

ONVIF[®]

Access Rules Device Test Specification

Version 18.12

December 2018

© 2018 ONVIF, Inc. All rights reserved.

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

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

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

REVISION HISTORY

Vers.	Date	Description
15.06		First issue of Access Rules Test Specification
17.01	Jan, 2017	ACCESS_RULES-3-1-11 CREATE ACCESS PROFILE - NOT EMPTY ACCESS PROFILE TOKEN was updated according ticket #1124.
17.12	Jul 14, 2017	<p>ACCESS_RULES-3-1-5 CREATE ACCESS PROFILE was updated according ticket # 1365.</p> <p>The following annexes were added according to #1365:</p> <p>A.12 Get service capabilities (Schedule)</p> <p>A.13 Generate UID value for iCalendar</p> <p>A.14 Generate iCalendar value for Schedule</p> <p>A.15 Create special day group</p> <p>A.16 Create Schedule</p>
17.12	Jul 20, 2017	<p>ACCESS_RULES-3-1-5 CREATE ACCESS PROFILE was updated according ticket # 1365.</p> <p>The following annexes were added according to #1365:</p> <p>A.17 Delete schedule</p> <p>A.18 Delete special day group</p>
17.12	Aug 16, 2017	<p>Current document name was changed from Access Rules Test Specification to Access Rules Device Test Specification.</p> <p>The document formatting was updated.</p>
18.06	Jun 21, 2018	Reformatting document using new template
18.12	Oct 16, 2018	<p>The following were updated in the scope of #1671:</p> <p>Scope\Access Profile (updated with SetAccessProfile)</p> <p>Test Policy\Access Profile (updated with SetAccessProfile)</p> <p>ACCESS_RULES-3-1-15 SET NEW ACCESS PROFILE (added)</p> <p>ACCESS_RULES-3-1-16 SET ACCESS PROFILE (added)</p> <p>Annex A.19 Create Pull Point Subscription (added)</p> <p>Annex A.20 Delete Subscription (added)</p> <p>Annex A.21 Retrieve Credential Changed Event by PullPoint (added)</p>
18.12	Dec 21, 2018	Switching Hub description in 'Network Configuration for DUT' section was updated according to #1737

Table of Contents

- 1 Introduction 7**
 - 1.1 Scope 7
 - 1.1.1 Capabilities 8
 - 1.1.2 Access Profile Info 8
 - 1.1.3 Access Profile 8
 - 1.1.4 Events 9
 - 1.1.5 Consistency 9
- 2 Terms and Definitions 10**
 - 2.1 Definitions 10
 - 2.2 Abbreviations 10
- 3 Test Overview 11**
 - 3.1 Test Setup 11
 - 3.1.1 Network Configuration for DUT 11
 - 3.2 Prerequisites 12
 - 3.3 Test Policy 12
 - 3.3.1 Capabilities 12
 - 3.3.2 Access Profile Info 13
 - 3.3.3 Access Profile 14
 - 3.3.4 Events 17
 - 3.3.5 Consistency 17
- 4 Access Rules Test Cases 18**
 - 4.1 Capabilities 18
 - 4.1.1 ACCESS RULES SERVICE CAPABILITIES 18
 - 4.1.2 GET SERVICES AND GET ACCESS RULES SERVICE CAPABILITIES
CONSISTENCY 18
 - 4.2 Access Profile Info 20
 - 4.2.1 GET ACCESS PROFILE INFO 20
 - 4.2.2 GET ACCESS PROFILE INFO LIST - LIMIT 22
 - 4.2.3 GET ACCESS PROFILE INFO LIST - START REFERENCE AND LIMIT 24
 - 4.2.4 GET ACCESS PROFILE INFO LIST - NO LIMIT 27

4.2.5	GET ACCESS PROFILE INFO WITH INVALID TOKEN	29
4.2.6	GET ACCESS PROFILE INFO - TOO MANY ITEMS	30
4.3	Access Profile	32
4.3.1	GET ACCESS PROFILES	32
4.3.2	GET ACCESS PROFILE LIST - LIMIT	34
4.3.3	GET ACCESS PROFILE LIST - START REFERENCE AND LIMIT	36
4.3.4	GET ACCESS PROFILE LIST - NO LIMIT	40
4.3.5	CREATE ACCESS PROFILE	43
4.3.6	MODIFY ACCESS PROFILE	47
4.3.7	DELETE ACCESS PROFILE	51
4.3.8	GET ACCESS PROFILES WITH INVALID TOKEN	54
4.3.9	GET ACCESS PROFILES - TOO MANY ITEMS	55
4.3.10	CREATE ACCESS PROFILE - NOT EMPTY ACCESS PROFILE TOKEN	56
4.3.11	CREATE ACCESS PROFILE - MULTIPLE SCHEDULES NOT SUPPORTED	57
4.3.12	MODIFY ACCESS PROFILE WITH INVALID TOKEN	59
4.3.13	DELETE ACCESS PROFILE WITH INVALID TOKEN	60
4.3.14	SET NEW ACCESS PROFILE	61
4.3.15	SET ACCESS PROFILE	66
4.4	Events	71
4.4.1	ACCESS PROFILE CHANGED EVENT	71
4.4.2	ACCESS PROFILE REMOVED EVENT	72
4.5	Consistency	74
4.5.1	ACCESS POLICIES AND ACCESS POINT CONSISTENCY	74
A	Helper Procedures and Additional Notes	76
A.1	Get access profiles information list	76
A.2	Get service capabilities	77
A.3	Get access profiles list	77
A.4	Get schedule list	78
A.5	Get access point information list	80

A.6	Delete access profile	81
A.7	Free storage for additional access profile	81
A.8	Get access profile	82
A.9	Get access profile info	83
A.10	Restore access profile	84
A.11	Create access profile	84
A.12	Get service capabilities (Schedule)	85
A.13	Generate UID value for iCalendar	86
A.14	Generate iCalendar value for Schedule	86
A.15	Create special day group	88
A.16	Create Schedule	89
A.17	Delete schedule	90
A.18	Delete special day group	91
A.19	Create Pull Point Subscription	91
A.20	Delete Subscription	92
A.21	Retrieve Access Profile Changed Event by PullPoint	93

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 Access Rules 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 Access Rules 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 Access Rules Service, which is defined in [ONVIF Access Rules Service].

The principal intended purposes are:

- Provide self-assessment tool for implementations.
- Provide comprehensive test suite coverage for [ONVIF Network Interface Specs].

This specification **does not** address the following:

- Product use cases and non-functional (performance and regression) testing.
- SOAP Implementation Interoperability test i.e. Web Service Interoperability Basic Profile version 2.0 (WS-I BP 2.0).
- Network protocol implementation Conformance test for HTTP, HTTPS, RTP and RTSP protocol.
- 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 Access Rules Test Specification covers the ONVIF Access Rules 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 capabilities with GetServiceCapabilities command
- Getting capabilities with GetServices command via Device service

1.1.2 Access Profile Info

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

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

- Getting access profile information with GetAccessProfileInfo command
- Getting access profile information list with GetAccessProfileInfoList command

1.1.3 Access Profile

The Access Profile section covers the test cases needed for getting access profile from an ONVIF device.

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

- Getting access profile with GetAccessProfiles command
- Getting access profile list with GetAccessProfileList command
- Creating access profile with CreateAccessProfile command
- Modifying access profile with ModifyAccessProfile command
- Deleting access profile with DeleteAccessProfile command
- Creating and modifying access profile with SetAccessProfile command

1.1.4 Events

The 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

1.1.5 Consistency

Consistency test cases cover verification of consistency between different entities and commands.

Consistency between the following entities is covered by the following test case:

- Access Policies and Access Point

2 Terms and Definitions

2.1 Definitions

This section describes terms and definitions used in this document.

Access Policy	An association of an access point and a schedule. It defines when an access point can be accessed using an access profile which contains this access policy. Several access policies specifying different schedules for the same access point will result in a union of the schedule.
Access Profile	A collection of access policies, used to define role based access. Two types of access profile can be defined for example: standard access profile and temporary access profile. The difference between the two, is that temporary access profile have optional validity from / to dates, and standard access profile do not.
Access Point	A logical composition of a physical door and ID point(s) controlling access in one direction.
Credentials	A physical/tangible object, a piece of knowledge, or a facet of a person's physical being, that enables an individual access to a given physical facility or computer-based information system.
Credential Number	A sequence of bytes uniquely identifying a credential at an access point.
Validity Period	From a certain point in time, to a later point in time. If a validity period is set on several entities (such as credentials, access profile and the association between them) the resulting validity period is the intersection of the three period.
Schedule	A set of time periods, for example: working hours (weekdays from 08:00 AM to 06:00 PM). It may also include one or more special days schedule.
Role based access	It is a method of regulating access to resources, for example: premises, building etc.

2.2 Abbreviations

This section describes abbreviations used in this document.

DUT	Device Under Test
HTTP	Hypertext Transfer Protocol
PACS	Physical Access Control System

3 Test Overview

This section describes about the test setup and prerequisites needed, and the test policies that should be followed for test case execution.

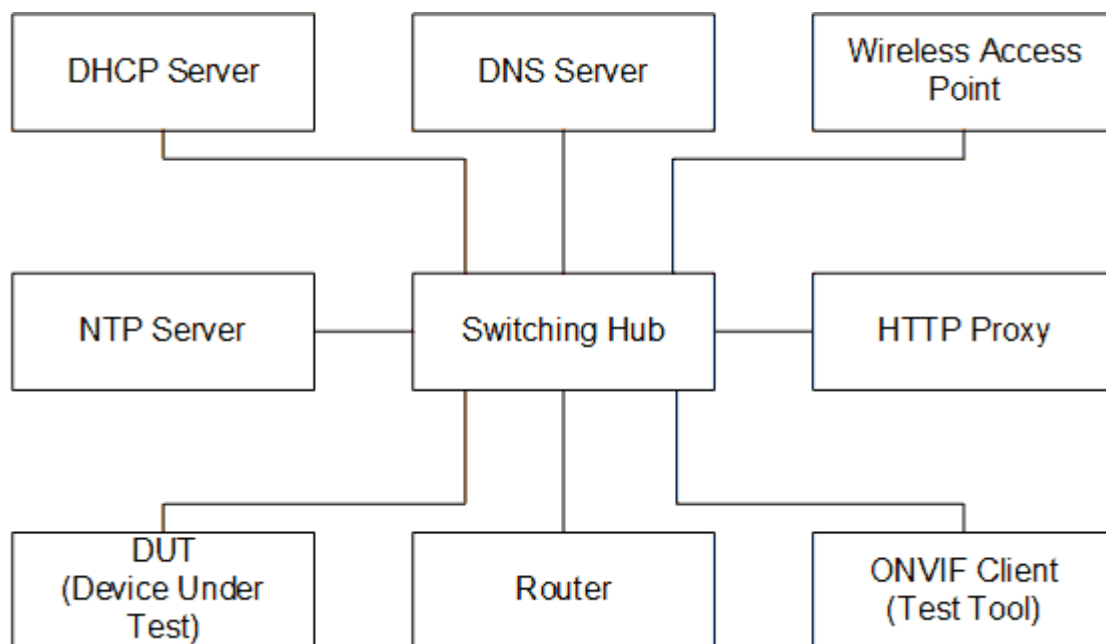
3.1 Test Setup

3.1.1 Network Configuration for DUT

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

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

Figure 3.1. 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.

3.2 Prerequisites

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

- The DUT shall be configured with an IPv4 address.
- The DUT shall be IP reachable [in the test configuration].
- The DUT shall be able to be discovered by the Test Tool.
- The DUT shall be configured with the time i.e. manual configuration of UTC time and if NTP is supported by DUT, then NTP time shall be synchronized with NTP Server.

3.3 Test Policy

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

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

3.3.1 Capabilities

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

- DUT shall give the Access Rules Service entry point by GetServices command, if DUT supports this service. If DUT does not support Access Rules Service, 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 4.1](#) for Capabilities Test Cases.

3.3.2 Access Profile Info

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

- DUT shall give the Access Rules Service entry point by GetServices command, if DUT supports this service. If DUT does not support Access Rules Service, these test cases will be skipped.
- DUT shall support the following commands:
 - GetServices
 - GetServiceCapabilities
 - GetAccessProfileInfo
 - GetAccessProfileInfoList
- DUT shall not return more items in GetAccessProfileInfo and GetAccessProfileInfoList responses than specified in service capabilities by MaxLimit.
- DUT shall not return more items in GetAccessProfileInfoList response than specified by Limit parameter in a request.
- DUT shall not return more AccessProfilesInfo items in GetAccessProfileInfoList responses than specified in service capabilities by MaxAccessProfiles.
- DUT shall not return items with the same tokens in GetAccessProfileInfoList responses for one access profile info list receiving.
- DUT shall not return any fault if GetAccessProfileInfo was invoked for non-existing AccessProfile token. Such tokens shall be ignored.
- DUT shall return SOAP 1.2 fault message (InvalidArgs/TooManyItems) if more items than MaxLimit was requested by GetAccessProfileInfo command.
- The following tests are performed

- Getting access profile info with GetAccessProfileInfo command
- Getting access profile info list with GetAccessProfileInfoList command with using different Limit and NextReference values
- Getting access profile info with invalid access profile token
- Getting access profile info with number of requested items is greater than MaxLimit

Please refer to [Section 4.2](#) for Access Profile Info Test Cases.

