# ONVIF®

# Access Control Device Test Specification

Version 20.12

December, 2020

## REVISION HISTORY

| Vers. | Date | Description |
|---|---|---|
| 13.06 | Jun, 2013 | First issue |
| 13.12 | Dec, 2013 | Minor updates. |
| 17.12 | Aug 08, 2017 | Current document name was changed from Access Control Test Specification to Access Control Device Test Specification.<br><br>The document formatting was updated. |
| 18.06 | Jun 21, 2018 | Reformatting document using new template |
| 18.12 | Oct 05, 2018 | The following were updated in the scope of #1677:<br><br>Scope\Access Denied Events (tns1:AccessControl/Denied/CredentialNotFound added)<br><br>Test Overview\Prerequisites (updated with new test with user interaction)<br><br>Test Overview\Test Policy\Access Denied Events (tns1:AccessControl/Denied/CredentialNotFound added)<br><br>Annex A.1 Get Complete Access Point Info List (format updated)<br><br>Annex A.3 Create Pull Point Subscription (added)<br><br>Annex A.4 Delete Subscription (added)<br><br>Annex A.5 Retrieve CredentialNotFound Event by PullPoint (added)<br><br>ACCESSCONTROL-9-1-4 ACCESS CONTROL - ACCESS DENIED WITH CREDENTIAL EVENT (CREDENTIAL NOT FOUND) (added) |
| 18.12 | Dec 21, 2018 | Switching Hub description in 'Network Configuration for DUT' section was updated according to #1737 |
| 20.06 | Aug 15, 2019 | The following were updated in the scope of #1880:<br><br>Scope/Access Granted Events (was updated with tns1:AccessControl/AccessGranted/Identifier)<br><br>Test Policy/Access Granted Events (was updated with tns1:AccessControl/AccessGranted/Identifier)<br><br>ACCESSCONTROL-6-1-3 ACCESS CONTROL - ACCESS GRANTED TO IDENTIFIER EVENT (was added)<br><br>Annex A.6 Retrieve Access Granted Identifier Event by PullPoint (was added)<br><br>Prerequisites (was updated with ACCESSCONTROL-6-1-3 test with manual action) |
| 20.06 | Aug 15, 2019 | The following were updated in the scope of #1882:<br><br>Scope/Access Denied Events (was updated with tns1:AccessControl/Denied/Identifier)<br><br>Test Policy/Access Granted Events (was updated with tns1:AccessControl/Denied/Identifier) |

| | | |
|---|---|---|
| | | ACCESSCONTROL-9-1-5 ACCESS CONTROL - ACCESS DENIED TO IDENTIFIER EVENT (was added)<br><br>Annex A.7 Retrieve Denied Identifier Event by PullPoint (was added)<br><br>Prerequisites (was updated with ACCESSCONTROL-9-1-5 test with manual action) |
| 20.06 | Aug 15, 2019 | The following were updated in the scope of #1923:<br><br>ACCESSCONTROL-9-1-5 ACCESS CONTROL - ACCESS DENIED TO IDENTIFIER EVENT (pre-requisite and step was updated to include clarification from CR2633)<br><br>ACCESSCONTROL-6-1-3 ACCESS CONTROL - ACCESS GRANTED TO IDENTIFIER EVENT (pre-requisite and step was updated to include clarification from CR2633) |
| 20.06 | Jan 15, 2020 | The following were updated in the scope of #1878:<br><br>Scope\External Authorization (tns1:AccessControl/Request/Identifier were added)<br><br>Prerequisites (new test cases with manual action were added, new prerequisite for Identifier Access capability were added)<br><br>Test Policy\External Authorization (new test policies for Identifier External Authorization were added)<br><br>ACCESSCONTROL-11-1-9 ACCESS CONTROL - ACCESS GRANTED TO IDENTIFIER (EXTERNAL AUTHORIZATION) (new)<br><br>ACCESSCONTROL-11-1-10 ACCESS CONTROL - ACCESS DENIED TO IDENTIFIER (EXTERNAL AUTHORIZATION) (new)<br><br>ACCESSCONTROL-11-1-11 ACCESS CONTROL - ACCESS TIMEOUT TO IDENTIFIER (EXTERNAL AUTHORIZATION) (new)<br><br>Annex A.8 Retrieve Request Identifier Event by PullPoint (new)<br><br>Annex A.9 Retrieve Request Timeout Event by PullPoint (new)<br><br>Annex A.10 Retrieve AccessGranted Event by PullPoint for Identifier External Authorization (new)<br><br>Annex A.11 Retrieve Denied Event by PullPoint for Identifier External Authorization (new) |
| 20.06 | Feb 13, 2020 | The following were updated in the scope of #1676:<br><br>ACCESSCONTROL-14-1-1 GET ACCESS POINTS (new)<br><br>ACCESSCONTROL-14-1-2 GET ACCESS POINT LIST - LIMIT (new)<br><br>ACCESSCONTROL-14-1-3 GET ACCESS POINT LIST - START REFERENCE AND LIMIT (new)<br><br>ACCESSCONTROL-14-1-4 GET ACCESS POINT LIST - NO LIMIT (new)<br><br>ACCESSCONTROL-15-1-1 GET AREAS (new)<br><br>ACCESSCONTROL-15-1-2 GET AREA LIST - LIMIT (new) |

| | | |
|---|---|---|
| | | ACCESSCONTROL-15-1-3 GET AREA LIST - START REFERENCE AND LIMIT (new)<br><br>ACCESSCONTROL-15-1-4 GET AREA LIST - NO LIMIT (new)<br><br>Annex A.12 Get Access Points List (new)<br><br>Annex A.13 Get Service Capabilities (new)<br><br>Annex A.14 Get Areas List (new) |
| 20.06 | Feb 28, 2020 | The following test case was updated in the scope of #1676:<br><br>ACCESSCONTROL-1-1-2 GET SERVICES AND GET ACCESS CONTROL SERVICE CAPABILITIES CONSISTENCY (NOTE updated with adding of new capabilities in check)<br><br>The following test cases and annexes were added in the scope of #1676:<br><br>ACCESSCONTROL-14-1-5 CREATE ACCESS POINT (ACCESS POINT CAPABILITIES TRUE)<br><br>ACCESSCONTROL-14-1-6 CREATE ACCESS POINT (ACCESS POINT CAPABILITIES FALSE)<br><br>ACCESSCONTROL-14-1-7 MODIFY ACCESS POINT<br><br>ACCESSCONTROL-14-1-8 DELETE ACCESS POINT<br><br>ACCESSCONTROL-14-1-9 GET ACCESS POINTS WITH INVALID TOKEN<br><br>ACCESSCONTROL-14-1-10 GET ACCESS POINTS - TOO MANY ITEMS<br><br>ACCESSCONTROL-14-1-11 CREATE ACCESS POINT - NOT EMPTY ACCESS POINT TOKEN<br><br>ACCESSCONTROL-14-1-12 MODIFY ACCESS POINT WITH INVALID TOKEN<br><br>ACCESSCONTROL-14-1-13 DELETE ACCESS POINT WITH INVALID TOKEN<br><br>ACCESSCONTROL-15-1-5 CREATE AREA<br><br>ACCESSCONTROL-15-1-6 MODIFY AREA<br><br>ACCESSCONTROL-15-1-7 DELETE AREA<br><br>ACCESSCONTROL-15-1-8 GET AREAS WITH INVALID TOKEN<br><br>ACCESSCONTROL-15-1-9 GET AREAS - TOO MANY ITEMS<br><br>ACCESSCONTROL-15-1-10 CREATE AREA - NOT EMPTY AREA TOKEN<br><br>ACCESSCONTROL-15-1-11 MODIFY AREA WITH INVALID TOKEN<br><br>ACCESSCONTROL-15-1-12 DELETE AREA WITH INVALID TOKEN<br><br>A.15 Free Storage for Additional Access Point |

| | | |
|---|---|---|
| | | A.16 Find Access Point To Delete<br><br>A.17 Retrieve Access Point Changed Event by PullPoint (similar to annex for DoorCahnged event)<br><br>A.18 Get Area Token<br><br>A.19 Create Area<br><br>A.20 Get Door Token<br><br>A.21 Create Door<br><br>A.22 Get Door Control Service Capabilities<br><br>A.23 Get Complete Door Info List<br><br>A.24 Get Access Point<br><br>A.25 Get Access Point Info<br><br>A.26 Delete Access Point<br><br>A.27 Create Access Point<br><br>A.28 Retrieve Access Point Removed Event by PullPoint<br><br>A.29 Free Storage for Additional Area<br><br>A.30 Find Area To Delete<br><br>A.31 Retrieve Area Changed Event by PullPoint<br><br>A.32 Get Area<br><br>A.33 Get Area Info<br><br>A.34 Delete Area<br><br>A.35 Retrieve Area Removed Event by PullPoint |
| 20.06 | Feb 28, 2020 | The following test cases and annexes were added in the scope of #1669:<br><br>ACCESSCONTROL-14-1-14 CREATE NEW ACCESS POINT WITH SET ACCESS POINT (ACCESS POINT CAPABILITIES TRUE)<br><br>ACCESSCONTROL-14-1-15 CREATE NEW ACCESS POINT WITH SET ACCESS POINT (ACCESS POINT CAPABILITIES FALSE)<br><br>ACCESSCONTROL-14-1-16 MODIFY ACCESS POINT WITH SET ACCESS POINT<br><br>ACCESSCONTROL-14-1-17 SET ACCESS POINT - EMPTY ACCESS POINT TOKEN<br><br>ACCESSCONTROL-15-1-13 CREATE NEW AREA WITH SET AREA<br><br>ACCESSCONTROL-15-1-14 MODIFY AREA WITH SET AREA<br><br>ACCESSCONTROL-15-1-15 SET AREA - EMPTY AREA TOKEN |
| 20.06 | Mar 05, 2020 | The following test cases were updated in the scope of #1954: |

| | | |
|---|---|---|
| | | ACCESSCONTROL-1-1-1 ACCESS CONTROL SERVICE CAPABILITIES (steps 5 and 6 added) |
| 20.06 | May 13, 2020 | Pre-Requisite of the following test cases updated with adding of Pull-Point Notification feature according to #1999:<br><br>ACCESSCONTROL-11-1-1 ACCESS CONTROL - ACCESS GRANTED TO ANONYMOUS (EXTERNAL AUTHORIZATION)<br><br>ACCESSCONTROL-11-1-2 ACCESS CONTROL - ACCESS DENIED TO ANONYMOUS (EXTERNAL AUTHORIZATION)<br><br>ACCESSCONTROL-11-1-3 ACCESS CONTROL - ACCESS TIMEOUT TO ANONYMOUS (EXTERNAL AUTHORIZATION)<br><br>ACCESSCONTROL-11-1-4 ACCESS CONTROL - ACCESS GRANTED WITH CREDENTIAL (EXTERNAL AUTHORIZATION)<br><br>ACCESSCONTROL-11-1-5 ACCESS CONTROL - ACCESS DENIED WITH CREDENTIAL (EXTERNAL AUTHORIZATION)<br><br>ACCESSCONTROL-11-1-6 ACCESS CONTROL - ACCESS TIMEOUT WITH CREDENTIAL (EXTERNAL AUTHORIZATION)<br><br>ACCESSCONTROL-11-1-9 ACCESS CONTROL - ACCESS GRANTED TO IDENTIFIER (EXTERNAL AUTHORIZATION)<br><br>ACCESSCONTROL-11-1-10 ACCESS CONTROL - ACCESS DENIED TO IDENTIFIER (EXTERNAL AUTHORIZATION)<br><br>ACCESSCONTROL-11-1-11 ACCESS CONTROL - ACCESS TIMEOUT TO IDENTIFIER (EXTERNAL AUTHORIZATION)<br><br>ACCESSCONTROL-5-1-1 ACCESS CONTROL - ACCESS POINT ENABLED EVENT<br><br>ACCESSCONTROL-5-1-2 ACCESS CONTROL - ACCESS POINT ENABLED EVENT STATE CHANGE<br><br>ACCESSCONTROL-6-1-1 ACCESS CONTROL - ACCESS GRANTED TO ANONYMOUS EVENT<br><br>ACCESSCONTROL-6-1-2 ACCESS CONTROL - ACCESS GRANTED WITH CREDENTIAL EVENT<br><br>ACCESSCONTROL-6-1-3 ACCESS CONTROL - ACCESS GRANTED TO IDENTIFIER EVENT<br><br>ACCESSCONTROL-7-1-1 ACCESS CONTROL - ACCESS TAKEN BY ANONYMOUS EVENT<br><br>ACCESSCONTROL-7-1-2 ACCESS CONTROL - ACCESS TAKEN WITH CREDENTIAL EVENT<br><br>ACCESSCONTROL-8-1-1 ACCESS CONTROL - ACCESS NOT TAKEN BY ANONYMOUS EVENT<br><br>ACCESSCONTROL-8-1-2 ACCESS CONTROL - ACCESS NOT TAKEN WITH CREDENTIAL EVENT<br><br>ACCESSCONTROL-9-1-1 ACCESS CONTROL – ACCESS DENIED TO ANONYMOUS EVENT<br><br>ACCESSCONTROL-9-1-2 ACCESS CONTROL – ACCESS DENIED WITH CREDENTIAL EVENT |

| | | |
|---|---|---|
| | | ACCESSCONTROL-9-1-3 ACCESS CONTROL – ACCESS DENIED WITH CREDENTIAL EVENT (CREDENTIAL NOT FOUND – CARD)<br><br>ACCESSCONTROL-9-1-4 ACCESS CONTROL – ACCESS DENIED WITH CREDENTIAL EVENT (CREDENTIAL NOT FOUND)<br><br>ACCESSCONTROL-9-1-5 ACCESS CONTROL – ACCESS DENIED TO IDENTIFIER EVENT<br><br>ACCESSCONTROL-10-1-2 ACCESS CONTROL – DURESS<br><br>ACCESSCONTROL-12-1-1 ACCESS CONTROL – ADD OR CHANGE ACCESS POINT EVENT<br><br>ACCESSCONTROL-12-1-2 ACCESS CONTROL – REMOVE ACCESS POINT EVENT<br><br>ACCESSCONTROL-13-1-1 ACCESS CONTROL – ADD OR CHANGE AREA EVENT<br><br>ACCESSCONTROL-13-1-2 ACCESS CONTROL – REMOVE AREA EVENT<br><br>ACCESSCONTROL-14-1-5 CREATE ACCESS POINT (ACCESS POINT CAPABILITIES TRUE)<br><br>ACCESSCONTROL-14-1-6 CREATE ACCESS POINT (ACCESS POINT CAPABILITIES FALSE)<br><br>ACCESSCONTROL-14-1-7 MODIFY ACCESS POINT<br><br>ACCESSCONTROL-14-1-8 DELETE ACCESS POINT<br><br>ACCESSCONTROL-15-1-5 CREATE AREA<br><br>ACCESSCONTROL-15-1-6 MODIFY AREA<br><br>ACCESSCONTROL-15-1-7 DELETE AREA |
| 20.12 | Oct 26, 2020 | Access Point Entity added into Pre-Requisite of the following test cases in the scope of #1866:<br><br>ACCESSCONTROL-2-1-1 GET ACCESS POINT INFO<br><br>ACCESSCONTROL-2-1-4 GET ACCESS POINT INFO LIST - START REFERENCE AND LIMIT<br><br>ACCESSCONTROL-2-1-6 GET ACCESS POINT INFO - TOO MANY ITEMS<br><br>ACCESSCONTROL-2-1-7 GET ACCESS POINT STATE<br><br>ACCESSCONTROL-2-1-8 GET ACCESS POINT STATE WITH INVALID TOKEN<br><br>ACCESSCONTROL-2-1-9 ENABLE/DISABLE ACCESS POINT<br><br>ACCESSCONTROL-2-1-10 ENABLE/DISABLE ACCESS POINT - COMMAND NOT SUPPORTED<br><br>ACCESSCONTROL-2-1-11 ENABLE ACCESS POINT WITH INVALID TOKEN |

| | | |
|---|---|---|
| | | ACCESSCONTROL-2-1-12 DISABLE ACCESS POINT WITH INVALID TOKEN<br><br>ACCESSCONTROL-4-1-1 GET AREA INFO LIST AND GET ACCESS POINT INFO LIST CONSISTENCY<br><br>ACCESSCONTROL-5-1-1 ACCESS CONTROL - ACCESS POINT ENABLED EVENT<br><br>ACCESSCONTROL-5-1-2 ACCESS CONTROL - ACCESS POINT ENABLED EVENT STATE CHANGE<br><br>ACCESSCONTROL-6-1-1 ACCESS CONTROL - ACCESS GRANTED TO ANONYMOUS EVENT<br><br>ACCESSCONTROL-6-1-2 ACCESS CONTROL - ACCESS GRANTED WITH CREDENTIAL EVENT<br><br>ACCESSCONTROL-6-1-3 ACCESS CONTROL - ACCESS GRANTED TO IDENTIFIER EVENT<br><br>ACCESSCONTROL-7-1-1 ACCESS CONTROL - ACCESS TAKEN BY ANONYMOUS EVENT<br><br>ACCESSCONTROL-7-1-2 ACCESS CONTROL - ACCESS TAKEN WITH CREDENTIAL EVENT<br><br>ACCESSCONTROL-8-1-1 ACCESS CONTROL - ACCESS NOT TAKEN BY ANONYMOUS EVENT<br><br>ACCESSCONTROL-8-1-2 ACCESS CONTROL - ACCESS NOT TAKEN WITH CREDENTIAL EVENT<br><br>ACCESSCONTROL-9-1-1 ACCESS CONTROL - ACCESS DENIED TO ANONYMOUS EVENT<br><br>ACCESSCONTROL-9-1-2 ACCESS CONTROL - ACCESS DENIED WITH CREDENTIAL EVENT<br><br>ACCESSCONTROL-9-1-3 ACCESS CONTROL - ACCESS DENIED WITH CREDENTIAL EVENT (CREDENTIAL NOT FOUND - CARD)<br><br>ACCESSCONTROL-9-1-4 ACCESS CONTROL - ACCESS DENIED WITH CREDENTIAL EVENT (CREDENTIAL NOT FOUND)<br><br>ACCESSCONTROL-9-1-5 ACCESS CONTROL - ACCESS DENIED TO IDENTIFIER EVENT<br><br>ACCESSCONTROL-10-1-2 ACCESS CONTROL - DURESS<br><br>ACCESSCONTROL-11-1-1 ACCESS CONTROL - ACCESS GRANTED TO ANONYMOUS (EXTERNAL AUTHORIZATION)<br><br>ACCESSCONTROL-11-1-2 ACCESS CONTROL - ACCESS DENIED TO ANONYMOUS (EXTERNAL AUTHORIZATION)<br><br>ACCESSCONTROL-11-1-3 ACCESS CONTROL - ACCESS TIMEOUT TO ANONYMOUS (EXTERNAL AUTHORIZATION)<br><br>ACCESSCONTROL-11-1-4 ACCESS CONTROL - ACCESS GRANTED WITH CREDENTIAL (EXTERNAL AUTHORIZATION)<br><br>ACCESSCONTROL-11-1-5 ACCESS CONTROL - ACCESS DENIED WITH CREDENTIAL (EXTERNAL AUTHORIZATION) |

| | | |
|---|---|---|
| | | ACCESSCONTROL-11-1-6 ACCESS CONTROL - ACCESS TIMEOUT WITH CREDENTIAL (EXTERNAL AUTHORIZATION)<br><br>ACCESSCONTROL-11-1-7 EXTERNAL AUTHORIZATION WITH INVALID TOKEN<br><br>ACCESSCONTROL-11-1-8 EXTERNAL AUTHORIZATION - COMMAND NOT SUPPORTED<br><br>ACCESSCONTROL-11-1-9 ACCESS CONTROL - ACCESS GRANTED TO IDENTIFIER (EXTERNAL AUTHORIZATION)<br><br>ACCESSCONTROL-11-1-10 ACCESS CONTROL - ACCESS DENIED TO IDENTIFIER (EXTERNAL AUTHORIZATION)<br><br>ACCESSCONTROL-11-1-11 ACCESS CONTROL - ACCESS TIMEOUT TO IDENTIFIER (EXTERNAL AUTHORIZATION) |
| 20.12 | Oct 26, 2020 | The following test cases were updated in the scope of #1866:<br><br>ACCESSCONTROL-2-1-2 GET ACCESS POINT INFO WITH INVALID TOKEN (step 8 updated)<br><br>ACCESSCONTROL-2-1-3 GET ACCESS POINT INFO LIST - LIMIT (steps 4, 5, 8.4, 12, 13.3 added, step 8 and 13: condition added)<br><br>ACCESSCONTROL-2-1-5 GET ACCESS POINT INFO LIST - NO LIMIT (step 7 added) |
| 20.12 | Nov 09, 2020 | The following test cases were updated in the scope of #2098:<br><br>ACCESSCONTROL-11-1-2 ACCESS CONTROL - ACCESS DENIED TO ANONYMOUS (EXTERNAL AUTHORIZATION) (Reason value in ExternalAuthorization request changed; check of Reason vaue in event removed)<br><br>ACCESSCONTROL-11-1-5 ACCESS CONTROL - ACCESS DENIED WITH CREDENTIAL (EXTERNAL AUTHORIZATION) (Reason value in ExternalAuthorization request changed; check of Reason vaue in event removed) |
| 20.12 | Nov 09, 2020 | The following test cases was added in the scope of #2097:<br><br>ACCESSCONTROL-4-1-2 ACCESS POINT CAPABILITIES CONSISTENCY |
| 20.12 | Nov 10, 2020 | Pre-Requisites of the following test cases were updated in the scope of #1866:<br><br>ACCESSCONTROL-6-1-1 ACCESS CONTROL – ACCESS GRANTED TO ANONYMOUS EVENT<br><br>ACCESSCONTROL-6-1-2 ACCESS CONTROL – ACCESS GRANTED WITH CREDENTIAL EVENT<br><br>ACCESSCONTROL-7-1-2 ACCESS CONTROL – ACCESS TAKEN WITH CREDENTIAL EVENT<br><br>ACCESSCONTROL-8-1-2 ACCESS CONTROL – ACCESS NOT TAKEN WITH CREDENTIAL EVENT<br><br>ACCESSCONTROL-9-1-1 ACCESS CONTROL – ACCESS DENIED TO ANONYMOUS EVENT<br><br>ACCESSCONTROL-9-1-2 ACCESS CONTROL – ACCESS DENIED WITH CREDENTIAL EVENT |

| | | |
|---|---|---|
| | | ACCESSCONTROL-11-1-4 ACCESS CONTROL – ACCESS GRANTED WITH CREDENTIAL (EXTERNAL AUTHORIZATION)<br><br>ACCESSCONTROL-11-1-5 ACCESS CONTROL – ACCESS DENIED WITH CREDENTIAL (EXTERNAL AUTHORIZATION)<br><br>ACCESSCONTROL-11-1-6 ACCESS CONTROL – ACCESS TIMEOUT WITH CREDENTIAL (EXTERNAL AUTHORIZATION) |
| 20.12 | Dec 09, 2020 | 'IdentiferAccess' was repleced with 'IdentifierAccess' according to wsdl change in the scope of #2133. |

