResponse message

Procedure Result:

PASS –
   • The DUT passed all assertions.

FAIL –
   • The DUT did not send DeleteSecurityLevelResponse message.

A.23 Delete Schedule

Name: HelperDeleteSchedule

Procedure Purpose: Helper procedure to delete schedule.

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

Input: Schedule Token (scheduleToken).

Returns: None.

Procedure:

1. ONVIF client invokes DeleteSchedule with parameters
   • Token =: scheduleToken
2. The DUT responds with empty `DeleteScheduleResponse` message

**Procedure Result:**

**PASS** –

- The DUT passed all assertions.

**FAIL** –

- The DUT did not send `DeleteScheduleResponse` message.

### A.24 Create Number of Security Levels

**Name:** HelperCreateSecurityLevels

**Procedure Purpose:** Helper procedure to create number of security levels required for test cases.

**Pre-requisite:** Authentication Behavior Service is received from the DUT.

**Input:** None.

**Returns:** The complete list of security levels information (`securityLevelsInfoCompleteList`). List of created security levels tokens (`createdSecurityLevelsTokensList`). The service capabilities (`cap`).

**Procedure:**

1. ONVIF Client retrieves a complete list of security levels info by following the procedure mentioned in Annex A.6 with the following input and output parameters

   - `out securityLevelInfoInitialList` - complete list of security level information

2. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.2 with the following input and output parameters

   - `out cap` - Authentication Behavior Service capabilities


5. If `requiredNumberOfSecurityLevel <=` number of SecurityLevelInfo items in `securityLevelInfoInitialList`, skip other steps of the procedure.

7. ONVIF client invokes **CreateSecurityLevel** with parameters

- SecurityLevel.token := ""
- SecurityLevel.Name := "Test Name"
- SecurityLevel.Priority := other then specified for SecurityLevelInfo items in securityLevelInfoCompleteList
- SecurityLevel.Description is skipped
- SecurityLevel.RecognitionGroup is skipped
- SecurityLevel.Extension is skipped

8. The DUT responds with **CreateSecurityLevelResponse** message with parameters

- Token =: securityLevelToken


11. Set numberOfSecurityLevelToBeCreated := numberOfSecurityLevelToBeCreated - 1

12. If numberOfSecurityLevelToBeCreated > 0, go to step 7

**Procedure Result:**

**PASS** –

- The DUT passed all assertions.

**FAIL** –

- The DUT did not send **CreateSecurityLevelResponse** message.

### A.25 Get Security Level List

**Name:** HelperGetSecurityLevelList

**Procedure Purpose:** Helper procedure to get complete security levels list with.

**Pre-requisite:** Authentication Behavior Service is received from the DUT.
Input: None.

Returns: The complete list of security levels (securityLevelCompleteList).

Procedure:

1. ONVIF client invokes GetSecurityLevelList with parameters
   - Limit skipped
   - StartReference skipped

2. The DUT responds with GetSecurityLevelListResponse message with parameters
   - NextStartReference =: nextStartReference
   - SecurityLevel list =: securityLevelCompleteList

3. Until nextStartReference is not null, repeat the following steps:
   - 3.1. ONVIF client invokes GetSecurityLevelList with parameters
       - Limit skipped
       - StartReference := nextStartReference
   - 3.2. The DUT responds with GetSecurityLevelListResponse message with parameters
       - NextStartReference =: nextStartReference
       - Schedule list =: securityLevelsListPart
   - 3.3. Set securityLevelCompleteList := securityLevelCompleteList + securityLevelsListPart

Procedure Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send GetSecurityLevelListResponse message.

A.26 Compare Security Level List and Security Level Info List

Name: HelperCompareSecurityLevelsList

Procedure Purpose: Helper procedure to compare Security Level List and Security Level Info List.
Pre-requisite: Authentication Behavior Service is received from the DUT.

Input: The list of security levels information (securityLevelInfoList). The list of security levels (securityLevelsList).

Returns: None.

Procedure:

1. If securityLevelsList does not contain all tokens from securityLevelInfoList, FAIL the test, restore the DUT state, and skip other steps.

2. If securityLevelsList contains tokens other than tokens from securityLevelInfoList, FAIL the test, restore the DUT state, and skip other steps.

3. For each SecurityLevelInfo.token token from securityLevelInfoList repeat the following steps:

   3.1. If securityLevelsList[token = token] item does not have equal field values to securityLevelInfoList[token = token] item, FAIL the test, restore the DUT state, and skip other steps.

Procedure Result:

PASS –

• The DUT passed all assertions.