3.3.3 Access Profile

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

- DUT shall give the Access Rules 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
 - GetAccessProfiles
 - GetAccessProfileList
 - GetAccessProfileInfoList
 - CreateAccessProfile
 - ModifyAccessProfile
 - DeleteAccessProfile
- If DUT supports Client Supplied Token as indicated by Capabilities.ClientSuppliedTokenSupported = true, DUT shall support the following commands:
 - SetAccessProfile
- DUT shall return only requested items in GetAccessProfiles response that specified in GetAccessProfiles request.
- DUT shall return all requested items in GetAccessProfiles response that specified in GetAccessProfiles request.

- DUT shall not return more items in GetAccessProfiles responses than specified in service capabilities by MaxLimit.
- DUT shall return the same information in GetAccessProfiles responses and in GetAccessProfileInfoList responses for the items with the same token.
- DUT shall not return more items in GetAccessProfileList response than specified by Limit parameter in a request.
- DUT shall not return items with the same tokens in GetAccessProfileList responses for one access profile list receiving.
- DUT shall return the same information in GetAccessProfiles responses and in GetAccessProfilesList responses for the items with the same token.
- DUT shall return the same information in GetAccessProfileList responses and in GetAccessProfileInfoList responses for the items with the same token.
- DUT shall return the access profiles in GetAccessProfileList responses and in GetAccessProfileInfoList responses.
- DUT shall support creation of access profile.
- DUT shall support modifying of access profile.
- DUT shall support deletion of access profile.
- If DUT supports Client Supplied Token as indicated by Capabilities.ClientSuppliedTokenSupported = true, creating and modifying of access profile using SetAccessProfile command.
- DUT shall not return any fault if GetAccessProfiles was invoked for non-existing AccessProfile token. Such tokens shall be ignored.
- DUT shall return SOAP 1.2 fault message (InvalidArgs/TooManyItems) if more items than MaxLimit was requested by GetAccessProfiles command.
- The DUT shall support creating of access profile.
- The DUT shall support modifying of access profile.
- The DUT shall support deleting of access profile.
- DUT shall return SOAP 1.2 fault message (InvalidArgs) if access profile token is specified in CreateAccessProfile request.
- DUT should return SOAP 1.2 fault message (CapabilityViolated/MultipleSchedulesPerAccessPointSupported) if multiple schedules for the same access point

is sent in the create access profile and MultipleSchedulesPerAccessPointSupported is not supported.

- DUT should return SOAP 1.2 fault message (InvalidArgVal/NotFound) if ModifyAccessProfile or DeleteAccessProfile command was invoked for non-existing access profile token.
- The following tests are performed
 - Getting access profile with GetAccessProfiles command and test that it includes the same information with GetAccessProfileList command
 - Getting access profile info list with GetAccessProfileList command with using different Limit and NextReference values and test that it includes the same information with GetAccessProfileInfoList command
 - Creating access profile with CreateAccessProfile command with empty token and test that corresponding notification message is received
 - Modifying access profile with ModifyAccessProfile command and test that corresponding notification message is received
 - Deleting access profile with DeleteAccessProfile command and test that corresponding notification message is received
 - Getting access profiles with invalid access profile token
 - Getting access profiles with number of requested items is greater than MaxLimit
 - Creating access profile with CreateAccessProfile command with specified token
 - Creating access profile with multiple schedules for the same access point when MultipleSchedulesPerAccessPointSupported is not supported
 - Modifying access profile with ModifyAccessProfile command with invalid token
 - Deleting access profile with DeleteAccessProfile command with invalid token
 - If DUT supports Client Supplied Token as indicated by Capabilities.ClientSuppliedTokenSupported = true:
 - Creating access profile with SetAccessProfile command with empty token and test that corresponding notification message is received
 - Modifying access profile with SetAccessProfile command and test that corresponding notification message is received

Please refer to [Section 4.3](#) for Access Profile Test Cases.

3.3.4 Events

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

- DUT shall give the Access Rules 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:
 - GetServices
 - GetEventProperties
- DUT shall support Pull Point Subscription and Topic Expression filter.
- DUT shall generate property events with initial state after subscription was done.
- DUT shall generate property events with current state after corresponding properties were changed.
- The following tests are performed
 - Getting event properties with GetEventProperties command

Please refer to [Section 4.4](#) for Events Test Cases.

3.3.5 Consistency

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

- DUT shall give the Access Rules Service and Access Control Service entry points by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.
- DUT shall support the following commands:
 - GetServices
 - GetAccessProfileInfo
 - GetCredentials
- The following tests are performed
 - Access Policies and Access Point consistency

Please refer to [Section 4.5](#) for Consistency Test Cases.

4 Access Rules Test Cases

4.1 Capabilities

4.1.1 ACCESS RULES SERVICE CAPABILITIES

Test Case ID: ACCESS_RULES-1-1-1

Specification Coverage: ServiceCapabilities (ONVIF Access Rules Service Specification), GetServiceCapabilities command (ONVIF Access Rules Service Specification)

Feature Under Test: GetServiceCapabilities (for Access Rules Service)

WSDL Reference: accessrules.wsdl

Test Purpose: To verify DUT Access Rules Service Capabilities.

Pre-Requirement: Access Rules Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServiceCapabilities**.
4. The DUT responds with **GetServiceCapabilitiesResponse** message with parameters
 - Capabilities =: *cap*

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

4.1.2 GET SERVICES AND GET ACCESS RULES SERVICE CAPABILITIES CONSISTENCY

Test Case ID: ACCESS_RULES-1-1-2

Specification Coverage: Capability exchange (ONVIF Core Specification), ServiceCapabilities (ONVIF Access Rules Service Specification), GetServiceCapabilities command (ONVIF Access Rules Service Specification)

Feature Under Test: GetServices, GetServiceCapabilities (for Access Rules Service)

WSDL Reference: devicemgmt.wsdl, accessrules.wsdl

Test Purpose: To verify Get Services and Access Rules Service Capabilities consistency.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Note: The following fields are compared at step 8:

- MaxLimit
- MaxAccessProfiles
- MaxAccessPoliciesPerAccessProfile
- MultipleSchedulesPerAccessPointSupported

4.2 Access Profile Info

4.2.1 GET ACCESS PROFILE INFO

Test Case ID: ACCESS_RULES-2-1-1

Specification Coverage: AccessProfileInfo (ONVIF Access Rules Service Specification), GetAccessProfileInfo command (ONVIF Access Rules Service Specification)

Feature Under Test: GetAccessProfileInfo

WSDL Reference: accessrules.wsdl

Test Purpose: To verify Get Access Profile Info.

Pre-Requisite: Access Rules 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 retrieves a complete list of access profile info (out *accessProfileInfoCompleteList*) by following the procedure mentioned in [Annex A.1](#).
4. If *accessProfileInfoCompleteList* is empty, skip other steps.
5. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in [Annex A.2](#).

6. Set the following:
 - *tokenList* := [subset of *accessProfileInfoCompleteList.token* values with items number equal to *cap.MaxLimit*]
7. ONVIF client invokes **GetAccessProfileInfo** with parameters
 - Token list := *tokenList*
8. The DUT responds with **GetAccessProfileInfoResponse** message with parameters
 - AccessProfileInfo list =: *accessProfileInfoList1*
9. If *accessProfileInfoList1* does not contain AccessProfileInfo item for each token from *tokenList*, FAIL the test and skip other steps.
10. If *accessProfileInfoList1* contains at least two AccessProfileInfo item with equal token, FAIL the test and skip other steps.
11. If *accessProfileInfoList1* contains other AccessProfileInfo items then listed in *tokenList*, FAIL the test and skip other steps.
12. For each AccessProfile.token *token* from *accessProfileInfoCompleteList* repeat the following steps:
 - 12.1. ONVIF client invokes **GetAccessProfileInfo** with parameters
 - Token[0] := *token*
 - 12.2. The DUT responds with **GetAccessProfileInfoResponse** message with parameters
 - AccessProfileInfo list =: *accessProfileInfoList2*
 - 12.3. If *accessProfileInfoList2* does not contain only one AccessProfileInfo item with token equal to *token*, FAIL the test and skip other steps.
 - 12.4. If *accessProfileInfoList2*[0] item is not equal to *accessProfileInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

Note: If number of items in *accessProfileInfoCompleteList*, less than Capabilities =: *cap.MaxLimit*, then all *accessProfileInfoCompleteList.Token* items shall be used for the step 6.

Note: The following fields are compared at step 12.4:

- AccessProfileInfo:
 - token
 - Name
 - Description

4.2.2 GET ACCESS PROFILE INFO LIST - LIMIT

Test Case ID: ACCESS_RULES-2-1-2

Specification Coverage: AccessProfileInfo (ONVIF Access Rules Service Specification), GetAccessProfileInfoList command (ONVIF Access Rules Service Specification)

Feature Under Test: GetAccessProfileInfoList

WSDL Reference: accessrules.wsdl

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

Pre-Requirement: Access Rules Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in [Annex A.2](#).
4. ONVIF client invokes **GetAccessProfileInfoList** with parameters
 - Limit := 1
 - StartReference skipped
5. The DUT responds with **GetAccessProfileInfoListResponse** message with parameters
 - NextStartReference =: *nextStartReference*

- *AccessProfileInfo* list =: *accessProfileInfoList1*
6. If *accessProfileInfoList1* contains more *AccessProfileInfo* items than 1, FAIL the test and skip other steps.
 7. If *cap.MaxLimit* is equal to 1, skip other steps.
 8. ONVIF client invokes **GetAccessProfileInfoList** with parameters
 - Limit := *cap.MaxLimit*
 - StartReference skipped
 9. The DUT responds with **GetAccessProfileInfoListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - *AccessProfileInfo* list =: *accessProfileInfoList2*
 10. If *accessProfileInfoList2* contains more *AccessProfileInfo* items than *cap.MaxLimit*, FAIL the test and skip other steps.
 11. If *cap.MaxLimit* is equal to 2, skip other steps.
 12. Set the following:
 - *limit* := [number between 1 and *cap.MaxLimit*]
 13. ONVIF client invokes **GetAccessProfileInfoList** with parameters
 - Limit := *limit*
 - StartReference skipped
 14. The DUT responds with **GetAccessProfileInfoListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - *AccessProfileInfo* list =: *accessProfileInfoList3*
 15. If *accessProfileInfoList3* contains more *AccessProfileInfo* items than *limit*, FAIL the test and skip other steps.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

4.2.3 GET ACCESS PROFILE INFO LIST - START REFERENCE AND LIMIT

Test Case ID: ACCESS_RULES-2-1-3

Specification Coverage: AccessProfileInfo (ONVIF Access Rules Service Specification), GetAccessProfileInfoList command (ONVIF Access Rules Service Specification)

Feature Under Test: GetAccessProfileInfoList

WSDL Reference: accessrules.wsdl

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

Pre-Requirement: Access Rules Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in [Annex A.2](#).
4. ONVIF client invokes **GetAccessProfileInfoList** with parameters
 - Limit := *cap.MaxLimit*
 - StartReference skipped
5. The DUT responds with **GetAccessProfileInfoListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - AccessProfileInfo list =: *accessProfileInfoList1*
6. If *accessProfileInfoList1* contains more AccessProfileInfo items than *cap.MaxLimit*, FAIL the test and skip other steps.
7. Until *nextStartReference* is not null, repeat the following steps:

- 7.1. ONVIF client invokes **GetAccessProfileInfoList** with parameters
 - Limit := *cap.MaxLimit*
 - StartReference := *nextStartReference*
- 7.2. The DUT responds with **GetAccessProfileInfoListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - AccessProfileInfo list =: *accessProfileInfoListPart*
- 7.3. If *accessProfileInfoListPart* contains more AccessProfileInfo items than *cap.MaxLimit*, FAIL the test and skip other steps.
- 7.4. Set the following:
 - $accessProfileInfoCompleteList1 := accessProfileInfoCompleteList1 + accessProfileInfoListPart$
8. If *accessProfileInfoCompleteList1* contains at least two AccessProfileInfo items with equal token, FAIL the test and skip other steps.
9. If *cap.MaxLimit* is equal to 1, skip other steps.
10. ONVIF client invokes **GetAccessProfileInfoList** with parameters
 - Limit := 1
 - StartReference skipped
11. The DUT responds with **GetAccessProfileInfoListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - AccessProfileInfo list =: *accessProfileInfoCompleteList2*
12. If *accessProfileInfoCompleteList2* contains more AccessProfileInfo items than 1, FAIL the test and skip other steps.
13. Until *nextStartReference* is not null, repeat the following steps:
 - 13.1. ONVIF client invokes **GetAccessProfileInfoList** with parameters
 - Limit := 1
 - StartReference := *nextStartReference*

13.2. The DUT responds with **GetAccessProfileInfoListResponse** message with parameters

- NextStartReference =: *nextStartReference*
- AccessProfileInfo list =: *accessProfileInfoListPart*

13.3. If *accessProfileInfoListPart* contains more AccessProfileInfo items than 1, FAIL the test and skip other steps.

13.4. Set the following:

- $accessProfileInfoCompleteList2 := accessProfileInfoCompleteList2 + accessProfileInfoListPart$

14. If *accessProfileInfoCompleteList2* contains at least two AccessProfileInfo item with equal token, FAIL the test and skip other steps.