Table of Contents

**1 Introduction ..... 19**

1.1 Scope ..... 19

1.1.1 Capabilities ..... 20

1.1.2 Access Point ..... 20

1.1.3 Area ..... 20

1.1.4 External Authorization ..... 20

1.1.5 Property Events ..... 21

1.1.6 Access Granted Events ..... 21

1.1.7 Access taken events ..... 21

1.1.8 Access not taken events ..... 21

1.1.9 Access Denied Events ..... 22

1.1.10 Duress events ..... 22

1.1.11 Consistency ..... 22

1.1.12 Access Point Change Events ..... 22

1.1.13 Area Change Events ..... 22

**2 Terms and Definitions ..... 24**

2.1 Definitions ..... 24

2.2 Abbreviations ..... 24

**3 Test Overview ..... 25**

3.1 Test Setup ..... 25

3.1.1 Network Configuration for DUT ..... 25

3.2 Prerequisites ..... 26

3.3 Test Policy ..... 28

3.3.1 Capabilities ..... 28

3.3.2 Access Point ..... 28

3.3.3 Area ..... 29

3.3.4 External Authorization ..... 30

3.3.5 Property Events ..... 31

3.3.6 Access Granted Events ..... 32

3.3.7 Access taken events ..... 33

3.3.8 Access not taken events ..... 34
3.3.9 Access Denied Events ..... 35
3.3.10 Duress events ..... 36
3.3.11 Consistency ..... 36
3.3.12 Access Point Change Events ..... 37
3.3.13 Area Change Events ..... 37
**4 Access Control Test Cases ..... 39**
4.1 Capabilities ..... 39
4.1.1 ACCESS CONTROL SERVICE CAPABILITIES ..... 39
4.1.2 GET SERVICES AND GET ACCESS CONTROL SERVICE CAPABILITIES CONSISTENCY ..... 40
4.2 Access Point ..... 41
4.2.1 GET ACCESS POINT INFO ..... 41
4.2.2 GET ACCESS POINT INFO WITH INVALID TOKEN ..... 43
4.2.3 GET ACCESS POINT INFO LIST - LIMIT ..... 44
4.2.4 GET ACCESS POINT INFO LIST - START REFERENCE AND LIMIT ..... 47
4.2.5 GET ACCESS POINT INFO LIST - NO LIMIT ..... 49
4.2.6 GET ACCESS POINT INFO - TOO MANY ITEMS ..... 51
4.2.7 GET ACCESS POINT STATE ..... 52
4.2.8 GET ACCESS POINT STATE WITH INVALID TOKEN ..... 53
4.2.9 ENABLE/DISABLE ACCESS POINT ..... 54
4.2.10 ENABLE/DISABLE ACCESS POINT - COMMAND NOT SUPPORTED ..... 57
4.2.11 ENABLE ACCESS POINT WITH INVALID TOKEN ..... 58
4.2.12 DISABLE ACCESS POINT WITH INVALID TOKEN ..... 59
4.3 Area ..... 60
4.3.1 GET AREA INFO ..... 60
4.3.2 GET AREA INFO WITH INVALID TOKEN ..... 62
4.3.3 GET AREA INFO LIST - LIMIT ..... 63
4.3.4 GET AREA INFO LIST - START REFERENCE AND LIMIT ..... 65
4.3.5 GET AREA INFO - TOO MANY ITEMS ..... 67
4.3.6 GET AREA INFO LIST - NO LIMIT ..... 68

4.4 External Authorization ........................................................................................ 70

4.4.1 ACCESS CONTROL - ACCESS GRANTED TO ANONYMOUS (EXTERNAL AUTHORIZATION) ............................................................................ 70

4.4.2 ACCESS CONTROL - ACCESS DENIED TO ANONYMOUS (EXTERNAL AUTHORIZATION) ............................................................................ 73

4.4.3 ACCESS CONTROL - ACCESS TIMEOUT TO ANONYMOUS (EXTERNAL AUTHORIZATION) ............................................................................ 77

4.4.4 ACCESS CONTROL - ACCESS GRANTED WITH CREDENTIAL (EXTERNAL AUTHORIZATION) ............................................................................ 80

4.4.5 ACCESS CONTROL - ACCESS DENIED WITH CREDENTIAL (EXTERNAL AUTHORIZATION) ............................................................................ 84

4.4.6 ACCESS CONTROL - ACCESS TIMEOUT WITH CREDENTIAL (EXTERNAL AUTHORIZATION) ............................................................................ 88

4.4.7 EXTERNAL AUTHORIZATION WITH INVALID TOKEN ............................... 91

4.4.8 EXTERNAL AUTHORIZATION - COMMAND NOT SUPPORTED ................ 92

4.4.9 ACCESS CONTROL - ACCESS GRANTED TO IDENTIFIER (EXTERNAL AUTHORIZATION) ............................................................................ 93

4.4.10 ACCESS CONTROL - ACCESS DENIED TO IDENTIFIER (EXTERNAL AUTHORIZATION) ............................................................................ 97

4.4.11 ACCESS CONTROL - ACCESS TIMEOUT TO IDENTIFIER (EXTERNAL AUTHORIZATION) ............................................................................ 100

4.5 Property Events ........................................................................................ 103

4.5.1 ACCESS CONTROL - ACCESS POINT ENABLED EVENT ...................... 103

4.5.2 ACCESS CONTROL - ACCESS POINT ENABLED EVENT STATE CHANGE ............................................................................ 106

4.6 Access granted events ........................................................................................ 109

4.6.1 ACCESS CONTROL - ACCESS GRANTED TO ANONYMOUS EVENT .... 109

4.6.2 ACCESS CONTROL - ACCESS GRANTED WITH CREDENTIAL EVENT . 112

4.6.3 ACCESS CONTROL - ACCESS GRANTED TO IDENTIFIER EVENT ....... 115

4.7 Access taken events ........................................................................................ 117

4.7.1 ACCESS CONTROL - ACCESS TAKEN BY ANONYMOUS EVENT .......... 117

4.7.2 ACCESS CONTROL - ACCESS TAKEN WITH CREDENTIAL EVENT ...... 120

4.8 Access not taken events ........................................................................ 123

4.8.1 ACCESS CONTROL - ACCESS NOT TAKEN BY ANONYMOUS EVENT . 123

4.8.2 ACCESS CONTROL - ACCESS NOT TAKEN WITH CREDENTIAL EVENT ........................................................................................................ 126

4.9 Access Denied Events ........................................................................... 130

4.9.1 ACCESS CONTROL - ACCESS DENIED TO ANONYMOUS EVENT ........ 130

4.9.2 ACCESS CONTROL - ACCESS DENIED WITH CREDENTIAL EVENT .... 133

4.9.3 ACCESS CONTROL - ACCESS DENIED WITH CREDENTIAL EVENT (CREDENTIAL NOT FOUND - CARD) ................................................................. 136

4.9.4 ACCESS CONTROL - ACCESS DENIED WITH CREDENTIAL EVENT (CREDENTIAL NOT FOUND) .................................................................... 138

4.9.5 ACCESS CONTROL - ACCESS DENIED TO IDENTIFIER EVENT ........... 141

4.10 Duress events ........................................................................................ 143

4.10.1 ACCESS CONTROL - DURESS ........................................................ 143

4.11 Consistency ............................................................................................ 146

4.11.1 GET AREA INFO LIST AND GET ACCESS POINT INFO LIST CONSISTENCY ........................................................................................ 146

4.11.2 ACCESS POINT CAPABILITIES CONSISTENCY .................................... 147

4.12 Access Point Configuration ....................................................................... 148

4.12.1 ACCESS CONTROL - ADD OR CHANGE ACCESS POINT EVENT ....... 148

4.12.2 ACCESS CONTROL - REMOVE ACCESS POINT EVENT ..................... 150

4.13 Area Configuration .................................................................................. 153

4.13.1 ACCESS CONTROL - ADD OR CHANGE AREA EVENT ....................... 153

4.13.2 ACCESS CONTROL - REMOVE AREA EVENT ..................................... 155

4.14 Access Point Management ....................................................................... 157

4.14.1 GET ACCESS POINTS ......................................................................... 157

4.14.2 GET ACCESS POINT LIST - LIMIT ...................................................... 160

4.14.3 GET ACCESS POINT LIST - START REFERENCE AND LIMIT .............. 162

4.14.4 GET ACCESS POINT LIST - NO LIMIT ................................................ 167

4.14.5 CREATE ACCESS POINT (ACCESS POINT CAPABILITIES TRUE) ....... 170

4.14.6 CREATE ACCESS POINT (ACCESS POINT CAPABILITIES FALSE) ...... 175
4.14.7 MODIFY ACCESS POINT ........................................................................ 180
4.14.8 DELETE ACCESS POINT ........................................................................ 186
4.14.9 GET ACCESS POINTS WITH INVALID TOKEN ..................................... 189
4.14.10 GET ACCESS POINTS - TOO MANY ITEMS ....................................... 191
4.14.11 CREATE ACCESS POINT - NOT EMPTY ACCESS POINT TOKEN ...... 192
4.14.12 MODIFY ACCESS POINT WITH INVALID TOKEN ................................ 194
4.14.13 DELETE ACCESS POINT WITH INVALID TOKEN ................................ 196
4.14.14 CREATE NEW ACCESS POINT WITH SET ACCESS POINT (ACCESS POINT CAPABILITIES TRUE) ............................................................ 197
4.14.15 CREATE NEW ACCESS POINT WITH SET ACCESS POINT (ACCESS POINT CAPABILITIES FALSE) ........................................................... 203
4.14.16 MODIFY ACCESS POINT WITH SET ACCESS POINT ......................... 208
4.14.17 SET ACCESS POINT - EMPTY ACCESS POINT TOKEN ..................... 214
4.15 Area Management ................................................................................. 216
4.15.1 GET AREAS ............................................................................................ 216
4.15.2 GET AREA LIST - LIMIT ......................................................................... 218
4.15.3 GET AREA LIST - START REFERENCE AND LIMIT ............................. 220
4.15.4 GET AREA LIST - NO LIMIT .................................................................. 225
4.15.5 CREATE AREA ....................................................................................... 227
4.15.6 MODIFY AREA ....................................................................................... 229
4.15.7 DELETE AREA ....................................................................................... 233
4.15.8 GET AREAS WITH INVALID TOKEN ..................................................... 235
4.15.9 GET AREAS - TOO MANY ITEMS ......................................................... 237
4.15.10 CREATE AREA - NOT EMPTY AREA TOKEN ..................................... 238
4.15.11 MODIFY AREA WITH INVALID TOKEN ................................................ 239
4.15.12 DELETE AREA WITH INVALID TOKEN ............................................... 240
4.15.13 CREATE NEW AREA WITH SET AREA ............................................... 241
4.15.14 MODIFY AREA WITH SET AREA ......................................................... 244
4.15.15 SET AREA - EMPTY AREA TOKEN ..................................................... 247
**A Helper Procedures and Additional Notes ................................................ 249**

A.1 Get Complete Access Point Info List .......... 249
A.2 Get Complete Area Info List .......... 250
A.3 Create Pull Point Subscription .......... 250
A.4 Delete Subscription .......... 251
A.5 Retrieve CredentialNotFound Event by PullPoint .......... 251
A.6 Retrieve Access Granted Identifier Event by PullPoint .......... 253
A.7 Retrieve Denied Identifier Event by PullPoint .......... 255
A.8 Retrieve Request Identifier Event by PullPoint .......... 257
A.9 Retrieve Request Timeout Event by PullPoint .......... 258
A.10 Retrieve AccessGranted Event by PullPoint for Identifier External Authorization . 260
A.11 Retrieve Denied Event by PullPoint for Identifier External Authorization .......... 261
A.12 Get Access Points List .......... 263
A.13 Get Service Capabilities .......... 264
A.14 Get Areas List .......... 265
A.15 Free Storage for Additional Access Point .......... 266
A.16 Find Access Point To Delete .......... 267
A.17 Retrieve Access Point Changed Event by PullPoint .......... 268
A.18 Get Area Token .......... 269
A.19 Create Area .......... 270
A.20 Get Door Token .......... 271
A.21 Create Door .......... 272
A.22 Get Door Control Service Capabilities .......... 274
A.23 Get Complete Door Info List .......... 275
A.24 Get Access Point .......... 275
A.25 Get Access Point Info .......... 276
A.26 Delete Access Point .......... 276
A.27 Create Access Point .......... 277
A.28 Retrieve Access Point Removed Event by PullPoint .......... 279
A.29 Free Storage for Additional Area .......... 280
A.30 Find Area To Delete .......... 281
A.31 Retrieve Area Changed Event by PullPoint .......... 281

A.32 Get Area ..... 283

A.33 Get Area Info ..... 283

A.34 Delete Area ..... 284

A.35 Retrieve Area Removed Event by PullPoint ..... 285

# 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 needed to verify the [ONVIF Network Interface Specs] and [ONVIF Conformance] requirements. And also 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 Control Test Specification acts as a supplementary document to the [ONVIF Network Interface Specs], illustrating necessary test cases to be executed and passed. And also this specification acts as an input document to the development of test tool which will be used to test the ONVIF device implementation conformance towards ONVIF standard. This test tool is referred as ONVIF Client hereafter.

## 1.1 Scope

This ONVIF Access Control 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 for testing individual requirements of ONVIF devices according to ONVIF Access Control service which is defined in [ONVIF Network Interface Specs].

The principal intended purposes are:

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

This specification does not address the following:

1. Product use cases and non-functional (performance and regression) testing.
2. SOAP Implementation Interoperability test i.e. Web Service Interoperability Basic Profile version 2.0 (WS-I BP 2.0).
3. Network protocol implementation Conformance test for HTTP, HTTPS, RTP protocol.
4. Wi-Fi Conformance test

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

This ONVIF Access Control Test Specification covers ONVIF Access Control service and Access Control Events which are a functional block of [ONVIF Network Interface Specs]. The following sections give a brief overview and scope of each functional block.

### 1.1.1 Capabilities

Capabilities test cases cover verification of getting Access Control Service capabilities. It means that the following commands are covered by these test cases:

- GetServices (Device Management Service)
- GetServiceCapabilities

### 1.1.2 Access Point

Access Point test cases cover verification of getting Access Point list, information, state and to enable/disable access point. It means that the following commands are covered by these test cases:

- GetAccessPointInfo
- GetAccessPointInfoList
- GetAccessPointState
- EnableAccessPoint
- DisableAccessPoint

### 1.1.3 Area

Area test cases cover verification of getting Area list and information. It means that the following commands are covered by these test cases:

- GetAreaInfo
- GetAreaInfoList

### 1.1.4 External Authorization

External Authorization test cases cover verification of external authorization procedure. It means that the following commands and events are covered by these test cases:

- ExternalAuthorization
- tns1:AccessControl/AccessGranted/Anonymous
- tns1:AccessControl/AccessGranted/Credential
- tns1:AccessControl/Denied/Anonymous

- tns1:AccessControl/Denied/Credential
- tns1:AccessControl/Request/Anonymous
- tns1:AccessControl/Request/Timeout
- tns1:AccessControl/Request/Credential
- tns1:AccessControl/Request/Identifier

### 1.1.5 Property Events

Property events test cases cover verification of property events provided by Access Control Service. It means that the following commands and events are covered by these test cases:

- EnableAccessPoint
- DisableAccessPoint
- tns1:AccessPoint/State/Enabled

### 1.1.6 Access Granted Events

Access Granted events test cases cover verification of access granted events provided by Access Control Service. It means that the following events are covered by these test cases:

- tns1:AccessControl/AccessGranted/Anonymous
- tns1:AccessControl/AccessGranted/Credential
- tns1:AccessControl/AccessGranted/Identifier

### 1.1.7 Access taken events

Access taken events test cases cover verification of access taken events provided by Access Control Service. It means that the following events are covered by these test cases:

- tns1:AccessControl/AccessTaken/Anonymous
- tns1:AccessControl/AccessTaken/Credential

### 1.1.8 Access not taken events

Access not taken events test cases cover verification of access not taken events provided by Access Control Service. It means that the following events are covered by these test cases:

- tns1:AccessControl/AccessNotTaken/Anonymous
- tns1:AccessControl/AccessNotTaken/Credential

### 1.1.9 Access Denied Events

Access denied events test cases cover verification of access denied events provided by Access Control Service. It means that the following events are covered by these test cases:

- tns1:AccessControl/Denied/Anonymous
- tns1:AccessControl/Denied/Credential
- tns1:AccessControl/Denied/CredentialNotFound/Card
- tns1:AccessControl/Denied/CredentialNotFound

### 1.1.10 Duress events

Duress events test cases cover verification of duress events provided by Access Control Service. It means that the following events are covered by the following test case:

- tns:AccessControl/Duress

### 1.1.11 Consistency

Consistency test cases cover verification of consistency between different entities and commands. It means that consistency between the following entities is covered by the following test case:

- Access Point Info and Get Area Info

### 1.1.12 Access Point Change Events

Access point change events test cases cover verification of access point change events provided by Access Control Service. It means that the following events are covered by these test cases:

- tns1:Configuration/AccessPoint/Change
- tns1:Configuration/AccessPoint/Removed

### 1.1.13 Area Change Events

Area change events test cases cover verification of area change events provided by Access Control Service. It means that the following events are covered by these test cases:

- tns1:Configuration/Area/Change
- tns1:Configuration/Area/Removed

# 2 Terms and Definitions

## 2.1 Definitions

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

| | |
|---|---|
| **Access Point** | A logical composition of a physical door and ID point(s) controlling access in one direction. |
| **Access Point Disable** | If an Access Point is disabled, it will not be considered in the decision making process and no commands will be issued from that Access Point to the Door configured for that Access Point. When an Access Point is disabled, the associated ID Point may or may not be disabled or shut down. Clients may still be able to command the Door Controller to control associated door even though that door is also referenced by a disabled access point. |
| **Credential** | 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. |
| **Door** | A physical door, barrier, turnstile, etc which can be controlled remotely and restricts access between two areas. A door is usually equipped with an electronic lock and a door monitor. |
| **Duress** | Forcing a person to provide access to a secure area against that person's wishes. |
| **ID Point** | A device that converts reader signals to protocols recognized by an authorization engine. It can be card reader, REX, biometric reader etc. |

## 2.2 Abbreviations

This section describes abbreviations used in this document.

| | |
|---|---|
| **HTTP** | Hypertext Transfer (or Transport) Protocol |
| **PACS** | Physical Access Control System |