FAIL –

• None.

Note: The following fields are compared at step 3.1:

• SecurityLevel/SecurityLevelInfo:
  
  • token
  
  • Name
  
  • Priority
  
  • Description

A.27 Retrieve Security Level Changed Event by PullPoint

Name: HelperPullSecurityLevelChanged
**Procedure Purpose:** Helper procedure to retrieve and check tns1:Configuration/SecurityLevel/Changed event with PullMessages.

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

**Input:** Subscription reference (s), current time for the DUT (ct), Subscription termination time (tt) and Security Level token (securityLevelToken).

**Returns:** None

**Procedure:**

1. Until operationDelay timeout expires, repeat the following steps:
   1.1. ONVIF Client waits for time $t := \min((tt-ct)/2, 1 \text{ second})$.
   1.2. ONVIF Client invokes **PullMessages** to the subscription endpoint s with parameters
      - Timeout := PT60S
      - MessageLimit := 1
   1.3. The DUT responds with **PullMessagesResponse** message with parameters
      - CurrentTime := ct
      - TerminationTime := tt
      - NotificationMessage list =: notificationMessageList
   1.4. If notificationMessageList is not empty and the SecurityLevelToken source simple item in notificationMessageList is equal to securityLevelToken, skip other steps and finish the procedure.
   1.5. If timeout1 timeout expires for step 1 without Notification with Token source simple item equal to securityLevelToken, FAIL the test, restore the DUT state, and skip other steps.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

**Note:** operationDelay will be taken from Operation Delay field of ONVIF Device Test Tool.
A.28 Retrieve Security Level Removed Event by PullPoint

**Name:** HelperPullSecurityLevelRemoved

**Procedure Purpose:** Helper procedure to retrieve and check tns1:Configuration/SecurityLevel/Removed event with PullMessages.

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

**Input:** Subscription reference \( (s) \), current time for the DUT \((ct)\), Subscription termination time \((tt)\) and Security Level token \((securityLevelToken)\).

**Returns:** None

**Procedure:**

1. Until \( operationDelay \) timeout expires, repeat the following steps:
   1.1. ONVIF Client waits for time \( t := \min\{(tt-ct)/2, 1 \text{ second}\} \).
   1.2. ONVIF Client invokes \texttt{PullMessages} to the subscription endpoint \( s \) with parameters
      - \textbf{Timeout} := PT60S
      - \textbf{MessageLimit} := 1
   1.3. The DUT responds with \texttt{PullMessagesResponse} message with parameters
      - \textbf{CurrentTime} := \( ct \)
      - \textbf{TerminationTime} := \( tt \)
      - \textbf{NotificationMessage list} := \texttt{notificationMessageList}
   1.4. If \texttt{notificationMessageList} is not empty and the SecurityLevelToken source simple item in \texttt{notificationMessageList} is equal to \( securityLevelToken \), skip other steps and finish the procedure.
   1.5. If \( timeout1 \) timeout expires for step 1 without Notification with Token source simple item equal to \( securityLevelToken \), FAIL the test, restore the DUT state, and skip other steps.

**Procedure Result:**

**PASS** –

- DUT passes all assertions.

**FAIL** –
A.29 Get Security Level Info

Name: HelperGetSecurityLevelInfo

Procedure Purpose: Helper procedure to get security level info.

Pre-requisite: Authentication Behavior Service is received from the DUT.

Input: Security Level Token (securityLevelToken).

Returns: Security Level Info List (securityLevelInfoList).

Procedure:

1. ONVIF client invokes GetSecurityLevelInfo with parameters
   - Token[0] := securityLevelToken

2. The DUT responds with GetSecurityLevelInfoResponse message with parameters
   - SecurityLevelInfo list =: securityLevelInfoList

Procedure Result:

PASS –
   - The DUT passed all assertions.

FAIL –
   - The DUT did not send GetSecurityLevelInfoResponse message.

A.30 Get Security Level

Name: HelperGetSecurityLevel

Procedure Purpose: Helper procedure to get security level.

Pre-requisite: Authentication Behavior Service is received from the DUT.

Input: Security Level Token (securityLevelToken).

Procedure:

1. ONVIF client invokes `GetSecurityLevels` with parameters
   ```
   • Token[0] := securityLevelToken
   ```

2. The DUT responds with `GetSecurityLevelsResponse` message with parameters
   ```
   • SecurityLevel list := securityLevelList
   ```

Procedure Result:

PASS –

• The DUT passed all assertions.

FAIL –

• The DUT did not send `GetSecurityLevelsResponse` message.