15. If *accessProfileInfoCompleteList2* does not contain all access profiles from *accessProfileInfoCompleteList1*, FAIL the test and skip other steps.

16. If *accessProfileInfoCompleteList2* contains access profiles other than access profiles from *accessProfileInfoCompleteList1*, FAIL the test and skip other steps.

17. If *cap.MaxLimit* is equal to 2, skip other steps.

18. Set the following:

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

19. ONVIF client invokes **GetAccessProfileInfoList** with parameters

- Limit := *limit*
- StartReference skipped

20. The DUT responds with **GetAccessProfileInfoListResponse** message with parameters

- NextStartReference =: *nextStartReference*
- AccessProfileInfo list =: *accessProfileInfoCompleteList3*

21. If *accessProfileInfoCompleteList3* contains more AccessProfileInfo items than *limit*, FAIL the test and skip other steps.

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

22.1. ONVIF client invokes **GetAccessProfileInfoList** with parameters

- Limit := *limit*
- StartReference := *nextStartReference*

22.2. The DUT responds with **GetAccessProfileInfoListResponse** message with parameters

- NextStartReference =: *nextStartReference*
- AccessProfileInfo list =: *accessProfileInfoListPart*

22.3. If *accessProfileInfoListPart* contains more AccessProfileInfo items than *limit*, FAIL the test and skip other steps.

22.4. Set the following:

- $accessProfileInfoCompleteList3 := accessProfileInfoCompleteList3 + accessProfileInfoListPart$

23. If *accessProfileInfoCompleteList3* contains at least two AccessProfileInfo item with equal token, FAIL the test and skip other steps.

24. If *accessProfileInfoCompleteList3* does not contain all access profiles from *accessProfileInfoCompleteList1*, FAIL the test and skip other steps.

25. If *accessProfileInfoCompleteList3* contains access profiles other than access profiles from *accessProfileInfoCompleteList1*, FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

4.2.4 GET ACCESS PROFILE INFO LIST - NO LIMIT

Test Case ID: ACCESS_RULES-2-1-4

Specification Coverage: AccessProfileInfo (ONVIF Access Rules Service Specification), GetAccessProfileInfoList command (ONVIF Access Rules Service Specification)

Feature Under Test: GetAccessProfileInfoList

WSDL Reference: accessrules.wsdl

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

Pre-Requisite: Access Rules Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in [Annex A.2](#).
4. ONVIF client invokes **GetAccessProfileInfoList** with parameters
 - Limit skipped
 - StartReference skipped
5. The DUT responds with **GetAccessProfileInfoListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - AccessProfileInfo list =: *accessProfileInfoCompleteList*
6. If *accessProfileInfoCompleteList* contains more AccessProfileInfo items than *cap.MaxLimit*, FAIL the test and skip other steps.
7. Until *nextStartReference* is not null, repeat the following steps:
 - 7.1. ONVIF client invokes **GetAccessProfileInfoList** with parameters
 - Limit skipped
 - StartReference := *nextStartReference*
 - 7.2. The DUT responds with **GetAccessProfileInfoListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - AccessProfileInfo list =: *accessProfileInfoListPart*
 - 7.3. If *accessProfileInfoListPart* contains more AccessProfileInfo items than *cap.MaxLimit*, FAIL the test and skip other steps.

7.4. Set the following:

- $accessProfileInfoCompleteList := accessProfileInfoCompleteList + accessProfileInfoListPart$

8. If *accessProfileInfoCompleteList* contains at least two *AccessProfileInfo* items with equal token, FAIL the test.
9. If *accessProfileInfoCompleteList* contains more *AccessProfileInfo* items than *cap.MaxAccessProfiles*, FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

4.2.5 GET ACCESS PROFILE INFO WITH INVALID TOKEN

Test Case ID: ACCESS_RULES-2-1-5

Specification Coverage: GetAccessProfileInfo command (ONVIF Access Rules Service Specification)

Feature Under Test: GetAccessProfileInfo

WSDL Reference: accessrules.wsdl

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

Pre-Requisite: Access Rules 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 retrieves a complete list of access profiles (out *accessProfileInfoCompleteList*) by following the procedure mentioned in [Annex A.1](#).
4. Set the following:

- *invalidToken* := value not equal to any *accessProfileInfoCompleteList.token*
5. ONVIF client invokes **GetAccessProfileInfo** with parameters
 - Token list := *invalidToken*
 6. The DUT responds with **GetAccessProfileInfoResponse** message with parameters
 - AccessProfileInfo list =: *accessProfileInfoList*
 7. If *accessProfileInfoList* is not empty, FAIL the test.
 8. If *accessProfileInfoCompleteList* is empty, skip other steps.
 9. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in [Annex A.2](#).
 10. If *cap.MaxLimit* is less than 2, skip other steps.
 11. ONVIF client invokes **GetAccessProfileInfo** with parameters
 - Token[0]:= *invalidToken*
 - Token[1]:= *accessProfileInfoCompleteList[0].token*
 12. The DUT responds with **GetAccessProfileInfoResponse** message with parameters
 - AccessProfileInfo list =: *accessProfileInfoList*
 13. If *accessProfileInfoList* is empty, FAIL the test.
 14. If *accessProfileInfoList* contains more than one item, FAIL the test.
 15. If *accessProfileInfoList[0].token* does not equal *accessProfileInfoCompleteList[0].token*, FAIL the test.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

4.2.6 GET ACCESS PROFILE INFO - TOO MANY ITEMS

Test Case ID: ACCESS_RULES-2-1-6

Specification Coverage: AccessProfileInfo (ONVIF Access Rules Service Specification), GetAccessProfileInfo command (ONVIF Access Rules Service Specification)

Feature Under Test: GetAccessProfileInfo

WSDL Reference: accessrules.wsdl

Test Purpose: To verify Get Access Profile Info in case if there a more items than MaxLimit in request.

Pre-Requisite: Access Rules 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 retrieves a complete list of access profiles (out *accessProfileInfoCompleteList*) by following the procedure mentioned in [Annex A.1](#).
4. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in [Annex A.2](#).
5. If *accessProfileInfoCompleteList.token* items number is less than *cap.MaxLimit* or equal to *cap.MaxLimit*, skip other steps.
6. Set the following:
 - *tokenList* := [subset of *accessProfileInfoCompleteList.token* values with items number equal to *cap.MaxLimit* + 1]
7. ONVIF client invokes **GetAccessProfileInfo** with parameters
 - Token list := *tokenList*
8. The DUT returns **env:Sender\ter:InvalidArgs\ter:TooManyItems** SOAP 1.2 fault.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

4.3 Access Profile

4.3.1 GET ACCESS PROFILES

Test Case ID: ACCESS_RULES-3-1-1

Specification Coverage: AccessProfileInfo (ONVIF Access Rules Service Specification), AccessProfile (ONVIF Access Rules Service Specification), GetAccessProfiles command (ONVIF Access Rules Service Specification)

Feature Under Test: GetAccessProfiles

WSDL Reference: accessrules.wsdl

Test Purpose: To verify Get Access Profiles.

Pre-Requisite: Access Rules 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 retrieves a complete list of access profiles (out *accessProfileCompleteList*) by following the procedure mentioned in [Annex A.3](#).
4. If *accessProfileCompleteList* is empty, skip other steps.
5. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in [Annex A.2](#).
6. Set the following:
 - *tokenList* := [subset of *accessProfileCompleteList.token* values with items number equal to *cap.MaxLimit*]
7. ONVIF client invokes **GetAccessProfiles** with parameters
 - Token list := *tokenList*
8. The DUT responds with **GetAccessProfilesResponse** message with parameters
 - AccessProfile list =: *accessProfileList1*

9. If *accessProfileList1* does not contain *AccessProfile* item for each token from *tokenList*, FAIL the test and skip other steps.
10. If *accessProfileList1* contains at least two *AccessProfile* items with equal token, FAIL the test and skip other steps.
11. If *accessProfileList1* contains other *AccessProfile* items then listed in *tokenList*, FAIL the test and skip other steps.
12. For each *AccessProfileInfo.token* token from *accessProfileCompleteList* repeat the following steps:
 - 12.1. ONVIF client invokes **GetAccessProfiles** with parameters
 - *Token[0]* := *token*
 - 12.2. The DUT responds with **GetAccessProfilesResponse** message with parameters
 - *AccessProfile* list =: *accessProfileList2*
 - 12.3. If *accessProfileList2* does not contain only one *AccessProfile* item with token equal to *token*, FAIL the test and skip other steps.
 - 12.4. If *accessProfileList2[0]* item does not have equal field values to *accessProfileCompleteList[token = token]* item, FAIL the test and skip other steps.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

Note: If number of items in *accessProfileCompleteList*, less than *cap.MaxLimit*, then all *accessProfileCompleteList.Token* items shall be used for the step 6.

Note: The following fields are compared at step 12.4:

- *AccessProfile*:
 - *token*
 - *Name*
 - *Description*

- AccessPolicy list (ScheduleToken is used as unique key for comparing)
 - ScheduleToken
 - Entity
 - EntityType

4.3.2 GET ACCESS PROFILE LIST - LIMIT

Test Case ID: ACCESS_RULES-3-1-2

Specification Coverage: AccessProfileInfo (ONVIF Access Rules Service Specification), AccessProfile (ONVIF Access Rules Service Specification), GetAccessProfileList command (ONVIF Access Rules Service Specification)

Feature Under Test: GetAccessProfileList

WSDL Reference: accessrules.wsdl

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

Pre-Requisite: Access Rules Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in [Annex A.2](#).
4. ONVIF client invokes **GetAccessProfileList** with parameters
 - Limit := 1
 - StartReference skipped
5. The DUT responds with **GetAccessProfileListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - AccessProfile list =: *accessProfileList1*

6. If *accessProfileList1* contains more AccessProfile items than 1, FAIL the test and skip other steps.
7. If *cap.MaxLimit* is equal to 1, skip other steps.
8. ONVIF client invokes **GetAccessProfileList** with parameters
 - Limit := *cap.MaxLimit*
 - StartReference skipped
9. The DUT responds with **GetAccessProfileListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - AccessProfile list =: *accessProfileList2*
10. If *accessProfileList2* contains more AccessProfile items than *cap.MaxLimit*, FAIL the test and skip other steps.
11. If *cap.MaxLimit* is equal to 2, skip other steps.
12. Set the following:
 - *limit* := [number between 1 and *cap.MaxLimit*]
13. ONVIF client invokes **GetAccessProfileList** with parameters
 - Limit := *limit*
 - StartReference skipped
14. The DUT responds with **GetAccessProfileListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - AccessProfile list =: *accessProfileList3*
15. If *accessProfileList3* contains more AccessProfile items than *limit*, FAIL the test and skip other steps.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

4.3.3 GET ACCESS PROFILE LIST - START REFERENCE AND LIMIT

Test Case ID: ACCESS_RULES-3-1-3

Specification Coverage: AccessProfileInfo (ONVIF Access Rules Service Specification), AccessProfile (ONVIF Access Rules Service Specification), GetAccessProfileList command (ONVIF Access Rules Service Specification)

Feature Under Test: GetAccessProfileList

WSDL Reference: accessrules.wsdl

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

Pre-Requisite: Access Rules Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in [Annex A.2](#).
4. ONVIF client invokes **GetAccessProfileList** with parameters
 - Limit := *cap.MaxLimit*
 - StartReference skipped
5. The DUT responds with **GetAccessProfileListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - AccessProfile list =: *accessProfileCompleteList1*
6. If *accessProfileCompleteList1* contains more AccessProfile items than *cap.MaxLimit*, FAIL the test and skip other steps.
7. Until *nextStartReference* is not null, repeat the following steps:
 - 7.1. ONVIF client invokes **GetAccessProfileList** with parameters