# 3 Test Overview

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

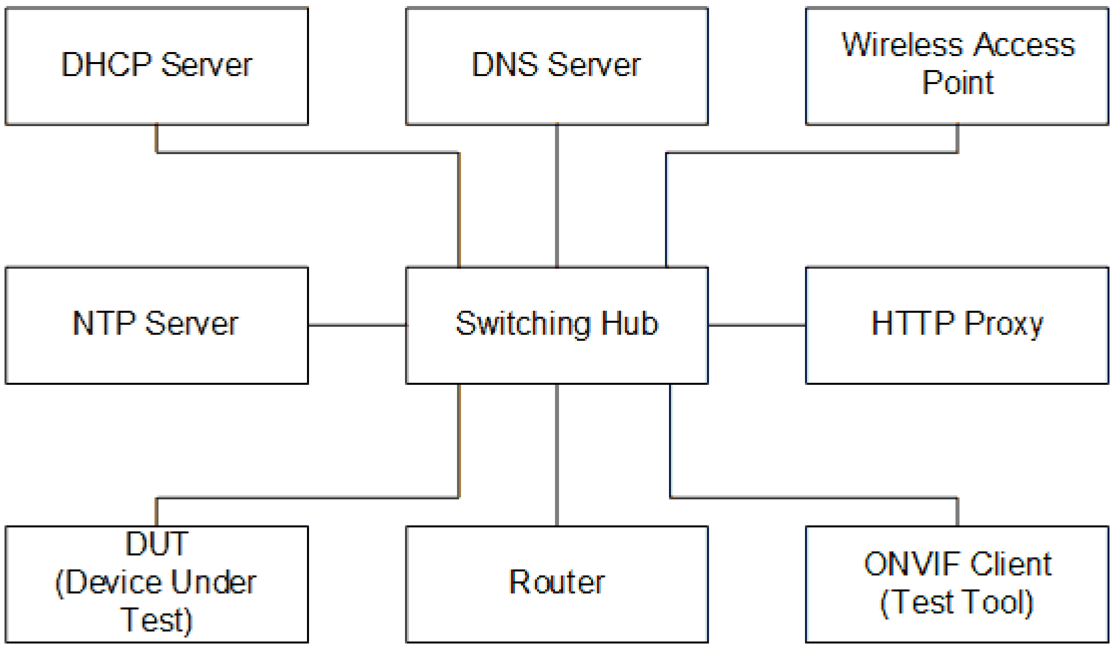
## 3.1 Test Setup

### 3.1.1 Network Configuration for DUT

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

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

**Figure 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 following:

- 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 ONVIF Device 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.
- The DUT time and ONVIF Device Test tool time shall be synchronized with each other either manually or by common NTP server.
- Test Operator shall configure Operation Delay properly so that it would have enough time to receive Notification messages for the following test cases for ONVIF Device Test Tool (see test description for more details):
  - ACCESSCONTROL-11-1-1 (see Section 4.4.1)
  - ACCESSCONTROL-11-1-2 (see Section 4.4.2)
  - ACCESSCONTROL-11-1-3 (see Section 4.4.3)
  - ACCESSCONTROL-11-1-4 (see Section 4.4.4)
  - ACCESSCONTROL-11-1-5 (see Section 4.4.5)
  - ACCESSCONTROL-11-1-6 (see Section 4.4.6)
  - ACCESSCONTROL-5-1-1 (see Section 4.5.1)
  - ACCESSCONTROL-5-1-2 (see Section 4.5.2)

  - ACCESSCONTROL-6-1-1 (see Section 4.6.1)
  - ACCESSCONTROL-6-1-2 (see Section 4.6.2)
  - ACCESSCONTROL-6-1-3 (see Section 4.6.3)
  - ACCESSCONTROL-7-1-1 (see Section 4.7.1)
  - ACCESSCONTROL-7-1-2 (see Section 4.7.2)
  - ACCESSCONTROL-8-1-1 (see Section 4.8.1)
  - ACCESSCONTROL-8-1-2 (see Section 4.8.2)
  - ACCESSCONTROL-9-1-1 (see Section 4.9.1)
  - ACCESSCONTROL-9-1-2 (see Section 4.9.2)
  - ACCESSCONTROL-9-1-3 (see Section 4.9.3)
  - ACCESSCONTROL-9-1-4 (see Section 4.9.4)
  - ACCESSCONTROL-9-1-5 (see Section 4.9.5)
  - ACCESSCONTROL-10-1-2 (see Section 4.10.1)
  - ACCESSCONTROL-11-1-9 (see Section 4.4.9)
  - ACCESSCONTROL-11-1-10 (see Section 4.4.10)
  - ACCESSCONTROL-11-1-11 (see Section 4.4.11)
  - ACCESSCONTROL-12-1-1 (see Section 4.12.1)
  - ACCESSCONTROL-12-1-2 (see Section 4.12.2)
  - ACCESSCONTROL-13-1-1 (see Section 4.13.1)
  - ACCESSCONTROL-13-1-2 (see Section 4.13.2)

- At least one Access Point is configured and added to the DUT.
- At least one Access Point with Enable/Disable capability is configured and added to the DUT, if Enable/Disable capability is supported by the DUT.
- At least one Access Point with Duress capability is configured and added to the DUT, if Duress capability is supported by the DUT.

- At least one Access Point with Access Taken capability is configured and added to the DUT, if Access Taken capability is supported by the DUT.
- At least one Access Point with External Authorization capability is configured and added to the DUT, if External Authorization capability is supported by the DUT.
- At least one Access Point with Identifier Access capability is configured and added to the DUT, if Identifier Access capability is supported by the DUT.
- At least one Access Point with Anonymous Access capability is configured and added to the DUT, if Anonymous Access capability is supported by the DUT.
- At least one Access Point with Anonymous Access capability and External Authorization capability is configured and added to the DUT, if Anonymous Access capability and External Authorization capability are supported by the DUT.
- At least one Access Point with Anonymous Access capability and Access Taken capability is configured and added to the DUT, if Anonymous Access capability and Access Taken capability are supported by the DUT.
- At least one Access Point with tns1:AccessControl/Denied/CredentialNotFound/Card event capability is configured and added to the DUT, if tns1:AccessControl/Denied/CredentialNotFound/Card capability is supported by the DUT.
- At least one Area is configured and added to the DUT, if it is possible for the DUT.

## 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

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

DUT shall support GetServices and GetServiceCapabilities command.

Please, refer to Section 4.1 for Capabilities Test Cases.

### 3.3.2 Access Point

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

DUT shall support GetServices and GetServiceCapabilities command.

DUT shall support at least one Access Point. And also at least one Access Point shall be added and configured on device.

DUT shall support the following commands:

- GetAccessPointInfo
- GetAccessPointInfoList
- GetAccessPointState

DUT shall return capabilities for each specific access point. If DUT returns Disable Access Point capability as supported by access point, then DUT shall support the following commands for this access point:

- DisableAccessPoint
- EnableAccessPoint

DUT shall not return any fault, if GetAccessPointInfo was invoked for nonexistent Access Point token. Such tokens shall be ignored.

DUT shall not return more items in GetAccessPointInfo and GetAccessPointInfoList responses than specified in service capabilities by MaxLimit.

DUT shall not return more items in GetAccessPointInfoList response than specified by Limit parameter in request.

DUT shall return SOAP 1.2 fault message (InvalidArgs/TooManyItems), if more items than MaxLimit were requested by GetAccessPointInfo command.

DUT should return SOAP 1.2 fault message (InvalidArgVal/NotFound), if GetAccessPointState, EnableAccessPoint or DisableAccessPoint command was invoked for nonexistent Access Point token.

DUT should return SOAP 1.2 fault message (ActionNotSupported/NotSupported), if EnableAccessPoint or DisableAccessPoint command was invoked for Access Point without Disable Access Point capability token.

Please, refer to Section 4.2 for Access Point Test Cases.

### 3.3.3 Area

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

DUT shall support GetServiceCapabilities command.

If DUT supports at least one Area, one Area shall be added and configured on device.

DUT shall support the following commands:

- GetAreaInfo
- GetAreaInfoList

DUT shall not return any fault, if GetAreaInfo was invoked for nonexistent Area token. Such tokens shall be ignored.

DUT shall not return more items in GetAreaInfo and GetAreaInfoList responses than specified in service capabilities by MaxLimit.

DUT shall not return more items in GetAreaInfoList response than specified by Limit parameter in request.

DUT shall return SOAP 1.2 fault message (InvalidArgs/TooManyItems), if more items than MaxLimit were requested by GetAreaInfo command.

Please, refer to Section 4.3 for Area Test Cases.

### 3.3.4 External Authorization

DUT shall give the Access Control Service entry point and Event Service entry point by GetServices command, if DUT supports Access Control service. Otherwise, these test cases will be skipped.

DUT shall support GetServiceCapabilities command.

DUT shall return capabilities for each specific access point.

If DUT returns External Authorization capability as supported by access point, then DUT shall support the following commands and events for this access point:

- ExternalAuthorization
- tns1:AccessControl/AccessGranted/Credential
- tns1:AccessControl/Denied/Credential
- tns1:AccessControl/Request/Timeout
- tns1:AccessControl/Request/Credential

If DUT returns External Authorization capability and Anonymous Access capability as supported by access point, then DUT shall support the following events for this access point:

- tns1:AccessControl/AccessGranted/Anonymous
- tns1:AccessControl/Denied/Anonymous
- tns1:AccessControl/Request/Anonymous

If DUT returns Identifier Access capability as supported by access point, then DUT shall support the following events for this access point:

- tns1:AccessControl/AccessGranted/Anonymous or tns1:AccessControl/AccessGranted/Credential
- tns1:AccessControl/Denied/Anonymous or tns1:AccessControl/Denied/Credential
- tns1:AccessControl/Request/Identifier
- tns1:AccessControl/Request/Timeout

DUT shall support GetEventProperties command and return all supported events in TopicSet.

DUT shall support Pull Point Subscription and Topic Expression filter.

DUT should return SOAP 1.2 fault message (InvalidArgVal/NotFound), if ExternalAuthorization command was invoked for nonexistent Access Point token.

DUT should return SOAP 1.2 fault message (ActionNotSupported/NotSupported), if ExternalAuthorization command was invoked for Access Point without External Authorization capability token.

A test operator shall manually invoke the following events if required:

- tns1:AccessControl/Request/Timeout
- tns1:AccessControl/Request/Credential
- tns1:AccessControl/Request/Anonymous
- tns1:AccessControl/Request/Identifier

Please, refer to Section 4.4 for External Authorization Test Cases.

### 3.3.5 Property Events

DUT shall give the Access Control Service entry point and Event Service entry point by GetServices command, if DUT supports Access Control service. Otherwise these test cases will be skipped.

DUT shall support GetServiceCapabilities command.

DUT shall support following property events:

- tns1:AccessPoint/State/Enabled

DUT shall return capabilities for each specific access point. If DUT returns Disable Access Point capability as supported by access point, then DUT shall support the following commands for this access point:

- DisableAccessPoint
- EnableAccessPoint

DUT shall support GetEventProperties command and return all supported events in TopicSet.

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.

Please, refer to Section 4.5 for Property Events Test Cases.

### 3.3.6 Access Granted Events

DUT shall give the Access Control Service entry point and Event Service entry point by GetServices command, if DUT supports Access Control service. Otherwise these test cases will be skipped.

DUT shall support GetServiceCapabilities command.

DUT shall return capabilities for each specific access point.

DUT shall support the following events for each access point:

- tns1:AccessControl/AccessGranted/Credential

If DUT returns Anonymous Access capability as supported by access point, then DUT shall support the following events for this access point:

- tns1:AccessControl/AccessGranted/Anonymous

If DUT returns tns1:AccessControl/AccessGranted/Identifier in topic set, then DUT shall support the following events:

- tns1:AccessControl/AccessGranted/Identifier

DUT shall support GetEventProperties command and return all supported events in TopicSet.

DUT shall support Pull Point Subscription and Topic Expression filter.

A test operator shall manually invoke the following events if required:

- tns1:AccessControl/AccessGranted/Anonymous
- tns1:AccessControl/AccessGranted/Credential
- tns1:AccessControl/AccessGranted/Identifier

Please, refer to Section 4.6 for Access granted events Test Cases.

### 3.3.7 Access taken events

DUT shall give the Access Control Service entry point and Event Service entry point by GetServices command, if DUT supports Access Control service. Otherwise, these test cases will be skipped.

DUT shall support GetServiceCapabilities command.

DUT shall return capabilities for each specific access point.

DUT shall support the following events for each access point:

- tns1:AccessControl/AccessGranted/Credential

If DUT returns Anonymous Access capability as supported by access point, then DUT shall support the following events for this access point:

- tns1:AccessControl/AccessGranted/Anonymous

If DUT returns Access Taken capability as supported by access point, then DUT shall support the following events for this access point:

- tns1:AccessControl/AccessTaken/Credential

If DUT returns Anonymous Access capability and Access Taken capability as supported by access point, then DUT shall support the following events for this access point:

- tns1:AccessControl/AccessTaken/Anonymous

DUT shall support GetEventProperties command and return all supported events in TopicSet.

DUT shall support Pull Point Subscription and Topic Expression filter.

A test operator shall manually invoke the following events if required:

- tns1:AccessControl/AccessGranted/Anonymous
- tns1:AccessControl/AccessGranted/Credential

- tns1:AccessControl/AccessTaken/Anonymous
- tns1:AccessControl/AccessTaken/Credential

Please, refer to Section 4.7 for Access taken events Test Cases.

### 3.3.8 Access not taken events

DUT shall give the Access Control Service entry point and Event Service entry point by GetServices command, if DUT supports Access Control service. Otherwise, these test cases will be skipped.

DUT shall support GetServiceCapabilities command.

DUT shall return capabilities for each specific access point.

DUT shall support the following events for each access point:

- tns1:AccessControl/AccessGranted/Credential

If DUT returns Anonymous Access capability as supported by access point, then DUT shall support the following events for this access point:

- tns1:AccessControl/AccessGranted/Anonymous

If DUT returns Access Taken capability as supported by access point, then DUT shall support the following events for this access point:

- tns1:AccessControl/AccessNotTaken/Credential

If DUT returns Anonymous Access capability and Access Taken capability as supported by access point, then DUT shall support the following events for this access point:

- tns1:AccessControl/AccessNotTaken/Anonymous

DUT shall support GetEventProperties command and return all supported events in TopicSet.

DUT shall support Pull Point Subscription and Topic Expression filter.

A test operator shall manually invoke the following events if required:

- tns1:AccessControl/AccessGranted/Anonymous
- tns1:AccessControl/AccessGranted/Credential
- tns1:AccessControl/AccessNotTaken/Anonymous
- tns1:AccessControl/AccessNotTaken/Credential

Please, refer to Section 4.8 for Access not taken events Test Cases.

### 3.3.9 Access Denied Events

DUT shall give the Access Control Service entry point and Event Service entry point by GetServices command, if DUT supports Access Control service. Otherwise, these test cases will be skipped.

DUT shall support GetServiceCapabilities command.

DUT shall return capabilities for each specific access point.

DUT shall support the following events for each access point:

- tns1:AccessControl/Denied/Credential

If DUT returns tns1:AccessControl/Denied/CredentialNotFound/Card as supported, DUT shall support the following event for at least one configured access point:

- tns1:AccessControl/Denied/CredentialNotFound/Card

If DUT returns tns1:AccessControl/Denied/CredentialNotFound as supported, DUT shall support the following event for at least one configured access point:

- tns1:AccessControl/Denied/CredentialNotFound

If DUT returns Anonymous Access capability as supported by access point, then DUT shall support the following events for this access point:

- tns1:AccessControl/Denied/Anonymous

If DUT returns tns1:AccessControl/Denied/Identifier in topic set, then DUT shall support the following events:

- tns1:AccessControl/Denied/Identifier

DUT shall support GetEventProperties command and return all supported events in TopicSet.

DUT shall support Pull Point Subscription and Topic Expression filter.

A test operator shall manually invoke the following events if required:

- tns1:AccessControl/Denied/Anonymous
- tns1:AccessControl/Denied/Credential
- tns1:AccessControl/Denied/CredentialNotFound/Card
- tns1:AccessControl/Denied/CredentialNotFound

- tns1:AccessControl/Denied/Identifier

Please, refer to Section 4.9 for Access Denied Events Test Cases.

### 3.3.10 Duress events

DUT shall give the Access Control Service entry point and Event Service entry point by GetServices command, if DUT supports Access Control service. Otherwise, these test cases will be skipped.

DUT shall support GetServiceCapabilities command.

DUT shall return capabilities for each specific access point.

If DUT returns Duress capability as supported by access point, then DUT shall support the following events for this access point:

- tns:AccessControl/Duress

DUT shall support GetEventProperties command and return all supported events in TopicSet.

DUT shall support Pull Point Subscription and Topic Expression filter.

A test operator shall manually invoke the following events if required:

- tns:AccessControl/Duress

Please, refer to Section 4.10 for Duress events Test Cases.

### 3.3.11 Consistency

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

DUT shall support at least one Access Point. And also at least one Access Point shall be added and configured on device.

If DUT supports at least one Area, at least one Area shall be added and configured on device.

DUT shall support the following commands:

- GetAccessPointInfoList

- GetAreaInfoList

DUT shall not return other Area tokens, than listed in GetAreaInfoList responses, in AreaTo and AreaFrom fields of GetAccessPointInfo’s.

Please, refer to Section 4.11 for Consistency Test Cases.

### 3.3.12 Access Point Change Events

DUT shall give the Access Control Service entry point and Event Service entry point by GetServices command, if DUT supports Access Control service. Otherwise, these test cases will be skipped.

DUT shall support GetEventProperties command and return all supported events in TopicSet.

DUT shall support Pull Point Subscription and Topic Expression filter.

If DUT supports adding or changing access points, then DUT shall support the following events:

- tns1:Configuration/AccessPoint/Change

If DUT returns tns1:Configuration/AccessPoint/Change topic in GetEventProperties command, then DUT should have possibility to invoke it.

If DUT supports removing access points, then DUT shall support the following events:

- tns1:Configuration/AccessPoint/Removed

If DUT returns tns1:Configuration/AccessPoint/Removed topic in GetEventProperties command, then DUT should have possibility to invoke it.

A test operator shall manually invoke the following events if required:

- tns1:Configuration/AccessPoint/Change
- tns1:Configuration/AccessPoint/Removed

Please, refer to Section 4.12 for Access Point Change events Test Cases.

### 3.3.13 Area Change Events

DUT shall give the Access Control Service entry point and Event Service entry point by GetServices command, if DUT supports Access Control service. Otherwise, these test cases will be skipped.

DUT shall support GetEventProperties command and return all supported events in TopicSet.

DUT shall support Pull Point Subscription and Topic Expression filter.

If DUT supports adding or changing areas, then DUT shall support the following events:

- tns1:Configuration/Area/Change

If DUT returns tns1:Configuration/Area/Change topic in GetEventProperties command, then DUT should have possibility to invoke it.

If DUT supports removing areas, then DUT shall support the following events:

- tns1:Configuration/Area/Removed

If DUT returns tns1:Configuration/Area/Removed topic in GetEventProperties command, then DUT should have possibility to invoke it.

A test operator shall manually invoke the following events if required:

- tns1:Configuration/Area/Change
- tns1:Configuration/Area/Removed

Please, refer to Section 4.13 for Area Change events Test Cases.

# 4 Access Control Test Cases

## 4.1 Capabilities

### 4.1.1 ACCESS CONTROL SERVICE CAPABILITIES

**Test Case ID:** ACCESSCONTROL-1-1-1

**Specification Coverage:** Capability exchange (ONVIF Core Specification), GetServiceCapabilities (ONVIF Access Control Service Specification)

**Feature Under Test:** GetServiceCapabilities (for Access Control Service)

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify DUT Access Control Service Capabilities.

**Pre-Requisite:** Access Control Service was received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServiceCapabilities**.
4. The DUT responds with a **GetServiceCapabilitiesResponse** message with parameters
    - Capabilities =: *cap*
5. If *cap*.@AccessPointManagementSupported = true and *cap*.@MaxAccessPoints = 0, FAIL the test and skip other steps.
6. If *cap*.@AreaManagementSupported = true and *cap*.@MaxAreas = 0, FAIL the test.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

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

### 4.1.2 GET SERVICES AND GET ACCESS CONTROL SERVICE CAPABILITIES CONSISTENCY

**Test Case ID:** ACCESSCONTROL-1-1-2

**Specification Coverage:** Capability exchange (ONVIF Core Specification), GetServiceCapabilities (ONVIF Access Control Service Specification)

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

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

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

**Pre-Requisite:** Access Control Service was received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetServices** request (IncludeCapability = true) to retrieve all services of the DUT with service capabilities.
4. Verify the **GetServicesResponse** message from the DUT.
5. ONVIF Client will invoke **GetServiceCapabilities** request to retrieve DUT Access Control service capabilities.
6. Verify the **GetServiceCapabilitiesResponse** message from the DUT.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

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

- The DUT did not send valid **GetServiceCapabilitiesResponse** message.
- The DUT sent different Capabilities in **GetServicesResponse** message and in **GetServiceCapabilitiesResponse** message.

**Note:** Service will be defined as Access Controller service if it has Namespace element that is equal to “http://www.onvif.org/ver10/accesscontrol/wsdl”.

**Note:** Capabilities in **GetServicesResponse** message and in **GetServiceCapabilitiesResponse** message will be assumed as different in the following cases:

- Values of MaxLimit attribute are different.
- MaxAccessPoints attribute values are different;
- MaxAreas attribute values are different;
- MaxAccessPoints attribute is skipped only for GetServices or only for GetServiceCapabilities request;
- MaxAreas attribute is skipped only for GetServices or only for GetServiceCapabilities request;
- ClientSuppliedTokenSupported attribute values are different;
- AccessPointManagementSupported attribute values are different.
- AreaManagementSupported attribute values are different.

## 4.2 Access Point

### 4.2.1 GET ACCESS POINT INFO

**Test Case ID:** ACCESSCONTROL-2-1-1

**Specification Coverage:** GetAccessPointInfo (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAccessPointInfo

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get Access Point Info.