- Limit := *cap.MaxLimit*
 - StartReference := *nextStartReference*
- 7.2. The DUT responds with **GetAccessProfileListResponse** message with parameters
- NextStartReference =: *nextStartReference*
 - AccessProfile list =: *accessProfileListPart*
- 7.3. If *accessProfileListPart* contains more AccessProfile items than *cap.MaxLimit*, FAIL the test and skip other steps.
- 7.4. Set the following:
- *accessProfileCompleteList1* := *accessProfileCompleteList1* + *accessProfileListPart*
8. If *accessProfileCompleteList1* contains at least two AccessProfile item with equal token, FAIL the test and skip other steps.
9. If *cap.MaxLimit* is equal to 1, do the following steps:
- 9.1. ONVIF Client retrieves a complete list of access profiles (out *accessProfileInfoCompleteList*) by following the procedure mentioned in [Annex A.1](#).
- 9.2. If *accessProfileCompleteList1* does not contain all access profiles from *accessProfileInfoCompleteList*, FAIL the test and skip other steps.
- 9.3. If *accessProfileCompleteList1* contains access profiles other than access profiles from *accessProfileInfoCompleteList*, FAIL the test and skip other steps.
- 9.4. For each AccessProfileInfo.token *token* from *accessProfileInfoCompleteList* repeat the following steps:
- 9.4.1. If *accessProfileCompleteList1*[token = *token*] item does not have equal field values to *accessProfileInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.
- 9.5. Skip other steps.
10. ONVIF client invokes **GetAccessProfileList** with parameters
- Limit := 1
 - StartReference skipped
11. The DUT responds with **GetAccessProfileListResponse** message with parameters

- NextStartReference =: *nextStartReference*
 - AccessProfile list =: *accessProfileCompleteList2*
12. If *accessProfileCompleteList2* contains more AccessProfile items than 1, FAIL the test and skip other steps.
13. Until *nextStartReference* is not null, repeat the following steps:
- 13.1. ONVIF client invokes **GetAccessProfileList** with parameters
- Limit := 1
 - StartReference := *nextStartReference*
- 13.2. The DUT responds with **GetAccessProfileListResponse** message with parameters
- NextStartReference =: *nextStartReference*
 - Access Profile list =: *accessProfileListPart*
- 13.3. If *accessProfileListPart* contains more AccessProfile items than 1, FAIL the test and skip other steps.
- 13.4. Set the following:
- *accessProfileCompleteList2* := *accessProfileCompleteList2* + *accessProfileListPart*
14. If *accessProfileCompleteList2* contains at least two AccessProfile item with equal token, FAIL the test and skip other steps.
15. If *accessProfileCompleteList2* does not contain all access profiles from *accessProfileCompleteList1*, FAIL the test and skip other steps.
16. If *accessProfileCompleteList2* contains access profiles other than access profiles from *accessProfileCompleteList1*, FAIL the test and skip other steps.
17. If *cap.MaxLimit* is equal to 2, do the following steps:
- 17.1. ONVIF Client retrieves a complete list of access profiles (out *accessProfileInfoCompleteList*) by following the procedure mentioned in [Annex A.1](#).
- 17.2. If *accessProfileCompleteList2* does not contain all access profiles from *accessProfileInfoCompleteList*, FAIL the test and skip other steps.
- 17.3. If *accessProfileCompleteList2* contains access profiles other than access profiles from *accessProfileInfoCompleteList*, FAIL the test and skip other steps.

17.4. For each `AccessProfileInfo.token` *token* from `accessProfileInfoCompleteList` repeat the following steps:

17.4.1. If `accessProfileCompleteList2[token = token]` item does not have equal field values to `accessProfileInfoCompleteList[token = token]` item, FAIL the test and skip other steps.

17.5. Skip other steps.

18. Set the following:

- *limit* := [number between 1 and `cap.MaxLimit`]

19. ONVIF client invokes **GetAccessProfileList** with parameters

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

20. The DUT responds with **GetAccessProfileListResponse** message with parameters

- `NextStartReference` =: *nextStartReference*
- `AccessProfile` list =: `accessProfileCompleteList3`

21. If `accessProfileCompleteList3` contains more `AccessProfile` items than *limit*, FAIL the test and skip other steps.

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

22.1. ONVIF client invokes **GetAccessProfileList** with parameters

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

22.2. The DUT responds with **GetAccessProfileListResponse** message with parameters

- `NextStartReference` =: *nextStartReference*
- `AccessProfile` list =: `accessProfileListPart`

22.3. If `accessProfileListPart` contains more `AccessProfile` items than *limit*, FAIL the test and skip other steps.

22.4. Set the following:

- `accessProfileCompleteList3` := `accessProfileCompleteList3` + `accessProfileListPart`

23. If *accessProfileCompleteList3* contains at least two *AccessProfile* item with equal token, FAIL the test and skip other steps.
24. If *accessProfileCompleteList3* does not contain all access profiles from *accessProfileCompleteList1*, FAIL the test and skip other steps.
25. If *accessProfileCompleteList3* contains access profiles other than access profiles from *accessProfileCompleteList1*, FAIL the test and skip other steps.
26. ONVIF Client retrieves a complete list of access profiles (out *accessProfileInfoCompleteList*) by following the procedure mentioned in [Annex A.1](#).
27. If *accessProfileCompleteList3* does not contain all access profiles from *accessProfileInfoCompleteList*, FAIL the test and skip other steps.
28. If *accessProfileCompleteList3* contains access profiles other than access profiles from *accessProfileInfoCompleteList*, FAIL the test and skip other steps.
29. For each *AccessProfileInfo.token token* from *accessProfileInfoCompleteList* repeat the following steps:
- 29.1. If *accessProfileCompleteList3[token = token]* item does not have equal field values to *accessProfileInfoCompleteList[token = token]* item, FAIL the test and skip other steps.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

Note: The following fields are compared at step [29.1](#):

- *AccessProfileInfo*:
 - token
 - Name
 - Description

4.3.4 GET ACCESS PROFILE LIST - NO LIMIT

Test Case ID: ACCESS_RULES-3-1-4

Specification Coverage: AccessProfileInfo (ONVIF Access Rules Service Specification), AccessProfile (ONVIF Access Rules Service Specification), GetAccessProfileList command (ONVIF Access Rules Service Specification)

Feature Under Test: GetAccessProfileList

WSDL Reference: accessrules.wsdl

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

Pre-Requisite: Access Rules Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in [Annex A.2](#).
4. ONVIF client invokes **GetAccessProfileList** with parameters
 - Limit skipped
 - StartReference skipped
5. The DUT responds with **GetAccessProfileListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - AccessProfile list =: *accessProfileCompleteList*
6. If *accessProfileCompleteList* contains more AccessProfile items than *cap.MaxLimit*, FAIL the test and skip other steps.
7. Until *nextStartReference* is not null repeat, the following steps:
 - 7.1. ONVIF client invokes **GetAccessProfileList** with parameters
 - Limit skipped
 - StartReference := *nextStartReference*
 - 7.2. The DUT responds with **GetAccessProfileListResponse** message with parameters
 - NextStartReference =: *nextStartReference*

- *AccessProfile* list =: *accessProfileListPart*
- 7.3. If *accessProfileListPart* contains more *AccessProfile* items than *cap.MaxLimit*, FAIL the test and skip other steps.
- 7.4. Set the following:
- *accessProfileCompleteList* := *accessProfileCompleteList* + *accessProfileListPart*
8. If *accessProfileCompleteList* contains at least two *AccessProfile* item with equal token, FAIL the test.
9. ONVIF Client retrieves a complete list of access profiles (out *accessProfileInfoCompleteList*) by following the procedure mentioned in [Annex A.1](#).
10. If *accessProfileCompleteList* does not contain all access profiles from *accessProfileInfoCompleteList*, FAIL the test and skip other steps.
11. If *accessProfileCompleteList* contains access profiles other than access profiles from *accessProfileInfoCompleteList*, FAIL the test and skip other steps.
12. For each *AccessProfileInfo.token token* from *accessProfileInfoCompleteList* repeat the following steps:
- 12.1. If *accessProfileCompleteList[token = token]* item does not have equal field values to *accessProfileInfoCompleteList[token = token]* item, FAIL the test and skip other steps.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

Note: The following fields are compared at step [12.1](#):

- *AccessProfileInfo*:
 - token
 - Name
 - Description

4.3.5 CREATE ACCESS PROFILE

Test Case ID: ACCESS_RULES-3-1-5

Specification Coverage: AccessProfileInfo (ONVIF Access Rules Service Specification), AccessProfile (ONVIF Access Rules Service Specification), CreateAccessProfile command (ONVIF Access Rules Service Specification).

Feature Under Test: CreateAccessProfile

WSDL Reference: accessrules.wsdl and event.wsdl

Test Purpose: To verify Create Access Profiles with empty token and to verify tns1:Configuration/AccessProfile/Changed event generation.

Pre-Requirement: Access Rules Service is received from the DUT. Schedule Service is received from the DUT. Event Service is received from the DUT. The DUT shall have enough free storage capacity for one additional Access Profile.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves a complete list of access profiles (out *accessProfileCompleteList1*) by following the procedure mentioned in [Annex A.3](#).
4. ONVIF Client checks free storage for additional Access Profile (in *accessProfileCompleteList1*, out *accessProfileToRestore*) by following the procedure mentioned in [Annex A.7](#).
5. If Access Control service is supported by the DUT, ONVIF Client retrieves a complete list of access point information (out *accessPointInfoCompleteList*) by following the procedure mentioned in [Annex A.5](#).
6. If *accessPointInfoCompleteList* is not empty:
 - 6.1. ONVIF Client retrieves a complete list of schedules (out *scheduleCompleteList*) by following the procedure mentioned in [Annex A.4](#).
 - 6.2. If *scheduleCompleteList* is not empty:
 - 6.2.1. Set *scheduleToken:= scheduleCompleteList[0].token*

- 6.2.2. Go to step 7.
- 6.3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in [Annex A.12](#)
- 6.4. ONVIF Client generates appropriate iCalendar value for the Schedule.Standard field (in *cap*, out *scheduleiCalendarValue*) by following the procedure mentioned in [Annex A.14](#).
- 6.5. If *cap.SpecialDaysSupported* is equal to true, ONVIF Client creates SpecialDayGroup (out *specialDayGroupToken*) by following the procedure mentioned in [Annex A.15](#).
- 6.6. ONVIF Client creates Schedule with Schedule token (out *scheduleToken*), with iCalendar value of the Schedule.Standard field (in *scheduleiCalendarValue*) and with SpecialDayGroupToken (in *specialDayGroupToken*) if Special Days is supported by the DUT by following the procedure mentioned in [Annex A.16](#).
7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
 - Filter.TopicExpression := "tns1:Configuration/AccessProfile/Changed"
8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
 - SubscriptionReference =: *s*
 - CurrentTime =: *ct*
 - TerminationTime =: *tt*
9. ONVIF client invokes **CreateAccessProfile** with parameters
 - AccessProfile.token := ""
 - AccessProfile.Name := "Test Access Profile"
 - AccessProfile.Description := "Test Description"
 - AccessProfile.AccessPolicy is skipped if *accessPointInfoCompleteList* is empty
 - AccessProfile.AccessPolicy[0].ScheduleToken := *scheduleToken*
 - AccessProfile.AccessPolicy[0].Entity := *accessPointInfoCompleteList*[0].token
 - AccessProfile.AccessPolicy[0].EntityType skipped
 - AccessProfile.AccessPolicy[0].Extension skipped

- AccessProfile.Extension skipped
10. The DUT responds with **CreateAccessProfileResponse** message with parameters
 - Token =: *accessProfileToken*
 11. If *accessProfileCompleteList1* contains item with token equal to *accessProfileToken*, FAIL the test and go to Step 24.
 12. Until *timeout1* timeout expires, repeat the following steps:
 - 12.1. ONVIF Client waits for time $t := \min\{(tt-ct)/2, 1 \text{ second}\}$.
 - 12.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
 - Timeout := PT60S
 - MessageLimit := 1
 - 12.3. The DUT responds with **PullMessagesResponse** message with parameters
 - CurrentTime =: *ct*
 - TerminationTime =: *tt*
 - NotificationMessage =: *m*
 - 12.4. If *m* is not null and the TopicExpression item in *m* is not equal to "tns1:Configuration/AccessProfile/Changed", FAIL the test and go to the step 24.
 - 12.5. If *m* is not null and the AccessProfileToken source simple item in *m* is not equal to *accessProfileToken*, go to the step 24.
 - 12.6. If *m* is not null and the AccessProfileToken source simple item in *m* is equal to *accessProfileToken*, go to the step 14.
 13. If *timeout1* timeout expires for step 12 without Notification with AccessProfileToken source simple item equal to *accessProfileToken*, FAIL the test and go to the step 24.
 14. ONVIF Client retrieves an access profile (in *accessProfileToken*, out *accessProfileList*) by following the procedure mentioned in Annex A.8.
 15. If *accessProfileList* contains more or less than one *AccessProfile* item, FAIL the test and go to step 24.
 16. If *accessProfileList*[0] item does not have equal field values to values from step 9, FAIL the test and go Step 24.