**Pre-Requisite:** Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point is configured and added to the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of access points from the DUT (see Annex A.1).
4. ONVIF Client will invoke **GetServiceCapabilities** request to retrieve MaxLimit capability.
5. Verify the **GetServiceCapabilitiesResponse** message from the DUT.
6. ONVIF Client will invoke **GetAccessPointInfo** request (Token list with subset of AccessPointInfo.token values from a complete list of access points at step 3 with token number equal to MaxLimit) to retrieve a subset of Access Point Information from the DUT.
7. Verify the **GetAccessPointInfoResponse** message from the DUT.
8. Verify that all requested AccessPointInfo items are in **GetAccessPointInfoResponse** message.
9. ONVIF Client will invoke **GetAccessPoint** request (Token = Token1, where Token1 is the first token from the complete list of access points at step 3) to retrieve AccessPoint Information for specified token from the DUT.
10. Verify the **GetAccessPointInfoResponse** message from the DUT.
11. Verify that **GetAccessPointInfoResponse** message contains AccessPointInfo only for specified token.
12. Repeat steps 9-11 for all other tokens from a complete list of access points at step 3.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send valid **GetAccessPointInfoResponse** message.
- The DUT returned **GetAccessPointInfoResponse** message that contains at least two AccessPointInfo items with equal tokens.

- The DUT did not return requested AccessPointInfo items in **GetAccessPointInfoResponse** message for step 7 or 10.
- The DUT returned more items than requested in **GetAccessPointInfoResponse** message for step 7 or 10.
- The DUT did not return at least one Access Point at step 3.
- The DUT did not send valid **GetServiceCapabilitiesResponse**.

**Note:** If the DUT did not return any AccessPointInfo on complete list of access points at step 3, skip steps 4-12, fail test and go to the next test.

**Note:** In case MaxLimit is equal to 0, 1 will be used instead in requests.

### 4.2.2 GET ACCESS POINT INFO WITH INVALID TOKEN

**Test Case ID:** ACCESSCONTROL-2-1-2

**Specification Coverage:** GetAccessPointInfo (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAccessPointInfo

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get Access Point Info with invalid Token.

**Pre-Requisite:** Access Control Service was received from the DUT. At least one Access Point is configured and added to the DUT if Access Point Entity is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of access points from the DUT (see Annex A.1).
4. ONVIF Client will invoke **GetServiceCapabilities** request to retrieve MaxLimit capability.
5. Verify the **GetServiceCapabilitiesResponse** message from the DUT.
6. ONVIF Client will invoke **GetAccessPointInfo** request (invalid Token).

7. Verify the **GetAccessPointInfoResponse** message from the DUT. Check that empty list was returned.

8. If DUT supports Access Point Entity feature and if MaxLimit > 1:

   - ONVIF Client will invoke **GetAccessPointInfo** request (invalid Token, valid Token).

   - Verify the **GetAccessPointInfoResponse** message from the DUT. Check that list which contains AccessPoint for valid Token only was returned.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send valid **GetAccessPointInfoResponse** message.
- The DUT returned **GetAccessPointInfoResponse** message that contains at least two AccessPointInfo items with equal tokens.
- The DUT did not send valid **GetAccessPointInfoResponse** message.
- The DUT returned **GetAccessPointInfoResponse** message that contains any AccessPointInfo items for step 7.
- The DUT returned **GetAccessPointInfoResponse** message that contains any AccessPointInfo items other than item for valid Token or does not contains AccessPointInfo item for valid token step 8.
- The DUT did not return at least one Access Point at step 3.
- The DUT did not send valid **GetServiceCapabilitiesResponse**.

**Note:** If MaxLimit is less than 2 skip step 8.

### 4.2.3 GET ACCESS POINT INFO LIST - LIMIT

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

**Specification Coverage:** GetAccessPointInfoList (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAccessPointInfoList

**WSDL Reference:** accesscontrol.wsdl

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

**Pre-Requisite:** Access Control Service was received from the DUT. At least one Access Point is configured and added to the DUT if Access Point Entity is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves complete list of access point info items by following the procedure mentioned in Annex A.1 with the following input and output parameters
   - out *accessPointInfoCompleteList* - complete access points info list
   - out *accessPointsNumber* - access points number
4. If DUT supports Access Point Entity feature and *accessPointsNumber* < = 0, FAIL the test and skip other steps.
5. If DUT does not support Access Point Entity feature and *accessPointsNumber* != 0, FAIL the test and skip other steps.
6. ONVIF Client will invoke **GetServiceCapabilities** request to retrieve MaxLimit capability.
7. Verify the **GetServiceCapabilitiesResponse** message from the DUT.
8. If MaxLimit > 0

   8.1. ONVIF Client invokes **GetAccessPointInfoList** request (Limit = 1, no StartReference) to retrieve the first item on the list of Access Point Information from the DUT.

   8.2. The DUT responds with a **GetAccessPointInfoListResponse** message with parameters
   - NextStartReference
   - List of AccessPointInfo := *accessPointInfoList1*

   8.3. If *accessPointInfoList1* contains number of items greater than 1, FAIL the test and skip other steps.

   8.4. If DUT does not support Access Point Entity and *accessPointInfoList1* is not empty, FAIL the test and skip other steps.

9. ONVIF Client invokes **GetAccessPointInfoList** request (Limit = MaxLimit, no StartReference) to retrieve the first part of Access Point Information list from the DUT and is limited by MaxLimit.

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

    - NextStartReference

    - List of AccessPointInfo := *accessPointInfoList2*

11. If *accessPointInfoList2* contains number of items greater than MaxLimit, FAIL the test and skip other steps.

12. If DUT does not support Access Point Entity and *accessPointInfoList2* is not empty, FAIL the test and skip other steps.

13. If MaxLimit > 0

    13.1. ONVIF Client invokes **GetAccessPointInfoList** request (Limit = ItemNumber, no StartReference, where ItemNumber is between 1 and minimal value of MaxLimit and total number of access points) to retrieve sublist of Access Point Information limited by ItemNumber from the DUT.

    13.2. The DUT responds with a **GetAccessPointInfoListResponse** message with parameters

    - NextStartReference

    - List of AccessPointInfo := *accessPointInfoList3*

    13.3. If *accessPointInfoList3* contains number of items greater than ItemNumber, FAIL the test and skip other steps.

    13.4. If DUT does not support Access Point Entity and *accessPointInfoList3* is not empty, FAIL the test and skip other steps.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

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

- The DUT did not return at least one Access Point on step 3.

- The DUT returned more AccessPointInfo items in **GetAccessPointInfoListResponse** message than it was specified in Limit parameter.
- The DUT returned more AccessPointInfo items in **GetAccessPointInfoListResponse** message than MaxLimit.
- The DUT did not send valid **GetServiceCapabilitiesResponse**.

**Note:** In case MaxLimit is equal to 0, 1 will be used instead for request.

### 4.2.4 GET ACCESS POINT INFO LIST - START REFERENCE AND LIMIT

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

**Specification Coverage:** GetAccessPointInfoList (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAccessPointInfoList

**WSDL Reference:** accesscontrol.wsdl

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

**Pre-Requisite:** Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point is configured and added to the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetServiceCapabilities** request to retrieve DUT access control service capabilities.
4. Verify the **GetServiceCapabilitiesResponse** (MaxLimit) message from the DUT.
5. ONVIF Client will invoke **GetAccessPointInfoList** request (Limit = MaxLimit, no StartReference) to retrieve the first part of Access Point Information list from the DUT.
6. Verify the **GetAccessPointInfoListResponse** message from the DUT. Verify that **GetAccessPointInfoListResponse** message does not contain number of items greater than Limit. Verify that **GetAccessPointInfoListResponse** message does not contain number of items greater than MaxLimit.

7. If **GetAccessPointInfoListResponse** message contains NextStartReference, repeat steps 5-6 with StartReference = [current NextStartReference]. Otherwise, skip other steps and finalize getting the complete access point list.

8. Verify that **GetAccessPointInfoListResponse** messages at step 6 does not contain AccessPointInfo items with the same tokens values all over the list, i.e. there shall be no **GetAccessPointInfoListResponse** messages containing the same value for AccessPointInfo items token.

9. ONVIF Client will invoke **GetAccessPointInfoList** request (Limit = 1, no StartReference) to retrieve the first part of Access Point Information list from the DUT.

10. Verify the **GetAccessPointInfoListResponse** message from the DUT. Verify that **GetAccessPointInfoListResponse** message does not contain number of items greater than Limit. Verify that **GetAccessPointInfoListResponse** message does not contain number of items greater than MaxLimit.

11. If **GetAccessPointInfoListResponse** message contains NextStartReference, repeat steps 9-12 with StartReference = [current NextStartReference]. Otherwise, skip other steps and finalize getting the complete access point list.

12. Verify that **GetAccessPointInfoListResponse** messages at step 10 does not contain AccessPointInfo items with the same tokens values all over the list, i.e. there shall be no **GetAccessPointInfoListResponse** messages containing the same value for AccessPointInfo items token.

13. Verify that the total list received during steps 5-7 has the same items as the list received during steps 9-11.

14. ONVIF Client will invoke **GetAccessPointInfoList** request (Limit = [limit_value], no StartReference, where limit_value is between 1 and MaxLimit) to retrieve the first part of Access Point Information list from the DUT.

15. Verify the **GetAccessPointInfoListResponse** message from the DUT. Verify that **GetAccessPointInfoListResponse** message does not contain number of items greater than Limit. Verify that **GetAccessPointInfoListResponse** message does not contain number of items greater than MaxLimit.

16. If **GetAccessPointInfoListResponse** message contains NextStartReference, repeat steps 14-17 with StartReference = [current NextStartReference]. Otherwise, skip other steps and finalize getting the complete access point list.

17. Verify that **GetAccessPointInfoListResponse** messages at step 15 does not contain AccessPointInfo items with the same tokens values all over the list, i.e. there shall

be no **GetAccessPointInfoListResponse** messages containing the same value for AccessPointInfo items token.

18. Verify that the total list received during steps 5-7 has the same items as the list received during steps 14-16.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send valid **GetAccessPointInfoListResponse** message.
- The DUT returned **GetAccessPointInfoListResponse** message that contains wrong item list than requested according to StartReference and Limit values.
- The DUT returned **GetAccessPointInfoListResponse** messages for step 6 contains at least one pair of the same tokens for AccessPointInfo item all over the messages.
- The DUT returned **GetAccessPointInfoListResponse** messages for step 10 contains at least one pair of the same tokens for AccessPointInfo item all over the messages.
- The DUT returned **GetAccessPointInfoListResponse** messages for step 15 contains at least one pair of the same tokens for AccessPointInfo item all over the messages.
- The DUT returned more AccessPointInfo items in **GetAccessPointInfoListResponse** message than it was specified in Limit parameter.
- The DUT returned more AccessPointInfo items in **GetAccessPointInfoListResponse** message than MaxLimit.
- The DUT did not send valid **GetServiceCapabilitiesResponse**.

**Note:** In case MaxLimit is equal to 0, 1 will be used instead for request.

### 4.2.5 GET ACCESS POINT INFO LIST - NO LIMIT

**Test Case ID:** ACCESSCONTROL-2-1-5

**Specification Coverage:** GetAccessPointInfoList (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAccessPointInfoList

**WSDL Reference:** accesscontrol.wsdl

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

**Pre-Requisite:** Access Control Service was received from the DUT. At least one Access Point is configured and added to the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetServiceCapabilities** request to retrieve DUT access control service capabilities.
4. Verify the **GetServiceCapabilitiesResponse** (MaxLimit) message from the DUT.
5. ONVIF Client will invoke **GetAccessPointInfoList** request (no Limit, no StartReference) to retrieve the first part of Access Point Information list from the DUT.
6. Verify the **GetAccessPointInfoListResponse** message from the DUT. Verify that **GetAccessPointInfoListResponse** message does not contain number of items greater than MaxLimit.
7. If DUT does not support Access Point Entity and AccessPointInfoList in **GetAccessPointInfoListResponse** is not empty, FAIL the test and skip other steps.
8. If **GetAccessPointInfoListResponse** message contains NextStartReference, repeat steps 5-6 with StartReference = [current NextStartReference]. Otherwise, skip other steps and finalize getting the complete access point list.
9. Verify that **GetAccessPointInfoListResponse** messages at step 6 does not contain AccessPointInfo items with the same tokens values all over the list, i.e. there shall be no **GetAccessPointInfoListResponse** messages containing the same value for AccessPointInfo items token.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send valid **GetAccessPointInfoListResponse** message.
- The DUT returned **GetAccessPointInfoListResponse** message that contains wrong item list than requested according to StartReference and Limit values.
- The DUT returned **GetAccessPointInfoListResponse** messages for step 6 contains at least one pair of the same tokens for AccessPointInfo item all over the messages.
- The DUT returned more AccessPointInfo items in **GetAccessPointInfoListResponse** message than MaxLimit.
- The DUT did not send valid **GetServiceCapabilitiesResponse**.

### 4.2.6 GET ACCESS POINT INFO - TOO MANY ITEMS

**Test Case ID:** ACCESSCONTROL-2-1-6

**Specification Coverage:** GetAccessPointInfo (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAccessPointInfo

**WSDL Reference:** accesscontrol.wsdl

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

**Pre-Requisite:** Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point is configured and added to the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of access points from the DUT (see Annex A.1).
4. ONVIF Client will invoke **GetServiceCapabilities** request to retrieve MaxLimit capability.
5. Verify the **GetServiceCapabilitiesResponse** message from the DUT.
6. If total number of access points is less than MaxLimit, skip other steps and go to the text test.
7. ONVIF Client will invoke **GetAccessPointInfo** request (Token list with subset of AccessPointInfo.token values from a complete list of access points at step 3 with token

number greater than MaxLimit) to retrieve a subset of Access Point Information from the DUT.

8. The DUT will generate SOAP 1.2 fault message (**InvalidArgs/TooManyItems**).

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send SOAP 1.2 fault message.
- The DUT sent incorrect SOAP 1.2 fault message (fault code, namespace, etc.).
- The DUT did not send valid **GetServiceCapabilitiesResponse**.
- The DUT did not return at least one Access Point on step 3.

**Note:** If the DUT did not return any AccessPointInfo on complete list of access points at step 3, skip steps 4-8, fail test and go to the next test.

### 4.2.7 GET ACCESS POINT STATE

**Test Case ID:** ACCESSCONTROL-2-1-7

**Specification Coverage:** GetAccessPointState (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAccessPointState

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get Access Point State.

**Pre-Requisite:** Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point is configured and added to the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

3. Get a complete list of access points from the DUT (see Annex A.1).
4. ONVIF Client will invoke **GetAccessPointState** request (TokenList.Token = Token1, where Token1 is the first token from the complete list of access points at step 3) to retrieve Access Point state for specified token from the DUT.
5. Verify the **GetAccessPointStateResponse** message from the DUT. Check that AccessPointState.Enabled is equal to true, if Access Point does not supports DisableAccessPoint capability, e.g. AccessPointInfo.Capabilities.DisableAccessPoint = false.
6. Repeat steps 4-5 for all other tokens from complete list of access points at step 3.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send valid **GetAccessPointStateResponse** message.
- The DUT did not return at least one Access Point on step 3.
- The DUT returned AccessPointState.Enabled = false for at least one Access Point in case AccessPointInfo.Capabilities.DisableAccessPoint = false for this Access Point.

**Note:** If the DUT did not return any AccessPointInfo on complete list of access points at step 3, skip steps 4-6, fail the test and go to the next test.

### 4.2.8 GET ACCESS POINT STATE WITH INVALID TOKEN

**Test Case ID:** ACCESSCONTROL-2-1-8

**Specification Coverage:** GetAccessPointState (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAccessPointState

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get Access Point State with invalid Token.

**Pre-Requisite:** Access Control Service address was received from the DUT. Access Point Entity is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetAccessPointState** request (invalid Token).
4. The DUT will generate SOAP 1.2 fault message (**InvalidArgVal/NotFound**).

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send SOAP 1.2 fault message.
- The DUT sent incorrect SOAP 1.2 fault message (fault code, namespace, etc.).

**Note:** Other faults than specified in the test are acceptable, but the specified are preferable.

### 4.2.9 ENABLE/DISABLE ACCESS POINT

**Test Case ID:** ACCESSCONTROL-2-1-9

**Specification Coverage:** EnableAccessPoint (ONVIF Access Control Service Specification), DisableAccessPoint (ONVIF Access Control Service Specification), GetAccessPointState (ONVIF Access Control Service Specification)

**Feature Under Test:** EnableAccessPoint, DisableAccessPoint, GetAccessPointState

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify that Enable and Disable Access Point and its State Change.

**Pre-Requisite:** Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with Enable/Disable capability equal to true is configured and added to the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.

2. Start the DUT.

3. Get complete list of access points from the DUT (see Annex A.1).

4. If there is no Access Points with DisableAccessPoint equal to true, fail the test and skip other steps.

5. If Access Point with Token1 (Token1 is the first AccessPointInfo.token from the complete list of access points at step 3) has AccessPointInfo.Capabilities.DisableAccessPoint equal to false, then skip steps 6-29 and go to the step 30.

6. ONVIF Client will invoke **GetAccessPointState** request (Token = “Token1”, where Token1 is the first AccessPointInfo.token from the complete list of access points at step 3) to get current access point state.

7. Verify the **GetAccessPointStateResponse** message from the DUT.

8. If Access Point with Token1 (Token1 is the first AccessPointState.token from the complete list of access points at step 3) has AccessPointState.Enabled equal to true, then skip steps 9-19 and go to the step 20.

9. ONVIF Client will invoke **EnableAccessPoint** request (Token = “Token1”, where Token1 is the first AccessPointInfo.token from the complete list of access points at step 3) to try enabling access point.

10. Verify the **EnableAccessPointResponse** message from the DUT.

11. ONVIF Client will invoke **GetAccessPointState** request (Token = “Token1”, where Token1 is the first AccessPointInfo.token from the complete list of access points at step 3) to get current access point state.

12. Verify the **GetAccessPointStateResponse** message from the DUT.

13. Verify that AccessPointState.Enabled is equal to true.

14. ONVIF Client will invoke **DisableAccessPoint** request (Token = “Token1”, where Token1 is the first AccessPointInfo.token from the complete list of access points at step 3) to try disabling access point.

15. Verify the **DisableAccessPointResponse** message from the DUT.

16. ONVIF Client will invoke **GetAccessPointState** request (Token = “Token1”, where Token1 is the first AccessPointInfo.token from the complete list of access points at step 3) to get current access point state.

17. Verify the **GetAccessPointStateResponse** message from the DUT.

18. Verify that AccessPointState.Enabled is present and equal to false.
19. Go to step 30.
20. ONVIF Client will invoke **DisableAccessPoint** request (Token = “Token1”, where Token1 is the first AccessPointInfo.token from the complete list of access points at step 3) to try disabling access point.
21. Verify the **DisableAccessPointResponse** message from the DUT.
22. ONVIF Client will invoke **GetAccessPointState** request (Token = “Token1”, where Token1 is the first AccessPointInfo.token from the complete list of access points at step 3) to get current access point state.
23. Verify the **GetAccessPointStateResponse** message from the DUT.
24. Verify that AccessPointState.Enabled is present and equal to false.
25. ONVIF Client will invoke **EnableAccessPoint** request (Token = “Token1”, where Token1 is the first AccessPointInfo.token from the complete list of access points at step 3) to try enabling access point.
26. Verify the **EnableAccessPointResponse** message from the DUT.
27. ONVIF Client will invoke **GetAccessPointState** request (Token = “Token1”, where Token1 is the first AccessPointInfo.token from the complete list of access points at step 3) to get current access point state.
28. Verify the **GetAccessPointStateResponse** message from the DUT.
29. Verify that AccessPointState.Enabled is equal to true or skipped.
30. Repeat steps 5-29 for all other tokens from the complete list of access points at step 3.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send valid **GetAccessPointStateResponse** message.
- The DUT did not send valid **EnableAccessPointResponse** message.
- The DUT did not send valid **DisableAccessPointResponse** message.

- The DUT did not change Access Point Enabled value to false after **DisableAccessPointResponse** message.
- The DUT did not change Access Point Enabled value to true after **EnableAccessPointResponse** message (Enabled element could be skipped in **GetAccessPointStateResponse** message for this case).
- The DUT did not return at least one Access Point at step 3.

**Note:** If the DUT did not return any AccessPointInfo on complete list of access points at step 3, skip steps 5-30, fail the test and go to the next test.

### 4.2.10 ENABLE/DISABLE ACCESS POINT - COMMAND NOT SUPPORTED

**Test Case ID:** ACCESSCONTROL-2-1-10

**Specification Coverage:** EnableAccessPoint (ONVIF Access Control Service Specification), DisableAccessPoint (ONVIF Access Control Service Specification)

**Feature Under Test:** EnableAccessPoint, DisableAccessPoint

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify that Enable and Disable Access Point in case Door does not support this capability.

**Pre-Requisite:** Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point is configured and added to the DUT. At least one Access Point with Enable/Disable capability equal to false is configured and added to the DUT, if possible.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get a complete list of access points from the DUT (see Annex A.1).
4. If Access Point with Token1 (Token1 is the first AccessPointInfo.token from the complete list of access points at step 3) has AccessPointInfo.Capabilities.DisableAccessPoint equal to true or skipped, then skip steps 5-8 and go to the step 9.