17. ONVIF Client retrieves an access profile info (in *accessProfileToken*, out *accessProfileInfoList*) by following the procedure mentioned in [Annex A.9](#).
18. If *accessProfileInfoList* contains more or less than one *AccessProfile* item, FAIL the test and go to step [24](#).
19. If *accessProfileInfoList*[0] item does not have equal field values to values from step [9](#), FAIL the test and go Step [24](#).
20. ONVIF Client retrieves a complete list of access profiles (out *accessProfileInfoCompleteList*) by following the procedure mentioned in [Annex A.1](#).
21. If *accessProfileInfoCompleteList* does not have *accessProfileInfo*.*[token:= accessProfileToken]* item equal field values to values from step [9](#), FAIL the test and go Step [24](#).
22. ONVIF Client retrieves a complete list of access profiles (out *accessProfileCompleteList2*) by following the procedure mentioned in [Annex A.3](#).
23. If *accessProfileCompleteList2* does not have *accessProfile*.*[token:= accessProfileToken]* item equal field values to values from step [9](#), FAIL the test and go to Step [24](#).
24. ONVIF Client deletes the Access Profile (in *accessProfileToken*) by following the procedure mentioned in [Annex A.6](#) to restore DUT configuration.
25. If there was access profile deleted at Step [4](#), restore it (in *accessProfileToRestore*) by following the procedure mentioned in [Annex A.10](#) to restore DUT configuration.
26. ONVIF Client deletes the Schedule (in *scheduleToken*) by following the procedure mentioned in [Annex A.17](#) to restore DUT configuration.
27. If *SpecialDayGroup* was created at step [6.5](#), ONVIF Client deletes the *SpecialDayGroup* (in *specialDayGroupToken*) by following the procedure mentioned in [Annex A.18](#) to restore DUT configuration.
28. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.
29. The DUT responds with **UnsubscribeResponse** message.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- The DUT did not send **CreateAccessProfilesResponse** message.

- The DUT did not send **CreatePullpointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message.
- The DUT did not send **UnsubscribeResponse** message.

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

Note: ONVIF Client skips whole `AccessProfile.AccessPolicy` field at step 9, if `AccessProfile.AccessPolicy[0].ScheduleToken` and `AccessProfile.AccessPolicy[0].Entity` field are skipped.

Note: The following fields are compared at steps 15, 21:

- AccessProfile:
 - token
 - Name
 - Description
 - AccessPolicy
 - ScheduleToken
 - Entity
 - EntityType

Note: The following fields are compared at steps 17, 19:

- AccessProfileInfo:
 - token
 - Name
 - Description

4.3.6 MODIFY ACCESS PROFILE

Test Case ID: ACCESS_RULES-3-1-7

Specification Coverage: AccessProfileInfo (ONVIF Access Rules Service Specification), AccessProfile (ONVIF Access Rules Service Specification), ModifyAccessProfile command (ONVIF Access Rules Service Specification).

Feature Under Test: ModifyAccessProfile

WSDL Reference: accessrules.wsdl and event.wsdl

Test Purpose: To verify Modify Access Profile and to verify tns1:Configuration/AccessProfile/Changed event generation.

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

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves a complete list of access profiles (out *accessProfileCompleteList1*) by following the procedure mentioned in [Annex A.3](#).
4. ONVIF Client checks free storage for additional Access Profile (in *accessProfileCompleteList1*, out *accessProfileToRestore*) by following the procedure mentioned in [Annex A.7](#).
5. ONVIF Client retrieves a complete list of schedules (out *scheduleCompleteList*) by following the procedure mentioned in [Annex A.4](#).
6. If Access Control service is supported by the DUT, ONVIF Client retrieves a complete list of access point information (out *accessPointInfoCompleteList*) by following the procedure mentioned in [Annex A.5](#).
7. ONVIF client creates access profile (in *scheduleCompleteList*, *accessPointInfoCompleteList*, out *accessProfileToken*) by following the procedure mentioned in [Annex A.11](#).
8. ONVIF Client invokes **CreatePullPointSubscription** with parameters
 - Filter.TopicExpression := "tns1:Configuration/AccessProfile/Changed"
9. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
 - SubscriptionReference =: s
 - CurrentTime =: ct

- TerminationTime =: *tt*

10. ONVIF client invokes **ModifyAccessProfile** with parameters

- AccessProfile.token := *accessProfileToken*
- AccessProfile.Name := "Test Access Profile 2"
- AccessProfile.Description := "Test Description 2"
- AccessProfile.AccessPolicy is skipped if *scheduleCompleteList* or *accessPointInfoCompleteList* is empty
- AccessProfile.AccessPolicy[0].ScheduleToken := *scheduleCompleteList*[0].token
- AccessProfile.AccessPolicy[0].Entity := *accessPointInfoCompleteList*[0].token
- AccessProfile.AccessPolicy[0].EntityType skipped
- AccessProfile.AccessPolicy[0].Extension skipped
- AccessProfile.Extension skipped

11. The DUT responds with empty **ModifyAccessProfileResponse** message.

12. Until *timeout1* timeout expires, repeat the following steps:

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

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

- Timeout := PT60S
- MessageLimit := 1

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

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

12.4. If *m* is not null and the TopicExpression item in *m* is not equal to "tns1:Configuration/AccessProfile/Changed", FAIL the test and go to the step 17.

12.5. If *m* is not null and the AccessProfileToken source simple item in *m* is not equal to *accessProfileToken*, go to the step 17.

- 12.6. If *m* is not null and the *AccessProfileToken* source simple item in *m* is equal to *accessProfileToken*, go to the step 17.
13. If *timeout1* timeout expires for step 12 without Notification with *AccessProfileToken* source simple item equal to *accessProfileToken*, FAIL the test and go to the step 17.
14. ONVIF Client retrieves an access profile (in *accessProfileToken*, out *accessProfileList*) by following the procedure mentioned in Annex A.8.
15. If *accessProfileList* contains more or less than one *AccessProfile* item, FAIL the test and go to step 17.
16. If *accessProfileList*[0] item does not have equal field values to values from step 10, FAIL the test and go Step 17.
17. ONVIF Client deletes the Access Profile (in *accessProfileToken*) by following the procedure mentioned in Annex A.6 to restore DUT configuration.
18. If there was access profile deleted at Step 4, restore it (in *accessProfileToRestore*) by following the procedure mentioned in Annex A.10 to restore DUT configuration.
19. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.
20. The DUT responds with **UnsubscribeResponse** message.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- The DUT did not send **ModifyAccessProfilesResponse** message.
- The DUT did not send **CreatePullpointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message.
- The DUT did not send **UnsubscribeResponse** message.

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

Note: The following fields are compared at step 16:

- AccessProfile:
 - token

- Name
- Description
- AccessPolicy
 - ScheduleToken
- Entity
- EntityType (skipped and "tac:AccessPointInfo" means the same, where tac is the namespace of the access control service: "http://www.onvif.org/ver10/accesscontrol/wsd")

4.3.7 DELETE ACCESS PROFILE

Test Case ID: ACCESS_RULES-3-1-8

Specification Coverage: AccessProfileInfo (ONVIF Access Rules Service Specification), AccessProfile (ONVIF Access Rules Service Specification), DeleteAccessProfile command (ONVIF Access Rules Service Specification).

Feature Under Test: DeleteAccessProfile

WSDL Reference: accessrules.wsd and event.wsd

Test Purpose: To verify Delete Access Profile and to verify tns1:Configuration/AccessProfile/Removed event generation.

Pre-Requirement: Access Rules Service is received from the DUT. Schedule Service is received from the DUT. Event Service is received from the DUT. The DUT shall have enough free storage capacity for one additional Access Profile.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves a complete list of access profiles (out *accessProfileCompleteList1*) by following the procedure mentioned in [Annex A.3](#).
4. ONVIF Client checks free storage for additional Access Profile (in *accessProfileCompleteList1*, out *accessProfileToRestore*) by following the procedure mentioned in [Annex A.7](#).

5. ONVIF Client retrieves a complete list of schedules (out *scheduleCompleteList*) by following the procedure mentioned in [Annex A.4](#).
6. If Access Control service is supported by the DUT, ONVIF Client retrieves a complete list of access point information (out *accessPointInfoCompleteList*) by following the procedure mentioned in [Annex A.5](#).
7. ONVIF client creates access profile (in *scheduleCompleteList*, *accessPointInfoCompleteList*, out *accessProfileToken*) by following the procedure mentioned in [Annex A.11](#).
8. ONVIF Client invokes **CreatePullPointSubscription** with parameters
 - Filter.TopicExpression := "tns1:Configuration/AccessProfile/Removed"
9. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
 - SubscriptionReference =: *s*
 - CurrentTime =: *ct*
 - TerminationTime =: *tt*
10. ONVIF client invokes **DeleteAccessProfile** with parameters
 - Token =: *accessProfileToken*
11. The DUT responds with empty **DeleteAccessProfileResponse** message.
12. Until *timeout1* timeout expires, repeat the following steps:
 - 12.1. ONVIF Client waits for time $t := \min\{(tt-ct)/2, 1 \text{ second}\}$.
 - 12.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
 - Timeout := PT60S
 - MessageLimit := 1
 - 12.3. The DUT responds with **PullMessagesResponse** message with parameters
 - CurrentTime =: *ct*
 - TerminationTime =: *tt*
 - NotificationMessage =: *m*

- 12.4. If *m* is not null and the TopicExpression item in *m* is not equal to "tns1:Configuration/AccessProfile/Removed", FAIL the test and go to the step 25.
- 12.5. If *m* is not null and the AccessProfileToken source simple item in *m* is not equal to *accessProfileToken*, go to the step 23.
- 12.6. If *m* is not null and the AccessProfileToken source simple item in *m* is equal to *accessProfileToken*, go to the step 14.
13. If *timeout1* timeout expires for step 12 without Notification with AccessProfileToken source simple item equal to *accessProfileToken*, FAIL the test and go to the step 23.
14. ONVIF Client retrieves an access profile (in *accessProfileToken*, out *accessProfileList*) by following the procedure mentioned in Annex A.8.
15. If *accessProfileList* is not empty, FAIL the test and go Step 23.
16. ONVIF Client retrieves an access profile info (in *accessProfileToken*, out *accessProfileInfo*) by following the procedure mentioned in Annex A.9.
17. If *accessProfileInfo* is not empty, FAIL the test and go Step 23.
18. ONVIF Client retrieves a complete list of access profiles info (out *accessProfileInfoCompleteList*) by following the procedure mentioned in Annex A.1.
19. If *accessProfileInfoCompleteList* have *accessProfileInfo* item with token equal to *accessProfileToken*, FAIL the test and go step 23.
20. ONVIF Client retrieves a complete list of access profiles (out *accessProfileCompleteList2*) by following the procedure mentioned in Annex A.3.
21. If there was access profile deleted at step 4:
 - *accessProfileCompleteList1* := *accessProfileCompleteList1* without *accessProfileToRestore*
22. If *accessProfileCompleteList2* is not equal to *accessProfileCompleteList1*, FAIL the test and go to step 23.
23. If there was access profile deleted at Step 4, restore it (in *accessProfileToRestore*) by following the procedure mentioned in Annex A.10 to restore DUT configuration.
24. ONVIF Client sends an **Unsubscribe** to the subscription endpoint s.
25. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **DeleteAccessProfilesResponse** message.
- The DUT did not send **CreatePullpointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message.
- The DUT did not send **UnsubscribeResponse** message.

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

4.3.8 GET ACCESS PROFILES WITH INVALID TOKEN

Test Case ID: ACCESS_RULES-3-1-9

Specification Coverage: AccessProfileInfo (ONVIF Access Rules Service Specification), AccessProfile (ONVIF Access Rules Service Specification), GetAccessProfiles command (ONVIF Access Rules Service Specification)

Feature Under Test: GetAccessProfiles

WSDL Reference: accessrules.wsdl

Test Purpose: To verify Get Access Profiles with invalid Token.

Pre-Requirement: Access Rules 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 retrieves a complete list of access profiles (out *accessProfileInfoCompleteList*) by following the procedure mentioned in [Annex A.1](#).
4. Set the following:
 - *invalidToken* := value not equal to any *accessProfileInfoCompleteList.token*
5. ONVIF client invokes **GetAccessProfiles** with parameters
 - Token list := *invalidToken*

6. The DUT responds with **GetAccessProfilesResponse** message with parameters
 - AccessProfile list =: *accessProfilesList*
7. If *accessProfilesList* is not empty, FAIL the test.
8. If *accessProfileInfoCompleteList* is empty, skip other steps.
9. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in [Annex A.2](#).
10. If *cap.MaxLimit* is less than 2, skip other steps.
11. ONVIF client invokes **GetAccessProfiles** with parameters
 - Token[0]:= *invalidToken*
 - Token[1]:= *accessProfileInfoCompleteList[0].token*
12. The DUT responds with **GetAccessProfilesResponse** message with parameters
 - AccessProfileInfo list =: *accessProfilesList*
13. If *accessProfilesList* is empty, FAIL the test.
14. If *accessProfilesList* contains more than one item, FAIL the test.
15. If *accessProfilesList[0].token* does not equal to *accessProfileInfoCompleteList[0].token*, FAIL the test.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

4.3.9 GET ACCESS PROFILES - TOO MANY ITEMS

Test Case ID: ACCESS_RULES-3-1-10

Specification Coverage: AccessProfile (ONVIF Access Rules Service Specification), GetAccessProfiles command (ONVIF Access Rules Service Specification)

Feature Under Test: GetAccessProfiles

WSDL Reference: accessrules.wsdl

Test Purpose: To verify Get Access Profiles in case if there a more items than MaxLimit in request.

Pre-Requisite: Access Rules 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 retrieves a complete list of access profiles (out *accessProfileCompleteList*) by following the procedure mentioned in [Annex A.3](#).
4. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in [Annex A.2](#).
5. If *accessProfileCompleteList.token* items number is less than *cap.MaxLimit* or equal to *cap.MaxLimit*, skip other steps.
6. Set the following:
 - *tokenList* := [subset of *accessProfileInfoCompleteList.token* values with items number equal to *cap.MaxLimit* + 1]
7. ONVIF client invokes **GetAccessProfiles** with parameters
 - Token list := *tokenList*
8. The DUT returns **env:Sender\ter:InvalidArgs\ter:TooManyItems** SOAP 1.2 fault.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

4.3.10 CREATE ACCESS PROFILE - NOT EMPTY ACCESS PROFILE TOKEN

Test Case ID: ACCESS_RULES-3-1-11

Specification Coverage: AccessProfile (ONVIF Access Rules Service Specification), CreateAccessProfile command (ONVIF Access Rules Service Specification).

Feature Under Test: CreateAccessProfile

WSDL Reference: accessrules.wsdl

Test Purpose: To verify Create Access Profiles with not empty token.

Pre-Requisite: Access Rules Service is received from the DUT. The DUT shall have enough free storage capacity for one additional Access Profile.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF client invokes **CreateAccessProfile** with parameters
 - AccessProfile.token := "AccessProfileToken"
 - AccessProfile.Name := "Test Access Profile"
 - AccessProfile.Description := "Test Description"
 - AccessProfile.AccessPolicy is skipped
4. The DUT returns **env:Sender\ter:InvalidArgs** SOAP 1.2 fault.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

4.3.11 CREATE ACCESS PROFILE - MULTIPLE SCHEDULES NOT SUPPORTED

Test Case ID: ACCESS_RULES-3-1-12

Specification Coverage: AccessProfileInfo (ONVIF Access Rules Service Specification), AccessProfile (ONVIF Access Rules Service Specification), CreateAccessProfile command (ONVIF Access Rules Service Specification).

Feature Under Test: CreateAccessProfile

WSDL Reference: accessrules.wsdl

Test Purpose: To verify Create Access Profiles with several access policies specifying different schedules for the same access point when MultipleSchedulesPerAccessPointSupported is not Supported.

Pre-Requirement: Access Rules Service is received from the DUT. Schedule Service is received from the DUT. The DUT shall have enough free storage capacity for one additional Access Profile.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in [Annex A.2](#).
4. If *cap.MultipleSchedulesPerAccessPointSupported* is equal to true, skip other steps.
5. ONVIF Client retrieves a complete list of schedules (out *scheduleCompleteList*) by following the procedure mentioned in [Annex A.4](#).
6. If *scheduleCompleteList* contains less than two schedules, skip other steps.
7. If Access Control service is supported by the DUT, ONVIF Client retrieves a complete list of access point information (out *accessPointInfoCompleteList*) by following the procedure mentioned in [Annex A.5](#).
8. If *accessPointInfoCompleteList* is empty, skip other steps.
9. ONVIF client invokes **CreateAccessProfile** with parameters
 - AccessProfile.token := ""
 - AccessProfile.Name := "Test Access Profile"
 - AccessProfile.Description := "Test Description"
 - AccessProfile.AccessPolicy[0].ScheduleToken := *scheduleCompleteList*[0].token

- AccessProfile.AccessPolicy[0].Entity := *accessPointInfoCompleteList*[0].token
- AccessProfile.AccessPolicy[0].EntityType skipped
- AccessProfile.AccessPolicy[0].Extension skipped
- AccessProfile.AccessPolicy[1].ScheduleToken := *scheduleCompleteList*[1].token
- AccessProfile.AccessPolicy[1].Entity := *accessPointInfoCompleteList*[0].token
- AccessProfile.AccessPolicy[1].EntityType skipped
- AccessProfile.AccessPolicy[1].Extension skipped
- AccessProfile.Extension skipped

10. The DUT returns **env:Sender\ter:CapabilityViolated \ter:MultipleSchedulesPerAccessPointSupported** SOAP 1.2 fault.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- The DUT did not send **env:Sender\ter:CapabilityViolated \ter:MultipleSchedulesPerAccessPointSupported** SOAP 1.2 fault

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

4.3.12 MODIFY ACCESS PROFILE WITH INVALID TOKEN

Test Case ID: ACCESS_RULES-3-1-13

Specification Coverage: ModifyAccessProfile command (ONVIF Access Rules Service Specification).

Feature Under Test: ModifyAccessProfile

WSDL Reference: accessrules.wsdl

Test Purpose: To verify Modify Access Profile with invalid Access Profile token.

Pre-Requisite: Access Rules 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 retrieves a complete list of access profile info (out *accessProfileInfoCompleteList*) by following the procedure mentioned in [Annex A.1](#).
4. Set the following:
 - *invalidToken* := value not equal to any *accessProfileInfoCompleteList.token*
5. ONVIF client invokes **ModifyAccessProfile** with parameters
 - *AccessProfile.token* := *invalidToken*
 - *AccessProfile.Name* := "Test Access Profile"
 - *AccessProfile.Description* := "Test Description"
 - *AccessProfile.AccessPolicy* is skipped
6. The DUT returns **env:SenderTerminates:InvalidArgument:NotFound** SOAP 1.2 fault.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT did not send **env:SenderTerminates:InvalidArgument: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.

4.3.13 DELETE ACCESS PROFILE WITH INVALID TOKEN

Test Case ID: ACCESS_RULES-3-1-14

Specification Coverage: DeleteAccessProfile command (ONVIF Access Rules Service Specification).

Feature Under Test: DeleteAccessProfile

WSDL Reference: accessrules.wsdl

Test Purpose: To verify Delete Access Profile with invalid Access Profile token.

Pre-Requisite: Access Rules 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 retrieves a complete list of access profile info (out *accessProfileInfoCompleteList*) by following the procedure mentioned in [Annex A.1](#).
4. Set the following:
 - *invalidToken* := value not equal to any *accessProfileInfoCompleteList.token*
5. ONVIF client invokes **DeleteAccessProfile** with parameters
 - Token =: *accessProfileToken*
6. The DUT returns **env:SenderTerminates:InvalidArgument:NotFound** SOAP 1.2 fault.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **env:SenderTerminates:InvalidArgument: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.

4.3.14 SET NEW ACCESS PROFILE

Test Case ID: ACCESS_RULES-3-1-15

Specification Coverage: AccessProfileInfo (ONVIF Access Rules Service Specification), AccessProfile (ONVIF Access Rules Service Specification), SetAccessProfile command (ONVIF Access Rules Service Specification).

Feature Under Test: SetAccessProfile, tns1:Configuration/AccessProfile/Changed

WSDL Reference: accessrules.wsdl and event.wsdl

Test Purpose: To verify Create Access Profiles using SetAccessProfile command and to verify tns1:Configuration/AccessProfile/Changed event generation.

Pre-Requisite: Access Rules Service is received from the DUT. Schedule Service is received from the DUT. Event Service is received from the DUT. Client Supplied Token is supported by the DUT. The DUT shall have enough free storage capacity for one additional Access Profile.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves complete list of access profiles by following the procedure mentioned in [Annex A.3](#) with the following input and output parameters
 - out *initialAccessProfileCompleteList* - access profile complete list
4. ONVIF Client checks free storage for additional access profile by following the procedure mentioned in [Annex A.7](#) with the following input and output parameters
 - in *initialAccessProfileCompleteList* - access profile complete list
 - out *accessProfileToRestore* - removed access profile
5. If Access Control service is supported by the DUT:
 - 5.1. ONVIF Client retrieves complete list of access point information by following the procedure mentioned in [Annex A.5](#) with the following input and output parameters
 - out *accessPointInfoCompleteList* - access point info complete list
6. If *accessPointInfoCompleteList* is not empty:
 - 6.1. ONVIF Client retrieves complete list of schedules by following the procedure mentioned in [Annex A.4](#) with the following input and output parameters
 - out *initialScheduleCompleteList* - schedule complete list
 - 6.2. If *scheduleCompleteList* is not empty:
 - 6.2.1. Set *scheduleToken* := *scheduleCompleteList*[0].token
 - 6.2.2. Go to step 7.
 - 6.3. ONVIF Client retrieves Schedule Service Capabilities by following the procedure mentioned in [Annex A.12](#) with the following input and output parameters

- out *cap* - Schedule Service capabilities
- 6.4. ONVIF Client generates appropriate iCalendar value for the Schedule.Standard field by following the procedure mentioned in [Annex A.14](#) with the following input and output parameters
- in *cap* - Schedule Service capabilities
 - out *scheduleiCalendarValue* - iCalendar value for the Schedule.Standard field
- 6.5. If *cap.SpecialDaysSupported* = true:
- 6.5.1. ONVIF Client creates special day group by following the procedure mentioned in [Annex A.15](#) with the following input and output parameters
- out *specialDayGroupToken* - token of new special day group
- 6.6. ONVIF Client creates schedule by following the procedure mentioned in [Annex A.16](#) with the following input and output parameters
- out *scheduleToken* - token of new schedule
 - in *scheduleiCalendarValue* - iCalendar value for the Schedule.Standard field
 - in *specialDayGroupToken* - special day group token
7. Set *accessProfileToken* := token that differs from tokens listed in *initialAccessProfileCompleteList*.
8. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in [Annex A.19](#) with the following input and output parameters
- in **tns1:Configuration/AccessProfile/Changed** - Notification Topic
 - out *s* - Subscription reference
 - out *currentTime* - current time for the DUT
 - out *terminationTime* - Subscription termination time
9. Set *accessProfile* :=
- *AccessProfile.token* := *accessProfileToken*
 - *AccessProfile.Name* := "Test Access Profile"
 - *AccessProfile.Description* := "Test Description"

- If *accessPointInfoCompleteList* contains at least one item:
 - *AccessProfile.AccessPolicy[0].ScheduleToken* := *scheduleToken*
 - *AccessProfile.AccessPolicy[0].Entity* := *accessPointInfoCompleteList[0].token*
 - *AccessProfile.AccessPolicy[0].EntityType* is skipped
 - *AccessProfile.AccessPolicy[0].Extension* is skippedotherwise:
 - *AccessProfile.AccessPolicy* is skipped
 - *AccessProfile.Extension* is skipped
10. ONVIF client invokes **SetAccessProfile** request with parameters
- *AccessProfile* := *accessProfile*
11. The DUT responds with **SetAccessProfileResponse** message.
12. ONVIF Client retrieves and checks **tns1:Configuration/AccessProfile/Changed** event for the specified Schedule 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 *accessProfileToken* - access profile token
13. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
- in *s* - Subscription reference
14. ONVIF Client retrieves a access profile by following the procedure mentioned in [Annex A.8](#) with the following input and output parameters
- in *accessProfileToken* - access profile token
 - out *accessProfileList* - the list of access profiles
15. If *accessProfileList[0]* item does not have equal to *accessProfile*, FAIL the test, restore the DUT state, and skip other steps.