5. ONVIF Client will invoke **EnableAccessPoint** request (Token = “Token1”, where Token1 is the first AccessPointInfo.token from the complete list of access points at step 3) to try enabling access point.

6. The DUT will generate SOAP 1.2 fault message (**ActionNotSupported/NotSupported**).

7. ONVIF Client will invoke **DisableAccessPoint** request ( “Token1”, where Token1 is the first AccessPointInfo.token from the complete list of access points at step 3) to try disabling access point.

8. The DUT will generate SOAP 1.2 fault message (**ActionNotSupported/NotSupported**).

9. Repeat steps 4-8 for all other tokens from the complete list of access points at step 3.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send SOAP 1.2 fault message.
- The DUT sent incorrect SOAP 1.2 fault message (fault code, namespace, etc.).
- The DUT did not return at least one Access Point on step 3.

**Note:** If the DUT did not return any AccessPointInfo on complete list of access points at step 3, skip steps 4-9, fail the test and go to the next test.

**Note:** Other faults than specified in the test are acceptable, but the specified are preferable.

### 4.2.11 ENABLE ACCESS POINT WITH INVALID TOKEN

**Test Case ID:** ACCESSCONTROL-2-1-11

**Specification Coverage:** EnableAccessPoint (ONVIF Access Control Service Specification)

**Feature Under Test:** EnableAccessPoint

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Enable Access Point with invalid Token.

**Pre-Requisite:** Access Control Service address was received from the DUT. Access Point Entity is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **EnableAccessPoint** request (invalid Token).
4. The DUT will generate SOAP 1.2 fault message (**InvalidArgVal/NotFound**).

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send SOAP 1.2 fault message.
- The DUT sent incorrect SOAP 1.2 fault message (fault code, namespace, etc.).

**Note:** Other faults than specified in the test are acceptable, but the specified are preferable.

### 4.2.12 DISABLE ACCESS POINT WITH INVALID TOKEN

**Test Case ID:** ACCESSCONTROL-2-1-12

**Specification Coverage:** DisableAccessPoint (ONVIF Access Control Service Specification)

**Feature Under Test:** DisableAccessPoint

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Disable Access Point with invalid Token.

**Pre-Requisite:** Access Control Service address was received from the DUT. Access Point Entity is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client will invoke **DisableAccessPoint** request (invalid Token).

4. The DUT will generate SOAP 1.2 fault message (**InvalidArgVal/NotFound**).

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send SOAP 1.2 fault message.

- The DUT sent incorrect SOAP 1.2 fault message (fault code, namespace, etc.).

**Note:** Other faults than specified in the test are acceptable, but the specified are preferable.

## 4.3 Area

### 4.3.1 GET AREA INFO

**Test Case ID:** ACCESSCONTROL-3-1-1

**Specification Coverage:** GetAreaInfo (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAreaInfo

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get Area Info.

**Pre-Requisite:** Access Control Service was received from the DUT. At least one Area is configured and added to the DUT, if the DUT allows this.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.

2. Start the DUT.

3. Get the complete list of areas from the DUT (see Annex A.2).

4. ONVIF Client will invoke **GetServiceCapabilities** request to retrieve MaxLimit capability.
5. Verify the **GetServiceCapabilitiesResponse** message from the DUT.
6. ONVIF Client will invoke **GetAreaInfo** request (Token list with subset of AreaInfo.token values from the complete list of areas at step 3 with token number equal to MaxLimit) to retrieve subset of Area Information from the DUT.
7. Verify the **GetAreaInfoResponse** message from the DUT.
8. Verify that all requested AreaInfo items are in **GetAreaInfoResponse** message.
9. ONVIF Client will invoke **GetAreaInfo** request (Token = Token1, where Token1 is the first token from the complete list of areas at step 3) to retrieve Area Information for specified token from the DUT.
10. Verify the **GetAreaInfoResponse** message from the DUT.
11. Verify that **GetAreaInfoResponse** message contains AreaInfo only for specified token.
12. Repeat steps 9-11 for all other tokens from complete list of areas at step 3.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send valid **GetAreaInfoResponse** message.
- The DUT returned **GetAreaInfoResponse** message that contains at least two AreaInfo items with equal tokens.
- The DUT did not return requested AreaInfo items in **GetAreaInfoResponse** message for step 7 or 10.
- The DUT returned more items than requested in **GetAreaInfoResponse** message for step 7 or 10.
- The DUT did not send valid **GetServiceCapabilitiesResponse**.

**Note:** If the DUT did not return any AreaInfo on complete list of areas at step 3, skip steps 4-12 and go to the next test.

**Note:** In case MaxLimit is equal to 0, 1 will be used instead in requests.

### 4.3.2 GET AREA INFO WITH INVALID TOKEN

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

**Specification Coverage:** GetAreaInfo (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAreaInfo

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get Area Info with invalid Token.

**Pre-Requisite:** Access Control Service was received from the DUT. At least one Area is configured and added to the DUT, if the DUT allows this.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of areas from the DUT (see Annex A.2).
4. ONVIF Client will invoke **GetServiceCapabilities** request to retrieve MaxLimit capability.
5. Verify the **GetServiceCapabilitiesResponse** message from the DUT.
6. ONVIF Client will invoke **GetAreaInfo** request (invalid Token).
7. Verify the **GetAreaInfoResponse** message from the DUT. Check that empty list was returned.
8. ONVIF Client will invoke **GetAreaInfo** request (invalid Token, valid Token).
9. Verify the **GetAreaInfoResponse** message from the DUT. Check that list which contains Area for valid Token only was returned.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send valid **GetAreaInfoResponse** message.

- The DUT returned **GetAreaInfoResponse** message that contains at least two AreaInfo items with equal tokens.
- The DUT did not send valid **GetAreaInfoResponse** message.
- The DUT returned **GetAreaInfoResponse** message that contains any AreaInfo items for step 7.
- The DUT returned **GetAreaInfoResponse** message that contains any AreaInfo items other than item for valid Token or does not contains AreaInfo item for valid token step 9.
- The DUT did not send valid **GetServiceCapabilitiesResponse**.

**Note:** If MaxLimit is less than 2, skip steps 8-9.

**Note:** If the DUT did not return any AreaInfo on complete list of areas at step 3, skip steps 8-9 and go to the next test.

### 4.3.3 GET AREA INFO LIST - LIMIT

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

**Specification Coverage:** GetAreaInfoList (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAreaInfoList

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get Area Info List using Limit.

**Pre-Requisite:** Access Control Service was received from the DUT. At least one Area is configured and added to the DUT, if the DUT allows this.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get a complete list of areas from the DUT (see Annex A.2)
4. ONVIF Client will invoke **GetServiceCapabilities** request to retrieve MaxLimit capability.
5. Verify the **GetServiceCapabilitiesResponse** message from the DUT.

6. ONVIF Client will invoke **GetAreaInfoList** request (Limit = 1, no StartReference) to retrieve the first item on the list of Area Information from the DUT.

7. Verify the **GetAreaInfoListResponse** message from the DUT.

8. Verify that **GetAreaInfoListResponse** message at step 7 does not contain number of items greater than 1. Verify that **GetAreaInfoListResponse** message at step 7 does not contain number of items greater than MaxLimit.

9. ONVIF Client will invoke **GetAreaInfoList** request (Limit = MaxLimit, no StartReference) to retrieve the first part list of Area Information from the DUT and is limited by MaxLimit.

10. Verify the **GetAreaInfoListResponse** message from the DUT.

11. Verify that **GetAreaInfoListResponse** message at step 10 does not contain number of items greater than MaxLimit.

12. ONVIF Client will invoke **GetAreaInfoList** request (Limit = ItemNumber, no StartReference, where ItemNumber between 1 and minimal value between MaxLimit and total number of areas) to retrieve sublist of Area Information, and is limited by ItemNumber from the DUT.

13. Verify the **GetAreaInfoListResponse** message from the DUT.

14. Verify that **GetAreaInfoListResponse** message at step 13 does not contain number of items greater than ItemNumber. Verify that **GetAreaInfoListResponse** message at step 13 does not contain number of items greater than MaxLimit.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send valid **GetAreaInfoListResponse** message.

- The DUT returned **GetAreaInfoListResponse** message that contains wrong item list than requested according to Limit value and StartReference value.

- The DUT returned more AreaInfo items in **GetAreaInfoListResponse** message than it was specified in Limit parameter.

- The DUT returned more AreaInfo items in **GetAreaInfoListResponse** message than MaxLimit.

- The DUT did not send valid **GetServiceCapabilitiesResponse**.

**Note:** If the DUT did not return any AreaInfo on complete list of areas at step 3, skip steps 4-14 and go to the next test.

**Note:** In case MaxLimit is equal to 0, 1 will be used instead in requests.

### 4.3.4 GET AREA INFO LIST - START REFERENCE AND LIMIT

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

**Specification Coverage:** GetAreaInfoList (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAreaInfoList

**WSDL Reference:** accesscontrol.wsdl

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

**Pre-Requisite:** Access Control Service was received from the DUT. At least one Area is configured and added to the DUT, if the DUT allows this.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetServiceCapabilities** request to retrieve area control service capabilities of the DUT.
4. Verify the **GetServiceCapabilitiesResponse** (MaxLimit) message from the DUT.
5. ONVIF Client will invoke **GetAreaInfoList** request (Limit = MaxLimit, no StartReference) to retrieve the first part of Area Information list from the DUT.
6. Verify the **GetAreaInfoListResponse** message from the DUT. Verify that **GetAreaInfoListResponse** message does not contain number of items greater than Limit. Verify that **GetAreaInfoListResponse** message does not contain number of items greater than MaxLimit.
7. If **GetAreaInfoListResponse** message contains NextStartReference, repeat steps 5-6 with StartReference = [current NextStartReference]. Otherwise, skip other steps and finalize getting complete area list.

8. Verify that **GetAreaInfoListResponse** messages at step 6 does not contain AreaInfo items with the same tokens values all over the list, i.e. there shall be no **GetAreaInfoListResponse** messages containing the same value for AreaInfo items token.

9. ONVIF Client will invoke **GetAreaInfoList** request (Limit = 1, no StartReference) to retrieve the first part of Area Information list from the DUT.

10. Verify the **GetAreaInfoListResponse** message from the DUT. Verify that **GetAreaInfoListResponse** message does not contain number of items greater than Limit. Verify that **GetAreaInfoListResponse** message does not contain number of items greater than MaxLimit.

11. If **GetAreaInfoListResponse** message contains NextStartReference, repeat steps 5-6 with StartReference = [current NextStartReference]. Otherwise, skip other steps and finalize getting complete area list.

12. Verify that **GetAreaInfoListResponse** messages at step 10 does not contain AreaInfo items with the same tokens values all over the list, i.e. there shall be no **GetAreaInfoListResponse** messages containing the same value for AreaInfo items token.

13. Verify that total list received during steps 5-7 has the same items as the list received during steps 9-11.

14. ONVIF Client will invoke **GetAreaInfoList** request (Limit = [limit_value], no StartReference, where limit_value is between 1 and MaxLimit) to retrieve the first part of Area Information list from the DUT.

15. Verify the **GetAreaInfoListResponse** message from the DUT. Verify that **GetAreaInfoListResponse** message does not contain number of items greater than Limit. Verify that **GetAreaInfoListResponse** message does not contain number of items greater than MaxLimit.

16. If **GetAreaInfoListResponse** message contains NextStartReference, repeat steps 5-6 with StartReference = [current NextStartReference]. Otherwise, skip other steps and finalize getting complete area list.

17. Verify that **GetAreaInfoListResponse** messages at step 15 does not contain AreaInfo items with the same tokens values all over the list, i.e. there shall be no **GetAreaInfoListResponse** messages containing the same value for AreaInfo items token.

18. Verify that total list received during steps 5-7 has the same items as the list received during steps 14-16.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send valid **GetAreaInfoListResponse** message.
- The DUT returned **GetAreaInfoListResponse** message that contains wrong item list than requested according to Offset and StartReference values.
- The DUT returned **GetAreaInfoListResponse** messages for step 6 contains at least one pair of the same tokens for AreaInfo item all over the messages.
- The DUT returned **GetAreaInfoListResponse** messages for step 10 contains at least one pair of the same tokens for AreaInfo item all over the messages.
- The DUT returned **GetAreaInfoListResponse** messages for step 15 contains at least one pair of the same tokens for AreaInfo item all over the messages.
- The DUT returned more AreaInfo items in **GetAreaInfoListResponse** message than it was specified in Limit parameter.
- The DUT did not send valid **GetServiceCapabilitiesResponse**.

**Note:** In case MaxLimit is equal to 0, 1 will be used instead in requests.

### 4.3.5 GET AREA INFO - TOO MANY ITEMS

**Test Case ID:** ACCESSCONTROL-3-1-9

**Specification Coverage:** GetAreaInfo (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAreaInfo

**WSDL Reference:** accesscontrol.wsdl

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

**Pre-Requisite:** Access Control Service was received from the DUT. At least one Area is configured and added to the DUT, if the DUT allows this.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.

2. Start the DUT.

3. Get the complete list of areas from the DUT (see Annex A.2).

4. ONVIF Client will invoke **GetServiceCapabilities** request to retrieve MaxLimit capability.

5. Verify the **GetServiceCapabilitiesResponse** message from the DUT.

6. If total number of areas is less than MaxLimit, skip other steps and go to the text test.

7. ONVIF Client will invoke **GetAreaInfo** request (Token list with subset of AreaInfo.token values from a complete list of areas at step 3 with token number greater than MaxLimit) to retrieve a subset of Area Information from the DUT.

8. The DUT will generate SOAP 1.2 fault message (**InvalidArgs/TooManyItems**).

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send SOAP 1.2 fault message.

- The DUT sent incorrect SOAP 1.2 fault message (fault code, namespace, etc.).

- The DUT did not send valid **GetServiceCapabilitiesResponse**.

**Note:** If the DUT did not return any AreaInfo on complete list of areas at step 3, skip steps 4-8 and go to the next test.

### 4.3.6 GET AREA INFO LIST - NO LIMIT

**Test Case ID:** ACCESSCONTROL-3-1-10

**Specification Coverage:** GetAreaInfoList (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAreaInfoList

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get Area Info List without using Limit.

**Pre-Requisite:** Access Control Service was received from the DUT. At least one Area is configured and added to the DUT, if the DUT allows this.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetServiceCapabilities** request to retrieve DUT access control service capabilities.
4. Verify the **GetServiceCapabilitiesResponse** (MaxLimit) message from the DUT.
5. ONVIF Client will invoke **GetAreaInfoList** request (no Limit, no StartReference) to retrieve the first part of Area Information list from the DUT.
6. Verify the **GetAreaInfoListResponse** message from the DUT. Verify that **GetAreaInfoListResponse** message does not contain number of items greater than MaxLimit.
7. If **GetAreaInfoListResponse** message contains NextStartReference, repeat steps 5-6 with StartReference = [current NextStartReference]. Otherwise, skip other steps and finalize getting the complete area list.
8. Verify that **GetAreaInfoListResponse** messages at step 6 does not contain AreaInfo items with the same tokens values all over the list, i.e. there shall be no **GetAreaInfoListResponse** messages containing the same value for AreaInfo items token.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send valid **GetAreaInfoListResponse** message.
- The DUT returned **GetAreaInfoListResponse** message that contains wrong item list than requested according to StartReference and Limit values.
- The DUT returned **GetAreaInfoListResponse** messages for step 6 contains at least one pair of the same tokens for AreaInfo item all over the messages.

- The DUT returned more AreaInfo items in **GetAreaInfoListResponse** message than MaxLimit.
- The DUT did not send valid **GetServiceCapabilitiesResponse**.

## 4.4 External Authorization

### 4.4.1 ACCESS CONTROL - ACCESS GRANTED TO ANONYMOUS (EXTERNAL AUTHORIZATION)

**Test Case ID:** ACCESSCONTROL-11-1-1

**Specification Coverage:** tns1:AccessControl/AccessGranted/Anonymous (Access Control Service Specification), tns1:AccessControl/Request/Anonymous (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification), ExternalAuthorization (ONVIF Access Control Service Specification)

**Feature Under Test:** ExternalAuthorization, tns1:AccessControl/AccessGranted/Anonymous, tns1:AccessControl/Request/Anonymous.

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

**Test Purpose:** To verify Access Granted to Anonymous scenario in the case of External Authorization.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with External Authorization capability and Anonymous Access capability is configured and added to the DUT. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of access points from the DUT (see Annex A.1).
4. If there are no Access Points with Capabilities.AnonymousAccess = "true" and Capabilities.ExternalAuthorization = "true" in complete list of access points, fail the test and skip other steps.

5. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
6. Verify the **GetEventPropertiesResponse** message from the DUT.
7. Check if there is an event with Topic tns1:AccessControl/AccessGranted/Anonymous. If there is no event with such Topic skip other steps, fail the test and go to the next test.
8. Check that this event is not a Property event (MessageDescription.IsProperty = "false").
9. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".
10. Check that this event contains Data.SimpleItemDescription item with Name = "External" and Type = "xs:boolean".
11. Check if there is an event with Topic tns1:AccessControl/Request/Anonymous. If there is no event with such Topic skip other steps, fail the test and go to the next test.
12. Check that this event is not a Property event (MessageDescription.IsProperty = "false").
13. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".
14. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:AccessControl/Request/Anonymous Topic and tns1:AccessControl/AccessGranted/Anonymous Topic as Filter and an InitialTerminationTime of timeout1.
15. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
16. Test Operator will invoke tns1:AccessControl/Request/Anonymous event for any Access Points with Capabilities.AnonymousAccess = "true" and Capabilities.ExternalAuthorization = "true".
17. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.
18. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 17.
19. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).
20. Verify that TopicExpression is equal to tns1:AccessControl/Request/Anonymous for the Notification message.

21. Verify that the Notification message contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to one of existing Access Point Tokens Capabilities.AnonymousAccess = "true" and Capabilities.ExternalAuthorization = "true" (e.g. complete list of access points contains Access Point with the same token and this Access Point has Capabilities.AnonymousAccess = "true" and Capabilities.ExternalAuthorization = "true").

22. ONVIF Client will invoke **ExternalAuthorization** request (AccessPointToken from Notification message, no CredentialToken, Reason = "Test Access Granted", Decision = "Granted") to Grant Access for Anonymous.

23. Verify that the DUT sends **ExternalAuthorizationResponse** message.

24. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

25. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 24.

26. Verify that TopicExpression is equal to tns1:AccessControl/AccessGranted/Anonymous for the Notification message.

27. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

28. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to Access Point Token from tns1:AccessControl/Request/Anonymous notification message.

29. Verify that the notification contains Data.SimpleItem item with Name = "External" and its Value is equal to “true”.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **ExternalAuthorizationResponse** message.

- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/Request/Anonymous event at step 18.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1: tns1:AccessControl/AccessGranted/Anonymous event at step 25.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid AccessPointToken value, invalid External value for tns1:AccessControl/AccessGranted/Anonymous notification message).
- The DUT did not return at least one Access Point on step 3.
- The DUT did not return valid Topic tns1:AccessControl/AccessGranted/Anonymous in **GetEventPropertiesResponse**.
- The DUT did not return valid Topic tns1:AccessControl/Request/Anonymous in **GetEventPropertiesResponse**.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** Test will be failed, if no Notification message is received for step 18 or 25 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

### 4.4.2 ACCESS CONTROL - ACCESS DENIED TO ANONYMOUS (EXTERNAL AUTHORIZATION)

**Test Case ID:** ACCESSCONTROL-11-1-2

**Specification Coverage:** tns1:AccessControl/Denied/Anonymous (Access Control Service Specification), tns1:AccessControl/Request/Anonymous (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification), ExternalAuthorization (ONVIF Access Control Service Specification)

**Feature Under Test:** ExternalAuthorization, tns1:AccessControl/Denied/Anonymous, tns1:AccessControl/Request/Anonymous.

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

**Test Purpose:** To verify Access Denied to Anonymous scenario in the case of External Authorization.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with External Authorization capability and Anonymous Access capability is configured and added to the DUT. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of access points from the DUT (see Annex A.1).
4. If there are no Access Points with Capabilities.AnonymousAccess = "true" and Capabilities.ExternalAuthorization = "true" in complete list of access points, fail the test and skip other steps.
5. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
6. Verify the **GetEventPropertiesResponse** message from the DUT.
7. Check if there is an event with Topic tns1:AccessControl/Denied/Anonymous. If there is no event with such Topic skip other steps, fail the test and go to the next test.
8. Check that this event is not a Property event (MessageDescription.IsProperty = "false").
9. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".
10. Check that this event contains Data.SimpleItemDescription item with Name = "External" and Type = "xs:boolean".
11. Check that this event contains Data.SimpleItemDescription item with Name = "Reason" and Type = " xs:string".

12. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:AccessControl/Request/Anonymous Topic and tns1:AccessControl/Denied/Anonymous Topic as Filter and an InitialTerminationTime of timeout1.

13. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.

14. Test Operator will invoke tns1:AccessControl/Request/Anonymous event for any Access Points with Capabilities.AnonymousAccess = "true" and Capabilities.ExternalAuthorization = "true".

15. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

16. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 15.

17. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

18. Verify that TopicExpression is equal to tns1:AccessControl/Request/Anonymous for the Notification message.

19. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to one of existing Access Point Tokens Capabilities.AnonymousAccess = "true" and Capabilities.ExternalAuthorization = "true" (e.g. complete list of access points contains Access Point with the same token and this Access Point has Capabilities.AnonymousAccess = "true" and Capabilities.ExternalAuthorization = "true").

20. ONVIF Client will invoke **ExternalAuthorization** request (AccessPointToken from Notification message, no CredentialToken, Reason = "Other", Decision = "Denied") to Deny Access for Anonymous.

21. Verify that the DUT sends **ExternalAuthorizationResponse** message.

22. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

23. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 22.

24. Verify that TopicExpression is equal to tns1:AccessControl/Denied/Anonymous for the Notification message.

25. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

26. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to Access Point Token from tns1:AccessControl/Request/Anonymous notification message.

27. Verify that the notification contains Data.SimpleItem item with Name = "External" and its Value is equal to “true”.

28. Verify that the notification contains Data.SimpleItem item with Name = "Reason" contains Value with type is equal to xs:string.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **ExternalAuthorizationResponse** message.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/Request/Anonymous event at step 16.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1: tns1:AccessControl/Denied/Anonymous event at step 23.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid AccessPointToken value invalid External or Reason value for tns1:AccessControl/Denied/Anonymous notification message).
- The DUT did not return at least one Access Point on step 3.
- The DUT did not return valid Topic tns1:AccessControl/Denied/Anonymous in **GetEventPropertiesResponse**.
- The DUT did not return valid Topic tns1:AccessControl/Request/Anonymous in **GetEventPropertiesResponse**.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** Test will be failed, if no Notification message is received for step 16 or 23 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

### 4.4.3 ACCESS CONTROL - ACCESS TIMEOUT TO ANONYMOUS (EXTERNAL AUTHORIZATION)

**Test Case ID:** ACCESSCONTROL-11-1-3

**Specification Coverage:** tns1:AccessControl/Request/Timeout (Access Control Service Specification), tns1:AccessControl/Request/Anonymous (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification), ExternalAuthorization (ONVIF Access Control Service Specification)

**Feature Under Test:** ExternalAuthorization, tns1:AccessControl/Request/Timeout, tns1:AccessControl/Request/Anonymous.

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

**Test Purpose:** To verify Access Timeout to Anonymous scenario in the case of External Authorization.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with External Authorization capability and Anonymous Access capability is configured and added to the DUT. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.

2. Start the DUT.

3. Get complete list of access points from the DUT (see Annex A.1).

4. If there are no Access Points with Capabilities.AnonymousAccess = "true" and Capabilities.ExternalAuthorization = "true" in complete list of access points, fail the test and skip other steps.

5. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.

6. Verify the **GetEventPropertiesResponse** message from the DUT.

7. Check if there is an event with Topic tns1:AccessControl/Request/Timeout. If there is no event with such Topic skip other steps, fail the test and go to the next test.

8. Check that this event is not a Property event (MessageDescription.IsProperty = "false").

9. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".

10. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:AccessControl/Request/Anonymous Topic and tns1:AccessControl/Request/Timeout Topic as Filter and an InitialTerminationTime of timeout1.

11. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.

12. Test Operator will invoke tns1:AccessControl/Request/Anonymous event for any Access Points with Capabilities.AnonymousAccess = "true" and Capabilities.ExternalAuthorization = "true".

13. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

14. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 13.

15. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

16. Verify that TopicExpression is equal to tns1:AccessControl/Request/Anonymous for the Notification message.

17. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to one of existing Access Point Tokens Capabilities.AnonymousAccess = "true" and Capabilities.ExternalAuthorization = "true" (e.g. complete list of access points contains Access Point with the same token and this Access Point has Capabilities.AnonymousAccess = "true" and Capabilities.ExternalAuthorization = "true").

18. ONVIF Client waits for timeout to initiate tns1:AccessControl/Request/Timeout event.

19. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

20. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 19.

21. Verify that TopicExpression is equal to tns1:AccessControl/Request/Timeout for the Notification message.

22. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

23. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to Access Point Token from tns1:AccessControl/Request/Anonymous notification message.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/Request/Anonymous event at step 14.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1: tns1:AccessControl/Denied/Anonymous event at step 20.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid AccessPointToken value).
- The DUT did not return at least one Access Point on step 3.
- The DUT did not return valid Topic tns1:AccessControl/Request/Timeout in **GetEventPropertiesResponse**.
- The DUT did not return valid Topic tns1:AccessControl/Request/Anonymous in **GetEventPropertiesResponse**.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** Test will be failed, if no Notification message is received for step 16 or 23 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

### 4.4.4 ACCESS CONTROL - ACCESS GRANTED WITH CREDENTIAL (EXTERNAL AUTHORIZATION)

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

**Specification Coverage:** tns1:AccessControl/AccessGranted/Credential (Access Control Service Specification), tns1:AccessControl/Request/Credential (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification), ExternalAuthorization (ONVIF Access Control Service Specification)

**Feature Under Test:** ExternalAuthorization, tns1:AccessControl/AccessGranted/Credential, tns1:AccessControl/Request/Credential.

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

**Test Purpose:** To verify Access Granted with Credential scenario in the case of External Authorization.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with External Authorization capability is configured and added to the DUT. Device supports Pull-Point Notification feature. Device supports tns1:AccessControl/AccessGranted/Credential. Device supports tns1:AccessControl/Request/Credential.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of access points from the DUT (see Annex A.1).
4. If there are no Access Point with Capabilities.ExternalAuthorization = "true" in complete list of access points, fail the test and skip other steps.
5. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
6. Verify the **GetEventPropertiesResponse** message from the DUT.
7. Check if there is an event with Topic tns1:AccessControl/AccessGranted/Credential. If there is no event with such Topic skip other steps, fail the test and go to the next test.
8. Check that this event is not a Property event (MessageDescription.IsProperty = "false").
9. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".
10. Check that this event contains Data.SimpleItemDescription item with Name = "CredentialToken" and Type = "pt:ReferenceToken".
11. Check that if this event contains Data.SimpleItemDescription item with Name = "CredentialHolderName", than it has Type = " xs:string".
12. Check that this event contains Data.SimpleItemDescription item with Name = "External" and Type = "xs:boolean".
13. Check if there is an event with Topic tns1:AccessControl/Request/Credential. If there is no event with such Topic skip other steps, fail the test and go to the next test.
14. Check that this event is not a Property event (MessageDescription.IsProperty = "false").
15. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".
16. Check that this event contains Data.SimpleItemDescription item with Name = "CredentialToken" and Type = "pt:ReferenceToken".
17. Check that if this event contains Data.SimpleItemDescription item with Name = "CredentialHolderName", than it has Type = "xs:string".
18. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:AccessControl/Request/Credential Topic and tns1:AccessControl/AccessGranted/Credential Topic as Filter and an InitialTerminationTime of timeout1.

19. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
20. Test Operator will invoke tns1:AccessControl/Request/Credential event for any Access Points with Capabilities.ExternalAuthorization = "true".
21. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.
22. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 21.
23. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).
24. Verify that TopicExpression is equal to tns1:AccessControl/Request/Credential for the Notification message.
25. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to one of existing Access Point Tokens Capabilities.ExternalAuthorization = "true" (e.g. complete list of access points contains Access Point with the same token and this Access Point has Capabilities.ExternalAuthorization = "true").
26. Verify that the notification contains Data.SimpleItem item with Name = "CredentialToken" and Value with type is equal to pt:ReferenceToken.
27. Verify that the notification which contains Data.SimpleItem item with Name = "CredentialHolderName" contains Value with type is equal to xs:string.
28. ONVIF Client will invoke **ExternalAuthorization** request (AccessPointToken from Notification message, CredentialToken from Notification message, Reason = "Test Access Granted", Decision = "Granted") to Grant Access with Credential.
29. Verify that the DUT sends **ExternalAuthorizationResponse** message.
30. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.
31. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 30.
32. Verify that TopicExpression is equal to tns1:AccessControl/AccessGranted/Credential for the Notification message.
33. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

34. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to Access Point Token from tns1:AccessControl/Request/Credential notification message.

35. Verify that the notification contains Data.SimpleItem item with Name = "CredentialToken" and Value with type is equal to pt:ReferenceToken. Verify that this value is equal to the same Data.SimpleItem from tns1:AccessControl/Request/Credential notification.

36. Verify that the notification which contains Data.SimpleItem item with Name = "CredentialHolderName" contains Value with type is equal to xs:string. Verify that this value is equal to the same Data.SimpleItem from tns1:AccessControl/Request/Credential notification.

37. Verify that the notification contains Data.SimpleItem item with Name = "External" and its Value is equal to “true”.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **ExternalAuthorizationResponse** message.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/Request/Credential event at step 22.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1: tns1:AccessControl/AccessGranted/Anonymous event at step 31.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid AccessPointToken value, invalid CredentialToken, invalid CredentialHolderName, invalid External value for tns1:AccessControl/AccessGranted/Credential notification message).
- The DUT did not return at least one Access Point on step 3.
- The DUT did not return valid Topic tns1:AccessControl/AccessGranted/Credential in **GetEventPropertiesResponse**.

- The DUT did not return valid Topic tns1:AccessControl/Request/Credential in **GetEventPropertiesResponse**.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** Test will be failed, if no Notification message is received for step 22 or 31 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

### 4.4.5 ACCESS CONTROL - ACCESS DENIED WITH CREDENTIAL (EXTERNAL AUTHORIZATION)

**Test Case ID:** ACCESSCONTROL-11-1-5

**Specification Coverage:** tns1:AccessControl/Denied/Credential (Access Control Service Specification), tns1:AccessControl/Request/Credential (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification), ExternalAuthorization (ONVIF Access Control Service Specification)

**Feature Under Test:** ExternalAuthorization, tns1:AccessControl/Denied/Credential, tns1:AccessControl/Request/Credential.

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

**Test Purpose:** To verify Access Denied with Credential scenario in the case of External Authorization.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with External Authorization capability is configured and added to the DUT. Device supports tns1:AccessControl/Denied/Credential. Device supports tns1:AccessControl/Request/Credential. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of access points from the DUT (see Annex A.1).
4. If there are no Access Point with and Capabilities.ExternalAuthorization = "true" in complete list of access points, fail the test and skip other steps.
5. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
6. Verify the **GetEventPropertiesResponse** message from the DUT.
7. Check if there is an event with Topic tns1:AccessControl/Denied/Credential. If there is no event with such Topic skip other steps, fail the test and go to the next test.
8. Check that this event is not a Property event (MessageDescription.IsProperty = "false").
9. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".
10. Check that this event contains Data.SimpleItemDescription item with Name = "CredentialToken" and Type = "pt:ReferenceToken".
11. Check that if this event contains Data.SimpleItemDescription item with Name = "CredentialHolderName", than it has Type = "xs:string".
12. Check that this event contains Data.SimpleItemDescription item with Name = "External" and Type = "xs:boolean".
13. Check that this event contains Data.SimpleItemDescription item with Name = "Reason" and Type = "xs:string".
14. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:AccessControl/Request/Credential Topic and tns1:AccessControl/Denied/Credential Topic as Filter and an InitialTerminationTime of timeout1.
15. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
16. Test Operator will invoke tns1:AccessControl/Request/Credential event for any Access Points with Capabilities.ExternalAuthorization = "true".

17. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

18. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 17.

19. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

20. Verify that TopicExpression is equal to tns1:AccessControl/Request/Credential for the Notification message.

21. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to one of existing Access Point Tokens Capabilities.ExternalAuthorization = "true" (e.g. complete list of access points contains Access Point with the same token and this Access Point has Capabilities.ExternalAuthorization = "true").

22. Verify that the notification contains Data.SimpleItem item with Name = "CredentialToken" and Value with type is equal to pt:ReferenceToken.

23. Verify that the notification which contains Data.SimpleItem item with Name = "CredentialHolderName" contains Value with type is equal to xs:string.

24. ONVIF Client will invoke **ExternalAuthorization** request (AccessPointToken from Notification message, CredentialToken from Notification message, Reason = "Other", Decision = "Denied") to Deny Access with Credential.

25. Verify that the DUT sends **ExternalAuthorizationResponse** message.

26. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

27. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 26.

28. Verify that TopicExpression is equal to tns1:AccessControl/Denied/Credential for the Notification message.

29. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

30. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to Access Point Token from tns1:AccessControl/Request/Credential notification message.

31. Verify that the notification contains Data.SimpleItem item with Name = «CredentialToken» and Value with type is equal to pt:ReferenceToken. Verify that this value is equal to the same Data.SimpleItem from tns1:AccessControl/Request/Credential notification.

32. Verify that the notification which contains Data.SimpleItem item with Name = "CredentialHolderName" contains Value with type is equal to xs:string. Verify that this value is equal to the same Data.SimpleItem from tns1:AccessControl/Request/Credential notification.

33. Verify that the notification contains Data.SimpleItem item with Name = "External" and its Value is equal to “true”.

34. Verify that the notification contains Data.SimpleItem item with Name = "Reason" contains Value with type is equal to xs:string.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **ExternalAuthorizationResponse** message.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/Request/Credential event at step 18.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1: tns1:AccessControl/Denied/Anonymous event at step 27.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid AccessPointToken value, invalid CredentialToken, invalid CredentialHolderName, invalid External value for tns1:AccessControl/Denied/Credential notification message).
- The DUT did not return at least one Access Point on step 3.
- The DUT did not return valid Topic tns1:AccessControl/Denied/Credential in **GetEventPropertiesResponse**.
- The DUT did not return valid Topic tns1:AccessControl/Request/Credential in **GetEventPropertiesResponse**.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** Test will be failed, if no Notification message is received for at 18 or 27 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

### 4.4.6 ACCESS CONTROL - ACCESS TIMEOUT WITH CREDENTIAL (EXTERNAL AUTHORIZATION)

**Test Case ID:** ACCESSCONTROL-11-1-6

**Specification Coverage:** tns1:AccessControl/Request/Timeout (Access Control Service Specification), tns1:AccessControl/Request/Credential (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification)

**Feature Under Test:** ExternalAuthorization, tns1:AccessControl/Request/Timeout, tns1:AccessControl/Request/Credential.

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

**Test Purpose:** To verify Access Timeout with Credential scenario in the case of External Authorization.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with External Authorization capability is configured and added to the DUT. Device supports Pull-Point Notification feature. Device supports tns1:AccessControl/Request/Credential.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.

2. Start the DUT.
3. Get complete list of access points from the DUT (see Annex A.1).
4. If there are no Access Point with Capabilities.ExternalAuthorization = "true" in complete list of access points, fail the test and skip other steps.
5. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
6. Verify the **GetEventPropertiesResponse** message from the DUT.
7. Check if there is an event with Topic tns1:AccessControl/Request/Timeout. If there is no event with such Topic skip other steps, fail the test and go to the next test.
8. Check that this event is not a Property event (MessageDescription.IsProperty = "false").
9. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".
10. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:AccessControl/Request/Credential Topic and tns1:AccessControl/Request/Timeout Topic as Filter and an InitialTerminationTime of timeout1.
11. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
12. Test Operator will invoke tns1:AccessControl/Request/Credential event for any Access Points with Capabilities.ExternalAuthorization = "true".
13. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.
14. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 13.
15. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).
16. Verify that TopicExpression is equal to tns1:AccessControl/Request/Credential for the Notification message.
17. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to one of existing Access Point Tokens Capabilities.ExternalAuthorization = "true" (e.g. complete list of access points contains Access Point with the same token and this Access Point has Capabilities.ExternalAuthorization = "true").

18. Verify that the notification contains Data.SimpleItem item with Name = "CredentialToken" and Value with type is equal to pt:ReferenceToken.

19. Verify that the notification which contains Data.SimpleItem item with Name = "CredentialHolderName" contains Value with type is equal to xs:string.

20. ONVIF Client waits for timeout to initiate tns1:AccessControl/Request/Timeout event.

21. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

22. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 21.

23. Verify that TopicExpression is equal to tns1:AccessControl/Request/Timeout for the Notification message.

24. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

25. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to Access Point Token from tns1:AccessControl/Request/Credential notification message.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/Request/Credential event at step 14.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/Request/Timeout event at step 22.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid AccessPointToken value, invalid CredentialToken, invalid CredentialHolderName).

- The DUT did not return at least one Access Point on step 3.
- The DUT did not return valid Topic tns1:AccessControl/Request/Credential in **GetEventPropertiesResponse**.
- The DUT did not return valid Topic tns1:AccessControl/Request/Timeout in **GetEventPropertiesResponse**.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** Test will be failed, if no Notification message is received for step 14 or 22 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

### 4.4.7 EXTERNAL AUTHORIZATION WITH INVALID TOKEN

**Test Case ID:** ACCESSCONTROL-11-1-7

**Specification Coverage:** ExternalAuthorization (ONVIF Access Control Service Specification)

**Feature Under Test:** ExternalAuthorization

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify External Authorization with invalid Token.

**Pre-Requisite:** Access Control Service address was received from the DUT. Access Point Entity is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client will invoke **ExternalAuthorization** request (invalid AccessPointToken).

4. The DUT will generate SOAP 1.2 fault message (**InvalidArgVal/NotFound**).

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send SOAP 1.2 fault message.

- The DUT sent incorrect SOAP 1.2 fault message (fault code, namespace, etc.).

**Note:** Other faults than specified in the test are acceptable, but the specified are preferable.

### 4.4.8 EXTERNAL AUTHORIZATION - COMMAND NOT SUPPORTED

**Test Case ID:** ACCESSCONTROL-11-1-8

**Specification Coverage:** ExternalAuthorization (ONVIF Access Control Service Specification)

**Feature Under Test:** ExternalAuthorization

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify External Authorization in case Access Point does not support it.

**Pre-Requisite:** Access Control Service address was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point is configured without External Authorization capability and added to the DUT, if possible.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.

2. Start the DUT.

3. Get complete list of access points from the DUT (see Annex A.1).

4. If Access Point with Token1 (Token1 is the first AccessPointInfo.token from the complete list of Access Points at step 3) has AccessPointInfo.Capabilities.ExternalAuthorization equal to true, then skip steps 5-6 and go to the step 7.

5. ONVIF Client will invoke **ExternalAuthorization** request (AccessPointToken = “Token1”, where Token1 is the first AccessPointInfo.token from the complete list of doors at step 3) to try perform external authorization.

6. The DUT will generate SOAP 1.2 fault message (**ActionNotSupported/NotSupported**).

7. Repeat steps 4-6 for all other tokens from the complete list of Access Point at step 3.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send SOAP 1.2 fault message.

- The DUT did not return at least one Access Point on step 3.

**Note:** If the DUT does not return any Access Point for step 3, skip steps 4-7, fail the test and go to the next test.

**Note:** Other faults than specified in the test are acceptable, but the specified are preferable.

### 4.4.9 ACCESS CONTROL - ACCESS GRANTED TO IDENTIFIER (EXTERNAL AUTHORIZATION)

**Test Case ID:** ACCESSCONTROL-11-1-9

**Specification Coverage:** tns1:AccessControl/AccessGranted/Anonymous (Access Control Service Specification), tns1:AccessControl/AccessGranted/Credential (Access Control Service Specification), tns1:AccessControl/Request/Identifier (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification), ExternalAuthorization (ONVIF Access Control Service Specification)

**Feature Under Test:** ExternalAuthorization, tns1:AccessControl/AccessGranted/Anonymous, tns1:AccessControl/AccessGranted/Credential, tns1:AccessControl/Request/Identifier.

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

**Test Purpose:** To verify Access Granted scenario in the case of External Authorization for Identifier.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with Identifier Access capability is configured and added to the DUT. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves complete list of access point info items by following the procedure mentioned in Annex A.1 with the following input and output parameters
   - out *accessPointInfoCompleteList* - complete access points info list
   - out *accessPointsNumber* - access points number
4. If there are no Access Points with Capabilities.IdentifierAccess = true in *accessPointInfoCompleteList*, FAIL the test and skip other steps.
5. ONVIF Client invokes **GetEventProperties**.
6. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list
7. If *topicSet* does not contain tns1:AccessControl/Request/Identifier topic, FAIL the test and skip other steps.
8. If *topicSet* does not contain naither tns1:AccessControl/AccessGranted/Anonymous topic nor tns1:AccessControl/AccessGranted/Credential topic, FAIL the test and skip other steps.
9. ONVIF Client verifies tns1:AccessControl/Request/Identifier topic (*requestIdentifierTopic*) from topicSet:

9.1. If *requestIdentifierTopic*.MessageDescription.IsProperty equals to true, FAIL the test, restore the DUT state, and skip other steps.

9.2. If *requestIdentifierTopic* does not contain MessageDescription.Source.SimpleItemDescription item with Name = "AccessPointToken", FAIL the test and skip other steps.

9.3. If *requestIdentifierTopic*.MessageDescription.Source.SimpleItemDescription with Name = "AccessPointToken" does not have Type = "pt:ReferenceToken", FAIL the test and skip other steps.

9.4. If *requestIdentifierTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "IdentifierType", FAIL the test and skip other steps.

9.5. If *requestIdentifierTopic*.MessageDescription.Data.SimpleItemDescription with Name = "IdentifierType" does not have Type = "xs:string", FAIL the test and skip other steps.