16. ONVIF Client retrieves a access profile info by following the procedure mentioned in [Annex A.9](#) with the following input and output parameters
 - in *accessProfileToken* - access profile token
 - out *accessProfileInfoList* - the list of access profiles info
17. If *accessProfileInfoList*[0] item does not have equal fields with *accessProfile*, FAIL the test, restore the DUT state, and skip other steps.
18. ONVIF Client retrieves complete list of access profiles info by following the procedure mentioned in [Annex A.1](#) with the following input and output parameters
 - out *accessProfileInfoCompleteList* - access profile info complete list
19. If *accessProfileInfoCompleteList* does not have *AccessProfileInfo*[token = *accessProfileToken*] item with equal fields with *accessProfile*, FAIL the test, restore the DUT state, and skip other steps.
20. ONVIF Client retrieves complete list of access profiles by following the procedure mentioned in [Annex A.3](#) with the following input and output parameters
 - out *accessProfileCompleteList* - access profile complete list
21. If *accessProfileCompleteList* does not have *AccessProfile*[token = *accessProfileToken*] item with equal fields with *accessProfile*, FAIL the test, restore the DUT state, and skip other steps.
22. ONVIF Client deletes the Access Profile by following the procedure mentioned in [Annex A.6](#) with the following input and output parameters
 - in *accessProfileToken* - access profile token
23. If there was access profile deleted at step 4:
 - 23.1. ONVIF Client restores access profile by following the procedure mentioned in [Annex A.10](#) with the following input and output parameters
 - in *accessProfileToRestore* - removed access profile
24. ONVIF Client deletes the Schedule by following the procedure mentioned in [Annex A.17](#) with the following input and output parameters
 - in *scheduleToken* - schedule token
25. If *cap.SpecialDaysSupported* = true:

25.1. ONVIF Client deletes special day group by following the procedure mentioned in [Annex A.18](#) with the following input and output parameters

- in *specialDayGroupToken* - special day group token

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **SetAccessProfilesResponse** message.

Note: The following fields are compared at steps [15](#) and [21](#):

- AccessProfile:
 - token
 - Name
 - Description
 - AccessPolicy
 - ScheduleToken
 - Entity
 - EntityType

Note: The following fields are compared at steps [17](#) and [19](#):

- AccessProfileInfo:
 - token
 - Name
 - Description

4.3.15 SET ACCESS PROFILE

Test Case ID: ACCESS_RULES-3-1-16

Specification Coverage: AccessProfileInfo (ONVIF Access Rules Service Specification), AccessProfile (ONVIF Access Rules Service Specification), SetAccessProfile command (ONVIF Access Rules Service Specification).

Feature Under Test: SetAccessProfile

WSDL Reference: accessrules.wsdl and event.wsdl

Test Purpose: To verify access profile modification using SetAccessProfile command and to verify tns1:Configuration/AccessProfile/Changed event generation.

Pre-Requisite: Access Rules Service is received from the DUT. Schedule Service is received from the DUT. Event Service is received from the DUT. Client Supplied Token is supported by the DUT. The DUT shall have enough free storage capacity for one additional Access Profile.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves complete list of access profiles by following the procedure mentioned in [Annex A.3](#) with the following input and output parameters
 - out *initialAccessProfileCompleteList* - access profile complete list
4. ONVIF Client checks free storage for additional access profile by following the procedure mentioned in [Annex A.7](#) with the following input and output parameters
 - in *initialAccessProfileCompleteList* - access profile complete list
 - out *accessProfileToRestore* - removed access profile
5. ONVIF Client retrieves complete list of schedules by following the procedure mentioned in [Annex A.4](#) with the following input and output parameters
 - out *initialScheduleCompleteList* - schedule complete list
6. If Access Control service is supported by the DUT:
 - 6.1. ONVIF Client retrieves complete list of access point information by following the procedure mentioned in [Annex A.5](#) with the following input and output parameters
 - out *accessPointInfoCompleteList* - access point info complete list

7. ONVIF Client creates access profile by following the procedure mentioned in [Annex A.11](#) with the following input and output parameters
 - in *scheduleCompleteList* - schedule complete list
 - in *accessPointInfoCompleteList* - access point info complete list
 - out *accessProfileToken* - token of new access profile
8. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in [Annex A.19](#) with the following input and output parameters
 - in **tns1:Configuration/AccessProfile/Changed** - Notification Topic
 - out *s* - Subscription reference
 - out *currentTime* - current time for the DUT
 - out *terminationTime* - Subscription termination time
9. Set *accessProfile* :=
 - *AccessProfile.token* := *accessProfileToken*
 - *AccessProfile.Name* := "Test Access Profile 2"
 - *AccessProfile.Description* := "Test Description 2"
 - If *accessPointInfoCompleteList* contains at least one item:
 - *AccessProfile.AccessPolicy[0].ScheduleToken* := *scheduleToken*
 - *AccessProfile.AccessPolicy[0].Entity* := *accessPointInfoCompleteList[0].token*
 - *AccessProfile.AccessPolicy[0].EntityType* is skipped
 - *AccessProfile.AccessPolicy[0].Extension* is skipped
 - otherwise:
 - *AccessProfile.AccessPolicy* is skipped
 - *AccessProfile.Extension* is skipped
10. ONVIF client invokes **SetAccessProfile** request with parameters
 - *AccessProfile* := *accessProfile*
11. The DUT responds with **SetAccessProfileResponse** message.

12. ONVIF Client retrieves and checks **tns1:Configuration/AccessProfile/Changed** event for the specified Schedule 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 *accessProfileToken* - access profile token
13. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in *s* - Subscription reference
14. ONVIF Client retrieves a access profile by following the procedure mentioned in [Annex A.8](#) with the following input and output parameters
 - in *accessProfileToken* - access profile token
 - out *accessProfileList* - the list of access profiles
15. If *accessProfileList[0]* item does not have equal to *accessProfile*, FAIL the test, restore the DUT state, and skip other steps.
16. ONVIF Client retrieves a access profile info by following the procedure mentioned in [Annex A.9](#) with the following input and output parameters
 - in *accessProfileToken* - access profile token
 - out *accessProfileInfoList* - the list of access profiles info
17. If *accessProfileInfoList[0]* item does not have equal fields with *accessProfile*, FAIL the test, restore the DUT state, and skip other steps.
18. ONVIF Client retrieves complete list of access profiles info by following the procedure mentioned in [Annex A.1](#) with the following input and output parameters
 - out *accessProfileInfoCompleteList* - access profile info complete list
19. If *accessProfileInfoCompleteList* does not have *AccessProfileInfo[token = accessProfileToken]* item with equal fields with *accessProfile*, FAIL the test, restore the DUT state, and skip other steps.
20. ONVIF Client retrieves complete list of access profiles by following the procedure mentioned in [Annex A.3](#) with the following input and output parameters

- out *accessProfileCompleteList* - access profile complete list
21. If *accessProfileCompleteList* does not have *AccessProfile[token = accessProfileToken]* item with equal fields with *accessProfile*, FAIL the test, restore the DUT state, and skip other steps.
22. ONVIF Client deletes the Access Profile by following the procedure mentioned in [Annex A.6](#) with the following input and output parameters
- in *accessProfileToken* - access profile token
23. If there was access profile deleted at step 4:
- 23.1. ONVIF Client restores access profile by following the procedure mentioned in [Annex A.10](#) with the following input and output parameters
- in *accessProfileToRestore* - removed access profile

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- The DUT did not send **SetAccessProfilesResponse** message.

Note: The following fields are compared at steps 15 and 21:

- AccessProfile:
 - token
 - Name
 - Description
 - AccessPolicy
 - ScheduleToken
 - Entity
 - EntityType

Note: The following fields are compared at steps 17 and 19:

- AccessProfileInfo:
 - token
 - Name
 - Description

4.4 Events

4.4.1 ACCESS PROFILE CHANGED EVENT

Test Case ID: ACCESS_RULES-5-1-1

Specification Coverage: Event Changed (ONVIF Access Rules Service Specification)

Feature Under Test: GetEventProperties

WSDL Reference: accessrules.wsdl and event.wsdl

Test Purpose: To verify tns1:Configuration/AccessProfile/Changed event format.

Pre-Requisite: Access Rules Service is supported by the DUT. Event 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 **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/AccessProfile/Changed topic, FAIL the test and skip other steps.
 6. ONVIF Client verifies tns1:Configuration/AccessProfile/Changed topic (*ChangedTopic*) from *topicSet*:
 - 6.1. If *ChangedTopic.MessageDescription.IsProperty* equals to true, FAIL the test and skip other steps.
 - 6.2. If *ChangedTopic* does not contain MessageDescription.Source.SimpleItemDescription item with Name = "AccessProfileToken", FAIL the test and skip other steps.
 - 6.3. If *ChangedTopic.MessageDescription.Source.SimpleItemDescription* with Name = "AccessProfileToken" does not have Type = "pt:ReferenceToken", FAIL the test and skip other steps.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

4.4.2 ACCESS PROFILE REMOVED EVENT

Test Case ID: ACCESS_RULES-5-1-2

Specification Coverage: Event Changed (ONVIF Access Rules Service Specification)

Feature Under Test: GetEventProperties

WSDL Reference: accessrules.wsdl and event.wsdl

Test Purpose: To verify tns1:Configuration/AccessProfile/Removed event format.

Pre-Requisite: Access Rules Service is supported by the DUT. Event 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 **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/AccessProfile/Removed topic, FAIL the test and skip other steps.
6. ONVIF Client verifies tns1:Configuration/AccessProfile/Removed topic (*RemovedTopic*) from *topicSet*:
 - 6.1. If *RemovedTopic*.MessageDescription.IsProperty equals to true, FAIL the test and skip other steps.
 - 6.2. If *RemovedTopic* does not contain MessageDescription.Source.SimpleItemDescription item with Name = "AccessProfileToken", FAIL the test and skip other steps.
 - 6.3. If *RemovedTopic*.MessageDescription.Source.SimpleItemDescription with Name = "AccessProfileToken" does not have Type = "pt:ReferenceToken", FAIL the test and skip other steps.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

4.5 Consistency

4.5.1 ACCESS POLICIES AND ACCESS POINT CONSISTENCY

Test Case ID: ACCESS_RULES-6-1-1

Specification Coverage: AccessPolicy (ONVIF Access Rules Service Specification), AccessPoint (ONVIF Access Control Service Specification) ServiceCapabilities (ONVIF Access Rules Service Specification), GetServiceCapabilities command (ONVIF Access Rules Service Specification)

Feature Under Test: GetAccessProfileList, GetAccessPointInfoList

WSDL Reference: accessrules.wsdl and accesscontrol.wsdl

Test Purpose: To verify that Access Policies List contains only Access Points from Access Point Info List.

Pre-Requirement: Access Rules Service is received from the DUT. Access Control 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 retrieves a complete list of access profiles (out *accessProfileCompleteList*) by following the procedure mentioned in [Annex A.3](#).
4. ONVIF Client retrieves a complete list of access point information (out *accessPointInfoCompleteList*) by following the procedure mentioned in [Annex A.5](#).
5. For each access profile (*accessProfile*) from *accessProfileCompleteList* do the following:
 - 5.1. For each *accessProfile.AccessPolicy* with skipped *EntityType* or *EntityType* equal to *tac:AccessPoint* repeat the following steps:
 - 5.1.1. If *accessPointInfoCompleteList* does not contain Access Point token equal to *accessProfile.AccessPolicy.EntityType* item, FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

Annex A Helper Procedures and Additional Notes

A.1 Get access profiles information list

Name: HelperGetAccessProfileInfoList

Procedure Purpose: Helper procedure to get complete access profiles information list.

Pre-requisite: Access Rules Service is received from the DUT.

Input: None.

Returns: The complete list of access profiles information (*accessProfileInfoCompleteList*).

Procedure:

1. ONVIF client invokes **GetAccessProfileInfoList** with parameters
 - Limit skipped
 - StartReference skipped
2. The DUT responds with **GetAccessProfileInfoListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - AccessProfileInfo list =: *accessProfileInfoCompleteList*
3. Until *nextStartReference* is not null, repeat the following steps:
 - 3.1. ONVIF client invokes **GetAccessProfileInfoList** with parameters
 - Limit skipped
 - StartReference := *nextStartReference*
 - 3.2. The DUT responds with **GetAccessProfileInfoListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - AccessProfileInfo list =: *accessProfileInfoListPart*
 - 3.3. Set the following:
 - *accessProfileInfoCompleteList* := *accessProfileInfoCompleteList* + *accessProfileInfoListPart*

Procedure Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

A.2 Get service capabilities

Name: HelperGetServiceCapabilities

Procedure Purpose: Helper procedure to get service capabilities.

Pre-requisite: Access Rules 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.3 Get access profiles list

Name: HelperGetAccessProfilesList

Procedure Purpose: Helper procedure to get complete access profiles list with.

Pre-requisite: Access Rules Service is received from the DUT.

Input: None.

Returns: The complete list of access profiles (*accessProfileCompleteList*).

Procedure:

1. ONVIF client invokes **GetAccessProfileList** with parameters
 - Limit skipped
 - StartReference skipped
2. The DUT responds with **GetAccessProfileListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - AccessProfile list =: *accessProfileCompleteList*
3. Until *nextStartReference* is not null, repeat the following steps:
 - 3.1. ONVIF client invokes **GetAccessProfileList** with parameters
 - Limit skipped
 - StartReference := *nextStartReference*
 - 3.2. The DUT responds with **GetAccessProfileListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - AccessProfile list =: *accessProfileListPart*
 - 3.3. Set the following:
 - *accessProfileCompleteList* := *accessProfileCompleteList* + *accessProfileListPart*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.4 Get schedule list

Name: HelperGetScheduleList

Procedure Purpose: Helper procedure to get complete schedule list.

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

Input: None.

Returns: The complete list of schedules (*scheduleCompleteList*).

Procedure:

1. ONVIF Client invokes **GetScheduleList** request with parameters
 - Limit skipped
 - StartReference skipped
2. The DUT responds with **GetScheduleListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - Schedule list =: *scheduleCompleteList*
3. Until *nextStartReference* is not null, repeat the following steps:
 - 3.1. ONVIF client invokes **GetScheduleList** with parameters
 - Limit skipped
 - StartReference := *nextStartReference*
 - 3.2. The DUT responds with **GetScheduleListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - Schedule list =: *scheduleListPart*
 - 3.3. Set the following:
 - *scheduleCompleteList* := *scheduleCompleteList* + *scheduleListPart*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.5 Get access point information list

Name: HelperGetAccessPointInfoList

Procedure Purpose: Helper procedure to get complete access point information list.

Pre-requisite: Access Control Service is received from the DUT.

Input: None.

Returns: The complete list of access point information (*accessPointInfoCompleteList*).

Procedure:

1. ONVIF Client invokes **GetAccessPointInfoList** request with parameters
 - Limit skipped
 - StartReference skipped
2. The DUT responds with **GetAccessPointInfoListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - Schedule list =: *accessPointInfoCompleteList*
3. Until *nextStartReference* is not null, repeat the following steps:
 - 3.1. ONVIF client invokes **GetAccessPointInfoList** with parameters
 - Limit skipped
 - StartReference := *nextStartReference*
 - 3.2. The DUT responds with **GetAccessPointInfoListResponse** message with parameters
 - NextStartReference =: *nextStartReference*
 - AccessPointInfo list =: *accessPointInfoListPart*
 - 3.3. Set the following:
 - *accessPointInfoCompleteList* := *accessPointInfoCompleteList* + *accessPointInfoListPart*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.6 Delete access profile

Name: HelperDeleteAccessProfile

Procedure Purpose: Helper procedure to delete access profile.

Pre-requisite: Access Rules Service is received from the DUT.

Input: Access Profile Token (*accessProfileToken*).

Returns: None.

Procedure:

1. ONVIF Client invokes **DeleteAccessProfile** request.
 - Token =: *accessProfileToken*
2. The DUT responds with empty **DeleteAccessProfileResponse** message.

Procedure Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

A.7 Free storage for additional access profile

Name: HelperCheckFreeStorageForAccessProfile

Procedure Purpose: Helper procedure to check free storage for additional access profile.

Pre-requisite: Access Rules Service is received from the DUT.

Input: The complete list of access profiles (*accessProfileCompleteList*).

Returns: Access Profile to restore (*accessProfileToRestore*).

Procedure:

1. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in [Annex A.2](#).
2. ONVIF client compares *cap.MaxAccessProfiles* with number of items at *accessProfileCompleteList*.
3. If number of items at *accessProfileCompleteList* less than *cap.MaxAccessProfiles*, skip other steps.
4. If number of items at *accessProfileCompleteList* equal to *cap.MaxAccessProfiles*, execute the following steps:
 - 4.1. ONVIF client invokes **GetAccessProfiles** with parameters
 - Token list := *accessProfileCompleteList[0].token*
 - 4.2. The DUT responds with **GetAccessProfilesResponse** message with parameters
 - AccessProfile list := *accessProfileToRestore*
 - 4.3. ONVIF Client deletes the Access Profile (in *accessProfileCompleteList[0].token*) by following the procedure mentioned in [Annex A.6](#).

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- The number of items at *accessProfileCompleteList* more than *cap.MaxAccessProfiles*.

A.8 Get access profile

Name: HelperGetAccessProfile

Procedure Purpose: Helper procedure to get access profile.

Pre-requisite: Access Rules Service is received from the DUT.

Input: Access Profile Token (*accessProfileToken*).

Returns: Access Profile List (*accessProfileList*).

Procedure:

1. ONVIF Client invokes **GetAccessProfiles** request with parameters
 - Token[0] := *accessProfileToken*
2. The DUT responds with **GetAccessProfilesResponse** message with parameters
 - AccessProfile list =: *accessProfileList*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.9 Get access profile info

Name: HelperGetAccessProfileInfo

Procedure Purpose: Helper procedure to get access profile info.

Pre-requisite: Access Rules Service is received from the DUT.

Input: Access Profile Token (*accessProfileToken*).

Returns: Access Profile Info List (*accessProfileInfoList*).

Procedure:

1. ONVIF Client invokes **GetAccessProfileInfo** request with parameters
 - Token := *accessProfileToken*
2. The DUT responds with **GetAccessProfileInfoResponse** message with parameters
 - AccessProfileInfo =: *accessProfileInfoList*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.10 Restore access profile

Name: HelperRestoreAccessProfile

Procedure Purpose: Helper procedure to restore access profile.

Pre-requisite: Access Rules Service is received from the DUT.

Input: Access Profile (*accessProfile*).

Returns: None.

Procedure:

1. ONVIF Client invokes **CreateAccessProfile** request with parameters
 - AccessProfile := *accessProfile*
2. The DUT responds with **CreateAccessProfileResponse** message with parameters
 - Token =: *accessProfileToken*

Procedure Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

A.11 Create access profile

Name: HelperCreateAccessProfile

Procedure Purpose: Helper procedure to Create access profile.

Pre-requisite: Access Rules Service is received from the DUT.

Input: Complete list of schedules (*scheduleCompleteList*), complete list of access point info (*accessPointInfoCompleteList*).

Returns: Access Profile Token (*accessProfileToken*).

Procedure:

1. ONVIF client invokes **CreateAccessProfile** with parameters
 - *AccessProfile.token* := ""
 - *AccessProfile.Name* := "Test Access Profile"
 - *AccessProfile.Description* := "Test Description"
 - *AccessProfile.AccessPolicy* is skipped if *scheduleCompleteList* or *accessPointInfoCompleteList* is empty
 - *AccessProfile.AccessPolicy[0].ScheduleToken* := *scheduleCompleteList[0].token*
 - *AccessProfile.AccessPolicy[0].Entity* := *accessPointInfoCompleteList[0].token*
 - *AccessProfile.AccessPolicy[0].EntityType* skipped
 - *AccessProfile.AccessPolicy[0].Extension* skipped
 - *AccessProfile.Extension* skipped
2. The DUT responds with **CreateAccessProfileResponse** message with parameters
 - *Token* =: *accessProfileToken*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.12 Get service capabilities (Schedule)

Name: HelperGetServiceCapabilitiesSchedule

Procedure Purpose: Helper procedure to get service capabilities for Schedule Service.

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.13 Generate UID value for iCalendar

Name: HelperUIDiCalendarGeneration

Procedure Purpose: Helper procedure to generate Unique Identifier value for UID field in iCalendar.

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

Input: None.

Returns: Unique Identifier value for UID field in iCalendar (*uid*) that is compliant to [RFC 2445].

Procedure:

1. ONVIF Client generates Globally Unique Identifier value.

A.14 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: The Schedule service capabilities (*cap*).

Returns: iCalendar value for Standard field (*scheduleiCalendarValue*) that is compliant to [RFC 2445].

Procedure:

1. ONVIF Client generates Unique Identifier value for UID field in iCalendar (out *uid*) by following the procedure mentioned in [Annex A.13](#).
2. If *cap.ExtendedRecurrenceSupported* is equal to true, set the following:

- *scheduleiCalendarValue* := "BEGIN:VCALENDAR

BEGIN:VEVENT

SUMMARY:Access from 9 AM to 6 PM

DTSTART:<current year>< current month>< current day>T090000

DTEND:<year of (current day + one week)><month of (current day + one week)><day of (current day + one week)>T180000

RRULE:FREQ=DAILY

UID:*uid*

END:VEVENT

END:VCALENDAR"

3. If *cap.ExtendedRecurrenceSupported* is equal to false, set the following:

- *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.15 Create special day group

Name:HelperCreateSpecialDayGroup

Procedure Purpose: Helper procedure to create SpecialDayGroup with Days field value that is compliant to [RFC 2445].

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported. The DUT shall have enough free storage capacity for one additional SpecialDayGroup.

Input: iCalendar value of the SpecialDayGroup.Days field (*days*) that is compliant to [RFC 2445].

Returns: Returns: Special day group token (*specialDayGroupToken*), iCalendar value of Days field (*days*) that is compliant to [RFC 2445].

Procedure:

1. ONVIF Client generates Unique Identifier value for UID field in iCalendar (out *uid*) by following the procedure mentioned in [Annex A.13](#).
2. If *days* is empty, set the following:

- *days* := "BEGIN:VCALENDAR

BEGIN:VEVENT

SUMMARY:Test special days

DTSTART:<current year><current month><current day>T000000

DTEND:<year of the next day><month of the next day><day of the next day>T000000

UID:*uid*

END:VEVENT

END:VCALENDAR"

3. ONVIF client invokes **CreateSpecialDayGroup** with parameters
 - SpecialDayGroup.token := ""

- `SpecialDayGroup.Name` := "Test SpecialDayGroup Name"
 - `SpecialDayGroup.Description` := "Test SpecialDayGroup Description"
 - `SpecialDayGroup.Days` := *days*
4. The DUT responds with **CreateSpecialDayGroupResponse** message with parameters
- `Token` =: *specialDayGroupToken*

Procedure Result:**PASS –**

- The DUT passed all assertions.

FAIL –

- The DUT did not send **CreateSpecialDayGroupResponse** message.

A.16 Create Schedule

Name:HelperCreateSchedule

Procedure Purpose: Helper procedure to create Schedule with Standard field value that is compliant to [RFC 2445].

Pre-requisite: Schedule Service is received from the DUT. The DUT shall have enough free storage capacity for one additional Schedule.

Input: iCalendar value of the Schedule.Standard field (*scheduleiCalendarValue*), SpecialDayGroupToken (*specialDayGroupToken*).

Returns: Schedule token (*scheduleToken*).

Procedure:

1. ONVIF client invokes **CreateSchedule** with parameters
 - `Schedule.token` := ""
 - `Schedule.Description` := "Test Description"
 - `Schedule.Name` := "Test Name"
 - `Schedule.Standard` := *scheduleiCalendarValue*

- Schedule.SpecialDays skipped if *specialDayGroupToken* is empty
 - Schedule.SpecialDays.GroupToken := *specialDayGroupToken*
 - Schedule.SpecialDays.TimeRange.From := "22:00:00"
 - Schedule.SpecialDays.TimeRange.Until := "23:00:00"
2. 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.17 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.18 Delete special day group

Name: HelperDeleteSpecialDayGroup

Procedure Purpose: Helper procedure to delete SpecialDayGroup.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Input: SpecialDayGroup Token (*specialDayGroupToken*).

Returns: None.

Procedure:

1. ONVIF client invokes **DeleteSpecialDayGroup** with parameters
 - Token =: *specialDayGroupToken*
2. The DUT responds with empty **DeleteSpecialDayGroupResponse** message

Procedure Result:

PASS –

- The DUT passed all assertions.

FAIL –

- The DUT did not send **DeleteSpecialDayGroupResponse** message.

A.19 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.20 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.21 Retrieve Access Profile Changed Event by PullPoint

Name: HelperPullAccessProfileChanged

Procedure Purpose: Helper procedure to retrieve and check `tns1:Configuration/AccessProfile/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 access profile token (*accessProfileToken*).

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* request 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 `AccessProfileToken` source simple item in *notificationMessageList* is equal to *accessProfileToken*, skip other steps and finish the procedure.
 - 1.5. If *operationDelay* timeout expires for step 1 without Notification with `AccessProfileToken` source simple item equal to *accessProfileToken*, 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.