9.6. If *requestIdentifierTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "FormatType", FAIL the test and skip other steps.

9.7. If *requestIdentifierTopic*.MessageDescription.Data.SimpleItemDescription with Name = "FormatType" does not have Type = "xs:string", FAIL the test and skip other steps.

9.8. If *requestIdentifierTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "IdentifierValue", FAIL the test and skip other steps.

9.9. If *requestIdentifierTopic*.MessageDescription.Data.SimpleItemDescription with Name = "IdentifierValue" does not have Type = "xs:hexBinary", FAIL the test and skip other steps.

10. Set *topicFilter* := **"tns1:AccessControl/Request/Identifier|tns1:AccessControl/AccessGranted/Anonymous|tns1:AccessControl/AccessGranted/Credential"** ("|tns1:AccessControl/AccessGranted/Anonymous" part to be skipped if *topicSet* does not contain tns1:AccessControl/AccessGranted/Anonymous topic, "|tns1:AccessControl/AccessGranted/Credential" part to be skipped if *topicSet* does not contain tns1:AccessControl/AccessGranted/Credential topic).

11. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.3 with the following input and output parameters

- in *topicFilter* - Notification Topic

- out *s* - Subscription reference
- out *currentTime* - current time for the DUT
- out *terminationTime* - Subscription termination time

12. Test Operator will invoke **tns1:AccessControl/Request/Identifier** event for any Access Points with Capabilities.IdentifierAccess = true.

13. ONVIF Client retrieves and checks **tns1:AccessControl/Request/Identifier** event for the specified credential identifiers by following the procedure mentioned in Annex A.8 with the following input and output parameters

    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - out *accessPointToken* - token of access point for which **tns1:AccessControl/Request/Identifier** notification was recieved

14. If *accessPointInfoCompleteList* does not contain item with token = *accessPointToken*, FAIL the test and skip other steps.

15. If item with token = *accessPointToken* from *accessPointInfoCompleteList* does not have Capabilities.IdentifierAccess = true, FAIL the test and skip other steps.

16. ONVIF client invokes **ExternalAuthorization** request with parameters

    - AccessPointToken := *accessPointToken*
    - CredentialToken is skipped
    - Reason := "Test Reason"
    - Decision := Granted

17. The DUT responds with **ExternalAuthorizationResponse** message.

18. ONVIF Client retrieves and checks **tns1:AccessControl/AccessGranted/Anonymous** or **tns1:AccessControl/AccessGranted/Credential** event for the external authorzation by following the procedure mentioned in Annex A.10 with the following input and output parameters

    - in *s* - Subscription reference

- in *currentTime* - current time for the DUT
- in *terminationTime* - subscription termination time
- in *accessPointToken* - token of access point for which **tns1:AccessControl/Request/Identifier** notification was recieved

19. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.4 with the following input and output parameters
    - in *s* - Subscription reference

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **ExternalAuthorizationResponse** message.

### 4.4.10 ACCESS CONTROL - ACCESS DENIED TO IDENTIFIER (EXTERNAL AUTHORIZATION)

**Test Case ID:** ACCESSCONTROL-11-1-10

**Specification Coverage:** tns1:AccessControl/Denied/Anonymous (Access Control Service Specification), tns1:AccessControl/Denied/Credential (Access Control Service Specification), tns1:AccessControl/Request/Identifier (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification), ExternalAuthorization (ONVIF Access Control Service Specification)

**Feature Under Test:** ExternalAuthorization, tns1:AccessControl/Denied/Anonymous, tns1:AccessControl/Denied/Credential, tns1:AccessControl/Request/Identifier.

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

**Test Purpose:** To verify Access Denied scenario in the case of External Authorization for Identifier.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with Identifier

Access capability is configured and added to the DUT. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves complete list of access point info items by following the procedure mentioned in Annex A.1 with the following input and output parameters
   - out *accessPointInfoCompleteList* - complete access points info list
   - out *accessPointsNumber* - access points number
4. If there are no Access Points with Capabilities.IdentifierAccess = true in *accessPointInfoCompleteList*, FAIL the test and skip other steps.
5. ONVIF Client invokes **GetEventProperties**.
6. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list
7. If *topicSet* does not contain tns1:AccessControl/Request/Identifier topic, FAIL the test and skip other steps.
8. If *topicSet* does not contain naither tns1:AccessControl/Denied/Anonymous topic nor tns1:AccessControl/Denied/Credential topic, FAIL the test and skip other steps.
9. Set *topicFilter* := **"tns1:AccessControl/Request/Identifier|tns1:AccessControl/Denied/Anonymous|tns1:AccessControl/Denied/Credential"** ("|tns1:AccessControl/Denied/Anonymous" part to be skipped if *topicSet* does not contain tns1:AccessControl/Denied/Anonymous topic, "|tns1:AccessControl/Denied/Credential" part to be skipped if *topicSet* does not contain tns1:AccessControl/Denied/Credential topic).

10. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.3 with the following input and output parameters
    - in *topicFilter* - Notification Topic
    - out *s* - Subscription reference
    - out *currentTime* - current time for the DUT
    - out *terminationTime* - Subscription termination time
11. Test Operator will invoke **tns1:AccessControl/Request/Identifier** event for any Access Points with Capabilities.IdentifierAccess = true.
12. ONVIF Client retrieves and checks **tns1:AccessControl/Request/Identifier** event for the specified credential identifiers by following the procedure mentioned in Annex A.8 with the following input and output parameters
    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - out *accessPointToken* - token of access point for which **tns1:AccessControl/Request/Identifier** notification was recieved
13. If *accessPointInfoCompleteList* does not contain item with token = *accessPointToken*, FAIL the test and skip other steps.
14. If item with token = *accessPointToken* from *accessPointInfoCompleteList* does not have Capabilities.IdentifierAccess = true, FAIL the test and skip other steps.
15. ONVIF client invokes **ExternalAuthorization** request with parameters
    - AccessPointToken := *accessPointToken*
    - CredentialToken is skipped
    - Reason := "Test Reason"
    - Decision := Denied
16. The DUT responds with **ExternalAuthorizationResponse** message.
17. ONVIF Client retrieves and checks **tns1:AccessControl/Denied/Anonymous** or **tns1:AccessControl/Denied/Credential** event for the external authorzation by following the procedure mentioned in Annex A.11 with the following input and output parameters

- in *s* - Subscription reference
- in *currentTime* - current time for the DUT
- in *terminationTime* - subscription termination time
- in *accessPointToken* - token of access point for which **tns1:AccessControl/Request/Identifier** notification was recieved

18. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.4 with the following input and output parameters

   - in *s* - Subscription reference

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **ExternalAuthorizationResponse** message.

### 4.4.11 ACCESS CONTROL - ACCESS TIMEOUT TO IDENTIFIER (EXTERNAL AUTHORIZATION)

**Test Case ID:** ACCESSCONTROL-11-1-11

**Specification Coverage:** tns1:AccessControl/Request/Timeout (Access Control Service Specification), tns1:AccessControl/Request/Identifier (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification), ExternalAuthorization (ONVIF Access Control Service Specification)

**Feature Under Test:** ExternalAuthorization, tns1:AccessControl/Request/Timeout, tns1:AccessControl/Request/Identifier.

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

**Test Purpose:** To verify Access Timeout to Identifier scenario in the case of External Authorization.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with Identifier

Access capability is configured and added to the DUT. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves complete list of access point info items by following the procedure mentioned in Annex A.1 with the following input and output parameters
   - out *accessPointInfoCompleteList* - complete access points info list
   - out *accessPointsNumber* - access points number
4. If there are no Access Points with Capabilities.IdentifierAccess = true in *accessPointInfoCompleteList*, FAIL the test and skip other steps.
5. ONVIF Client invokes **GetEventProperties**.
6. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list
7. If *topicSet* does not contain tns1:AccessControl/Request/Identifier topic, FAIL the test and skip other steps.
8. If *topicSet* does not contain tns1:AccessControl/Request/Timeout topic, FAIL the test and skip other steps.
9. ONVIF Client verifies tns1:AccessControl/Request/Timeout topic (*requestTimeoutTopic*) from topicSet:

   9.1. If *requestTimeoutTopic*.MessageDescription.IsProperty equals to true, FAIL the test, restore the DUT state, and skip other steps.

9.2. If *requestTimeoutTopic* does not contain MessageDescription.Source.SimpleItemDescription item with Name = "AccessPointToken", FAIL the test and skip other steps.

9.3. If *requestTimeoutTopic*.MessageDescription.Source.SimpleItemDescription with Name = "AccessPointToken" does not have Type = "pt:ReferenceToken", FAIL the test and skip other steps.

10. Set *topicFilter* := **"tns1:AccessControl/Request/Identifier|tns1:AccessControl/Request/Timeout"**.

11. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.3 with the following input and output parameters

    - in *topicFilter* - Notification Topic
    - out *s* - Subscription reference
    - out *currentTime* - current time for the DUT
    - out *terminationTime* - Subscription termination time

12. Test Operator will invoke **tns1:AccessControl/Request/Identifier** event for any Access Points with Capabilities.IdentifierAccess = true.

13. ONVIF Client retrieves and checks **tns1:AccessControl/Request/Identifier** event for the specified credential identifiers by following the procedure mentioned in Annex A.8 with the following input and output parameters

    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *accessPointInfoCompleteList* - complete access points info list
    - out *accessPointToken* - token of access point for which **tns1:AccessControl/Request/Identifier** notification was recieved

14. ONVIF Client waits for *operationDelay* to initiate tns1:AccessControl/Request/Timeout event.

15. ONVIF Client retrieves and checks **tns1:AccessControl/Request/Timeout** event for the specified credential identifiers by following the procedure mentioned in Annex A.9 with the following input and output parameters

- in *s* - Subscription reference
- in *currentTime* - current time for the DUT
- in *terminationTime* - subscription termination time
- in *accessPointToken* - token of access point for which **tns1:AccessControl/Request/Timeout** notification was recieved

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**

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

## 4.5 Property Events

### 4.5.1 ACCESS CONTROL - ACCESS POINT ENABLED EVENT

**Test Case ID:** ACCESSCONTROL-5-1-1

**Specification Coverage:** tns1:AccessPoint/State/Enabled (Access Control Service Specification), GetAccessPointState (ONVIF Access Control Service Specification), Properties (ONVIF Core Specification)

**Feature Under Test:** GetAccessPointState

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

**Test Purpose:** To verify tns1:AccessPoint/State/Enabled event generation after subscription and to verify tns1:AccessPoint/State/Enabled event format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with Enable/Disable capability is configured and added to the DUT. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of access points from the DUT (see Annex A.1).
4. If there is no Access Points with DisableAccessPoint equal to true, fail the test and skip other steps.
5. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
6. Verify the **GetEventPropertiesResponse** message from the DUT.
7. Check if there is an event with Topic tns1:AccessPoint/State/Enabled. If there is no event with such Topic skip other steps, fail the test and go to the next test.
8. Check that this event is a Property event (MessageDescription.IsProperty = "true").
9. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".
10. Check that this event contains Data.SimpleItemDescription item with Name = "State" and Type = " xs:boolean".
11. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:AccessPoint/State/Enabled Topic as Filter and an InitialTerminationTime of timeout1.
12. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
13. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.
14. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 13 until Notification for all Access Points will be received.
15. Verify received Notification messages (correct value for UTC time, TopicExpression and wsnt:Message).
16. Verify that TopicExpression is equal to tns1:AccessPoint/State/Enabled for all received Notification messages.
17. Verify that each notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to one of the existing Access Point Tokens (e.g.

complete list of access points contains Access Point with the same token). Verify that there are Notification messages for each Access Point.

18. Verify that each notification contains Data.SimpleItem item with Name = "State" and Value with type is equal to xs:boolean.

19. Verify that Notify PropertyOperation = "Initialized".

20. ONVIF Client will invoke **GetAccessPointState** request for each Access Point with corresponding tokens.

21. Verify the **GetAccessPointStateResponse** messages from the DUT. Verify that Data.SimpleItem item with Name = "Enabled" from Notification message has the same value with Enabled elements from corresponding **GetAccessPointStateResponse** messages for each AccessPoint.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a Notification message that contains a property event tns1:AccessPoint/State/Enabled at least for one Access Point.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid AccessPointToken or State values, PropertyOperation is not equal to "Initialized").
- The DUT did not return at least one Access Point at step 3.
- The DUT did not return Topic tns1:AccessPoint/State/Enabled in **GetEventPropertiesResponse**.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** ONVIF Client at step 14 will wait for Notification messages until notification for all Access Points is received or Operation Delay has expired. Notification messages for all Access Points are assumed as received, if the number of Notification messages is equal to the number of Access Points.

**Note:** If the DUT does not return any access point for step 3, skip steps 4-21, fail the test and go to the next test.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

### 4.5.2 ACCESS CONTROL - ACCESS POINT ENABLED EVENT STATE CHANGE

**Test Case ID:** ACCESSCONTROL-5-1-2

**Specification Coverage:** tns1:AccessPoint/State/Enabled (Access Control Service Specification), EnableAccessPoint (ONVIF Access Control Service Specification), DisableAccessPoint (ONVIF Access Control Service Specification), GetAccessPointState (ONVIF Access Control Service Specification)

**Feature Under Test:** EnableAccessPoint, DisableAccessPoint, GetAccessPointState

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

**Test Purpose:** To verify tns1:AccessPoint/State/Enabled event generation after property was changed and to verify tns1:AccessPoint/State/Enabled event format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with Enable/ Disable capability is configured and added to the DUT. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get a complete list of access points from the DUT (see Annex A.1).
4. If there is no Access Points with DisableAccessPoint equal to true, fail the test and skip other steps.
5. If Access Point with Token1 (Token1 is the first AccessPointInfo.token from the complete list of access points at step 3) has AccessPointInfo.Capabilities.DisableAccessPoint equal to false, then skip steps 6-20 and go to the step 21.
6. ONVIF Client will invoke **GetAccessPointState** request (TokenList.Token = Token1, where Token1 is the first token from the complete list of access points at step 3) to retrieve Access Point state for specified token from the DUT.
7. Verify the **GetAccessPointStateResponse** message from the DUT.
8. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:AccessPoint/State/Enabled Topic as Filter and an InitialTerminationTime of timeout1.
9. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
10. If Access Point with Token1 (Token1 is the first AccessPointInfo.token from the complete list of access points at step 3) has AccessPointState.Enabled equal to true, then skip steps 11-12 and go to the step 13.
11. ONVIF Client will invoke **EnableAccessPoint** request (Token = “Token1”, where Token1 is the first AccessPointInfo.token from the complete list of access points at step 3) to try enabling access point.
12. Verify the **EnableAccessPointResponse** message from the DUT. Go to the step 15.
13. ONVIF Client will invoke **DisableAccessPoint** request (Token = “Token1”, where Token1 is the first AccessPointInfo.token from the complete list of access points at step 3) to try disabling access point.
14. Verify the **DisableAccessPointResponse** message from the DUT.
15. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

16. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 15 until Notification with PropertyOperation = "Changed" for Access Point with Token = “Token1” will be received.

17. Verify received Notification messages (correct value for UTC time, TopicExpression and wsnt:Message).

18. Verify that TopicExpression is equal to tns1:AccessPoint/State/Enabled for all received Notification messages.

19. Verify that notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value = “Token1” (e.g. a complete list of access points contains Access Point with the same token).

20. Verify that notification contains Data.SimpleItem item with Name = "State" and Value with type is equal to xs:boolean and with value equal to current state of Access Point.

21. Repeat steps 5-20 for all other tokens from complete list of access points at step 3.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a Notification message that contains a property event tns1:AccessPoint/State/Enabled with valid AccessPointToken and Enabled value.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid AccessPointToken or State values).
- The DUT did not return at least one Access Point on step 3.
- The DUT did not send valid **EnableAccessPointResponse** message.
- The DUT did not send valid **DisableAccessPointResponse** message.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** All Notification messages except messages with PropertyOperation = "Changed" will be ignored.

**Note:** Test will be failed, if no required Notification messages are received for step 15 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

**Note:** If the DUT does not return any access point for step 3, skip steps 4-21, fail the test and go to the next test.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during a test, if InitialTerminationTime expires.

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

## 4.6 Access granted events

### 4.6.1 ACCESS CONTROL - ACCESS GRANTED TO ANONYMOUS EVENT

**Test Case ID:** ACCESSCONTROL-6-1-1

**Specification Coverage:** tns1:AccessControl/AccessGranted/Anonymous (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification)

**Feature Under Test:** tns1:AccessControl/AccessGranted/Anonymous

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

**Test Purpose:** To verify tns1:AccessControl/AccessGranted/Anonymous event generation and to verify tns1:AccessControl/AccessGranted/Anonymous event format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with

Anonymous Access capability is configured and added to the DUT. Device supports Pull-Point Notification feature. Device supports tns1:AccessControl/AccessGranted/Anonymous.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of access points from the DUT (see Annex A.1).
4. If there are no Access Point with Capabilities.AnonymousAccess = “true” in complete list of access points, fail the test and skip other steps.
5. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
6. Verify the **GetEventPropertiesResponse** message from the DUT.
7. Check if there is an event with Topic tns1:AccessControl/AccessGranted/Anonymous. If there is no event with such Topic skip other steps, fail the test and go to the next test.
8. Check that this event is not a Property event (MessageDescription.IsProperty = "false").
9. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".
10. Check that if this event contains Data.SimpleItemDescription item with Name = "External", than it has Type = "xs:boolean".
11. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:AccessControl/AccessGranted/Anonymous Topic as Filter and an InitialTerminationTime of timeout1.
12. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
13. Test Operator will invoke tns1:AccessControl/AccessGranted/Anonymous event for any Access Points with Capabilities.AnonymousAccess = “true”.
14. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.
15. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 14.
16. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

17. Verify that TopicExpression is equal to tns1:AccessControl/AccessGranted/Anonymous for the Notification message.

18. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to one of existing Access Point Tokens with Capabilities.AnonymousAccess = “true” (e.g. complete list of access points contains Access Point with the same token and this Access Point has Capabilities.AnonymousAccess = “true”).

19. Verify that the notification doesn’t contain Data.SimpleItem item with Name = "External", if it was not included in Topic in **GetEventPropertiesResponse** message. Verify that the notification doesn’t contain Data.SimpleItem item with Name = "External" or contains Data.SimpleItem item with Name = "External" and its Value is equal to “false”, if it was included in Topic in **GetEventPropertiesResponse** message.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/AccessGranted/Anonymous event.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid AccessPointToken value, invalid External).
- The DUT did not return at least one Access Point on step 3.
- The DUT did not return valid Topic tns1:AccessControl/AccessGranted/Anonymous in **GetEventPropertiesResponse**.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** Test will be failed, if no Notification message is received for step 15 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

### 4.6.2 ACCESS CONTROL - ACCESS GRANTED WITH CREDENTIAL EVENT

**Test Case ID:** ACCESSCONTROL-6-1-2

**Specification Coverage:** tns1:AccessControl/AccessGranted/Credential (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification)

**Feature Under Test:** tns1:AccessControl/AccessGranted/Credential

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

**Test Purpose:** To verify tns1:AccessControl/AccessGranted/Credential event generation and to verify tns1:AccessControl/AccessGranted/Credential event format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point is configured and added to the DUT. Device supports Pull-Point Notification feature. Device supports tns1:AccessControl/AccessGranted/Credential.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.

2. Start the DUT.

3. Get complete list of access points from the DUT (see Annex A.1).

4. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.

5. Verify the **GetEventPropertiesResponse** message from the DUT.

6. Check if there is an event with Topic tns1:AccessControl/AccessGranted/Credential. If there is no event with such Topic skip other steps, fail the test and go to the next test.

7. Check that this event is not a Property event (MessageDescription.IsProperty = "false").

8. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".

9. Check that this event contains Data.SimpleItemDescription item with Name = "CredentialToken" and Type = "pt:ReferenceToken".

10. Check that if this event contains Data.SimpleItemDescription item with Name = "CredentialHolderName", than it has Type = " xs:string".

11. Check that if this event contains Data.SimpleItemDescription item with Name = "External", than it has Type = "xs:boolean".

12. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:AccessControl/AccessGranted/Credential Topic as Filter and an InitialTerminationTime of timeout1.

13. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.

14. Test Operator will invoke tns1:AccessControl/AccessGranted/Credential event for any Access Points.

15. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

16. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 15.

17. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

18. Verify that TopicExpression is equal to tns1:AccessControl/AccessGranted/Credential for the Notification message.

19. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to one of existing Access Point Tokens (e.g. complete list of access points contains Access Point with the same token).

20. Verify that the notification contains Data.SimpleItem item with Name = "CredentialToken" and Value with type is equal to pt:ReferenceToken.

21. Verify that the notification which contains Data.SimpleItem item with Name = "CredentialHolderName" contains Value with type is equal to xs:string.

22. Verify that the notification doesn't contain Data.SimpleItem item with Name = "External", if it was not included in Topic in **GetEventPropertiesResponse** message. Verify that the notification doesn't contain Data.SimpleItem item with Name = "External" or contains Data.SimpleItem item with Name = "External" and its Value is equal to "false", if it was included in Topic in **GetEventPropertiesResponse** message.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/AccessGranted/Credential event.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid AccessPointToken value, invalid CredentialToken, invalid CredentialHolderName, invalid External).
- The DUT did not return at least one Access Point on step 3.
- The DUT did not return valid Topic tns1:AccessControl/AccessGranted/Credential in **GetEventPropertiesResponse**.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** Test will be failed, if no Notification message is received for step 16 during certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

### 4.6.3 ACCESS CONTROL - ACCESS GRANTED TO IDENTIFIER EVENT

**Test Case ID:** ACCESSCONTROL-6-1-3

**Specification Coverage:** Access granted/Identifier (ONVIF Access Control Service Specification)

**Feature Under Test:** tns1:AccessControl/AccessGranted/Identifier

**WSDL Reference:** event.wsdl

**Test Purpose:** To verify tns1:AccessControl/AccessGranted/Identifier event generation and to verify tns1:AccessControl/AccessGranted/Identifier event format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. tns1:AccessControl/AccessGranted/Identifier event topic is supported by the DUT. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client 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:AccessControl/AccessGranted/Identifier topic, FAIL the test, restore the DUT state, and skip other steps.

6. ONVIF Client verifies tns1:AccessControl/AccessGranted/Identifier topic (*accessGrantedIdentifierTopic*) from topicSet:

   6.1. If *accessGrantedIdentifierTopic*.MessageDescription.IsProperty equals to true, FAIL the test, restore the DUT state, and skip other steps.

   6.2. If *accessGrantedIdentifierTopic* does not contain MessageDescription.Source.SimpleItemDescription item with Name = "AccessPointToken", FAIL the test and skip other steps.

   6.3. If *accessGrantedIdentifierTopic*.MessageDescription.Source.SimpleItemDescription with Name = "AccessPointToken" does not have Type = "pt:ReferenceToken", FAIL the test and skip other steps.

   6.4. If *accessGrantedIdentifierTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "IdentifierType", FAIL the test and skip other steps.

   6.5. If *accessGrantedIdentifierTopic*.MessageDescription.Data.SimpleItemDescription with Name = "IdentifierType" does not have Type = "xs:string", FAIL the test and skip other steps.

   6.6. If *accessGrantedIdentifierTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "FormatType", FAIL the test and skip other steps.

   6.7. If *accessGrantedIdentifierTopic*.MessageDescription.Data.SimpleItemDescription with Name = "FormatType" does not have Type = "xs:string", FAIL the test and skip other steps.

   6.8. If *accessGrantedIdentifierTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "IdentifierValue", FAIL the test and skip other steps.

   6.9. If *accessGrantedIdentifierTopic*.MessageDescription.Data.SimpleItemDescription with Name = "IdentifierValue" does not have Type = "xs:hexBinary", FAIL the test and skip other steps.

7. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.3 with the following input and output parameters

   • in **"tns1:AccessControl/AccessGranted/Identifier"** - Notification Topic

- out *s* - Subscription reference
- out *currentTime* - current time for the DUT
- out *terminationTime* - Subscription termination time

8. Test Operator will invoke **tns1:AccessControl/AccessGranted/Identifier** event for any Access Point.

9. ONVIF Client retrieves and checks **tns1:AccessControl/AccessGranted/Identifier** event for the specified credential identifiers by following the procedure mentioned in Annex A.6 with the following input and output parameters

    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time

10. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.4 with the following input and output parameters

    - in *s* - Subscription reference

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**

## 4.7 Access taken events

### 4.7.1 ACCESS CONTROL - ACCESS TAKEN BY ANONYMOUS EVENT

**Test Case ID:** ACCESSCONTROL-7-1-1

**Specification Coverage:** tns:AccessControl/AccessGranted/Anonymous (Access Control Service Specification), AccessControl/AccessTaken/Anonymous (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification)

**Feature Under Test:** tns:AccessControl/AccessGranted/Anonymous

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

**Test Purpose:** To verify tns1:AccessControl/AccessTaken/Anonymous event generation and to verify tns1:AccessControl/AccessTaken/Anonymous event format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with Anonymous Access capability and Access Taken capability is configured and added to the DUT. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of access points from the DUT (see Annex A.1).
4. If there are no Access Point with Capabilities.AnonymousAccess = "true" and Capabilities.AccessTaken = "true" in complete list of access points, fail the test and skip other steps.
5. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
6. Verify the **GetEventPropertiesResponse** message from the DUT.
7. Check if there is an event with Topic tns1:AccessControl/AccessTaken/Anonymous. If there is no event with such Topic skip other steps, fail the test and go to the next test.
8. Check that this event is not a Property event (MessageDescription.IsProperty = "false").
9. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".
10. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:AccessControl/AccessGranted/Anonymous Topic and tns1:AccessControl/AccessTaken/Anonymous Topic as Filter and an InitialTerminationTime of timeout1.
11. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
12. Test Operator will invoke tns1:AccessControl/AccessGranted/Anonymous event for any Access Points with Capabilities.AnonymousAccess = “true” and Capabilities.AccessTaken = "true".

13. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

14. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 13.

15. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

16. Verify that TopicExpression is equal to tns1:AccessControl/AccessGranted/Anonymous for the Notification message.

17. Test Operator will invoke tns1:AccessControl/AccessTaken/Anonymous event for the same Access Point.

18. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

19. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 18.

20. Verify that TopicExpression is equal to tns1:AccessControl/AccessTaken/Anonymous for the Notification message.

21. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to AccessPointToken from tns1:AccessControl/AccessGranted/Anonymous notification and there is Access Point Tokens with Capabilities.AnonymousAccess = "true" and Capabilities.AccessTaken = "true" in the complete list of access points (e.g. complete list of access points contains Access Point with the same token and this Access Point has Capabilities.AnonymousAccess = "true" and Capabilities.AccessTaken = "true").

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/AccessGranted/Anonymous event on step 14.

- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/AccessTaken/Anonymous event on step 19.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid AccessPointToken value) on step 19.
- The DUT did not return at least one Access Point on step 3.
- The DUT did not return valid Topic tns1:AccessControl/AccessTaken/Anonymous in **GetEventPropertiesResponse**.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** Test will be failed, if no Notification message is received for steps 14 and 19 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

### 4.7.2 ACCESS CONTROL - ACCESS TAKEN WITH CREDENTIAL EVENT

**Test Case ID:** ACCESSCONTROL-7-1-2

**Specification Coverage:** tns:AccessControl/AccessGranted/Credential (Access Control Service Specification), tns1:AccessControl/AccessTaken/Credential (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification)

**Feature Under Test:** tns:AccessControl/AccessGranted/Credential

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

**Test Purpose:** To verify tns1:AccessControl/AccessTaken/Credential event generation and to verify tns1:AccessControl/AccessTaken/Credential event format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with Access Taken capability is configured and added to the DUT. Device supports Pull-Point Notification feature. Device supports tns1:AccessControl/AccessTaken/Credential

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of access points from the DUT (see Annex A.1).
4. If there are no Access Points with Capabilities.AccessTaken = "true" in complete list of access points, fail the test and skip other steps.
5. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
6. Verify the **GetEventPropertiesResponse** message from the DUT.
7. Check if there is an event with Topic tns1:AccessControl/AccessTaken/Credential. If there is no event with such Topic skip other steps, fail the test and go to the next test.
8. Check that this event is not a Property event (MessageDescription.IsProperty = "false").
9. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".
10. Check that this event contains Data.SimpleItemDescription item with Name = "CredentialToken" and Type = "pt:ReferenceToken".
11. Check that if this event contains Data.SimpleItemDescription item with Name = "CredentialHolderName", than it has Type = " xs:string".
12. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:AccessControl/AccessGranted/Credential Topic and tns1:AccessControl/AccessTaken/Credential Topic as Filter and an InitialTerminationTime of timeout1.
13. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
14. Test Operator will invoke tns1:AccessControl/AccessGranted/Credential event for any Access Points with Capabilities.AccessTaken = "true".
15. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

16. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 15.

17. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

18. Verify that TopicExpression is equal to tns1:AccessControl/AccessGranted/Credential for the Notification message.

19. Test Operator will invoke tns1:AccessControl/AccessTaken/Credential event for the same Access Point.

20. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

21. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 20.

22. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

23. Verify that TopicExpression is equal to tns1:AccessControl/AccessTaken/Credential for the Notification message.

24. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to AccessPointToken from tns1:AccessControl/AccessGranted/Credential notification and there is Access Point Tokens with Capabilities.AccessTaken = "true" in the complete list of access points (e.g. complete list of access points contains Access Point with the same token and this Access Point has Capabilities.AccessTaken = "true").

25. Verify that the notification contains Data.SimpleItem item with Name = "CredentialToken" and Value with type is equal to pt:ReferenceToken. Verify that this value is equal to the same Data.SimpleItem from tns1:AccessControl/AccessGranted/Credential notification.

26. Verify that the notification which contains Data.SimpleItem item with Name = "CredentialHolderName" contains Value with type is equal to xs:string. Verify that this value is equal to the same Data.SimpleItem from tns1:AccessControl/AccessGranted/Credential notification.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/AccessGranted/Credential event on step 16.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/AccessTaken/Credential event on step 21.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid AccessPointToken value, invalid CredentialToken value, and invalid CredentialHolderName value) on step 21.
- The DUT did not return at least one Access Point on step 3.
- The DUT did not return valid Topic tns1:AccessControl/AccessTaken/Credential in **GetEventPropertiesResponse**.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** Test will be failed, if no Notification message is received for step 16 and 19 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

## 4.8 Access not taken events

### 4.8.1 ACCESS CONTROL - ACCESS NOT TAKEN BY ANONYMOUS EVENT

**Test Case ID:** ACCESSCONTROL-8-1-1

**Specification Coverage:** tns:AccessControl/AccessGranted/Anonymous (Access Control Service Specification), AccessControl/AccessNotTaken/Anonymous (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification)

**Feature Under Test:** AccessControl/AccessNotTaken/Anonymous

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

**Test Purpose:** To verify tns1:AccessControl/AccessNotTaken/Anonymous event generation and to verify tns1:AccessControl/AccessNotTaken/Anonymous event format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with Anonymous Access capability and Access Taken capability is configured and added to the DUT. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of access points from the DUT (see Annex A.1).
4. If there are no Access Points with Capabilities.AnonymousAccess = "true" and Capabilities.AccessTaken = "true" in complete list of access points, fail the test and skip other steps.
5. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
6. Verify the **GetEventPropertiesResponse** message from the DUT.
7. Check if there is an event with Topic tns1:AccessControl/AccessNotTaken/Anonymous. If there is no event with such Topic skip other steps, fail the test and go to the next test.
8. Check that this event is not a Property event (MessageDescription.IsProperty = "false").
9. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".
10. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:AccessControl/AccessGranted/Anonymous Topic and tns1:AccessControl/AccessNotTaken/Anonymous Topic as Filter and an InitialTerminationTime of timeout1.

11. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.

12. Test Operator will invoke tns1:AccessControl/AccessGranted/Anonymous event for any Access Points with Capabilities.AnonymousAccess = “true” and Capabilities.AccessTaken = "true".

13. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

14. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 13.

15. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

16. Verify that TopicExpression is equal to tns1:AccessControl/AccessGranted/Anonymous for the Notification message.

17. Test Operator will invoke tns1:AccessControl/AccessNotTaken/Anonymous event for the same Access Point.

18. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

19. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 18.

20. Verify that TopicExpression is equal to tns1:AccessControl/AccessNotTaken/Anonymous for the Notification message.

21. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to AccessPointToken from tns1:AccessControl/AccessGranted/Anonymous notification and there is Access Point Tokens with Capabilities.AnonymousAccess = "true" and Capabilities.AccessTaken = "true" in the complete list of access points (e.g. complete list of access points contains Access Point with the same token and this Access Point has Capabilities.AnonymousAccess = "true" and Capabilities.AccessTaken = "true").

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/AccessGranted/Anonymous event on step 14.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/AccessNotTaken/Anonymous event on step 19.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid AccessPointToken value) on step 19.
- The DUT did not return at least one Access Point on step 3.
- The DUT did not return valid Topic tns1:AccessControl/AccessNotTaken/Anonymous in **GetEventPropertiesResponse**.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** Test will be failed, if no Notification message is received for steps 14 and 19 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

### 4.8.2 ACCESS CONTROL - ACCESS NOT TAKEN WITH CREDENTIAL EVENT

**Test Case ID:** ACCESSCONTROL-8-1-2

**Specification Coverage:** tns:AccessControl/AccessGranted/Credential (Access Control Service Specification), tns1:AccessControl/AccessNotTaken/Credential (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification)

**Feature Under Test:** tns1:AccessControl/AccessNotTaken/Credential

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

**Test Purpose:** To verify tns1:AccessControl/AccessNotTaken/Credential event generation and to verify tns1:AccessControl/AccessNotTaken/Credential event format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with Access Taken capability is configured and added to the DUT. Device supports Pull-Point Notification feature. Device supports tns1:AccessControl/AccessNotTaken/Credential.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of access points from the DUT (see Annex A.1).
4. If there are no Access Point with Capabilities.AccessTaken = "true" in complete list of access points, fail the test and skip other steps.
5. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
6. Verify the **GetEventPropertiesResponse** message from the DUT.
7. Check if there is an event with Topic tns1:AccessControl/AccessNotTaken/Credential. If there is no event with such Topic skip other steps, fail the test and go to the next test.
8. Check that this event is not a Property event (MessageDescription.IsProperty = "false").
9. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".
10. Check that this event contains Data.SimpleItemDescription item with Name = "CredentialToken" and Type = "pt:ReferenceToken".
11. Check that if this event contains Data.SimpleItemDescription item with Name = "CredentialHolderName", than it has Type = " xs:string".
12. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:AccessControl/AccessGranted/Credential Topic and tns1:AccessControl/AccessNotTaken/Credential Topic as Filter and an InitialTerminationTime of timeout1.

13. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
14. Test Operator will invoke tns1:AccessControl/AccessGranted/Credential event for any Access Points with Capabilities.AccessTaken = "true".
15. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.
16. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 15.
17. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).
18. Verify that TopicExpression is equal to tns1:AccessControl/AccessGranted/Credential for the Notification message.
19. Test Operator will invoke tns1:AccessControl/AccessNotTaken/Credential event for the same Access Point.
20. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.
21. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 20.
22. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).
23. Verify that TopicExpression is equal to tns1:AccessControl/AccessNotTaken/Credential for the Notification message.
24. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to AccessPointToken from tns1:AccessControl/AccessGranted/Credential notification and there is Access Point Tokens with Capabilities.AccessTaken = "true" in the complete list of access points (e.g. complete list of access points contains Access Point with the same token and this Access Point has Capabilities.AccessTaken = "true").
25. Verify that the notification contains Data.SimpleItem item with Name = "CredentialToken" and Value with type is equal to pt:ReferenceToken. Verify that this value is equal to the same Data.SimpleItem from tns1:AccessControl/AccessGranted/Credential notification.
26. Verify that the notification which contains Data.SimpleItem item with Name = "CredentialHolderName" contains Value with type is equal to xs:string. Verify that this value

is equal to the same Data.SimpleItem from tns1:AccessControl/AccessGranted/Credential notification.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/AccessGranted/Credential event on step 16.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/AccessNotTaken/Credential event on step 21.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid AccessPointToken value, invalid CredentialToken value, and invalid CredentialHolderName value) on step 21.
- The DUT did not return at least one Access Point on step 3.
- The DUT did not return valid Topic tns1:AccessControl/AccessNotTaken/Credential in **GetEventPropertiesResponse**.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime will expire.

**Note:** Test will be failed, if no Notification message is received for step 16 and 19 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

## 4.9 Access Denied Events

### 4.9.1 ACCESS CONTROL - ACCESS DENIED TO ANONYMOUS EVENT

**Test Case ID:** ACCESSCONTROL-9-1-1

**Specification Coverage:** tns1:AccessControl/Denied/Anonymous (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification)

**Feature Under Test:** tns1:AccessControl/Denied/Anonymous

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

**Test Purpose:** To verify tns1:AccessControl/Denied/Anonymous event generation and to verify tns1:AccessControl/Denied/Anonymous event format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with Anonymous Access capability is configured and added to the DUT. Device supports Pull-Point Notification feature. Device supports tns1:AccessControl/Denied/Anonymous.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of access points from the DUT (see Annex A.1).
4. If there are no Access Points with Capabilities.AnonymousAccess = “true” in complete list of access points, fail the test and skip other steps.
5. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
6. Verify the **GetEventPropertiesResponse** message from the DUT.
7. Check if there is an event with Topic tns1:AccessControl/Denied/Anonymous. If there is no event with such Topic skip other steps, fail the test and go to the next test.

8. Check that this event is not a Property event (MessageDescription.IsProperty = "false").

9. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".

10. Check that if this event contains Data.SimpleItemDescription item with Name = "External", than it has Type = "xs:boolean".

11. Check that this event contains Data.SimpleItemDescription item with Name = "Reason" and Type = " xs:string".

12. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:AccessControl/Denied/Anonymous Topic as Filter and an InitialTerminationTime of timeout1.

13. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.

14. Test Operator will invoke tns1:AccessControl/Denied/Anonymous event for any Access Points with Capabilities.AnonymousAccess = “true”.

15. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

16. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 15.

17. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

18. Verify that TopicExpression is equal to tns1:AccessControl/Denied/Anonymous for the Notification message.

19. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to one of existing Access Point Tokens with Capabilities.AnonymousAccess = “true” (e.g. complete list of access points contains Access Point with the same token and this Access Point has Capabilities.AnonymousAccess = “true”).

20. Verify that the notification doesn’t contain Data.SimpleItem item with Name = "External", if it was not included in Topic in **GetEventPropertiesResponse** message. Verify that the notification doesn’t contain Data.SimpleItem item with Name = "External" or contains Data.SimpleItem item with Name = "External" and its Value is equal to “false”, if it was included in Topic in **GetEventPropertiesResponse** message.

21. Verify that the notification contains Data.SimpleItem item with Name = "Reason" contains Value with type is equal to xs:string. Check that Reason value is one of the following

strings: “CredentialNotEnabled”, “CredentialNotActive”, “CredentialExpired”, “InvalidPIN”, “NotPermittedAtThisTime”, “Unauthorized”, “Other”.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/Denied/Anonymous event.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid AccessPointToken value, invalid External, invalid Reason).
- The DUT did not return at least one Access Point on step 3.
- The DUT did not return valid Topic tns1:AccessControl/Denied/Anonymous in **GetEventPropertiesResponse**.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** Test will be failed, if no Notification message will be received for step 16 during certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

## 4.9.2 ACCESS CONTROL - ACCESS DENIED WITH CREDENTIAL EVENT

**Test Case ID:** ACCESSCONTROL-9-1-2

**Specification Coverage:** tns1:AccessControl/Denied/Credential (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification)

**Feature Under Test:** tns1:AccessControl/Denied/Credential

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

**Test Purpose:** To verify tns1:AccessControl/Denied/Credential generation and to verify tns1:AccessControl/Denied/Credential event format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point is configured and added to the DUT. Device supports Pull-Point Notification feature. Device supports tns1:AccessControl/Denied/Credential.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of access points from the DUT (see Annex A.1).
4. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
5. Verify the **GetEventPropertiesResponse** message from the DUT.
6. Check if there is an event with Topic tns1:AccessControl/Denied/Credential. If there is no event with such Topic skip other steps, fail the test and go to the next test.
7. Check that this event is not a Property event (MessageDescription.IsProperty = "false").
8. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".
9. Check that this event contains Data.SimpleItemDescription item with Name = "CredentialToken" and Type = "pt:ReferenceToken".

10. Check that if this event contains Data.SimpleItemDescription item with Name = "CredentialHolderName", than it has Type = " xs:string".

11. Check that if this event contains Data.SimpleItemDescription item with Name = "External", than it has Type = "xs:boolean".

12. Check that this event contains Data.SimpleItemDescription item with Name = "Reason" and Type = " xs:string".

13. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:AccessControl/Denied/Credential Topic as Filter and an InitialTerminationTime of timeout1.

14. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.

15. Test Operator will invoke tns1:AccessControl/Denied/Credential event for any Access Points.

16. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

17. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 16.

18. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

19. Verify that TopicExpression is equal to tns1:AccessControl/Denied/Credential for the Notification message.

20. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to one of existing Access Point Tokens (e.g. complete list of access points contains Access Point with the same token).

21. Verify that the notification contains Data.SimpleItem item with Name = "CredentialToken" and Value with type is equal to pt:ReferenceToken.

22. Verify that the notification which contains Data.SimpleItem item with Name = "CredentialHolderName" contains Value with type is equal to xs:string.

23. Verify that the notification doesn't contain Data.SimpleItem item with Name = "External", if it was not included in Topic in **GetEventPropertiesResponse** message. Verify that the notification doesn't contain Data.SimpleItem item with Name = "External" or contains Data.SimpleItem item with Name = "External" and its Value is equal to “false”, if it was included in Topic in **GetEventPropertiesResponse** message.

24. Verify that the notification contains Data.SimpleItem item with Name = "Reason" contains Value with type is equal to xs:string. Check that Reason value is one of the following

strings: “CredentialNotEnabled”, “CredentialNotActive”, “CredentialExpired”, “InvalidPIN”, “NotPermittedAtThisTime”, “Unauthorized”, “Other”.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/Denied/Credential event.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid AccessPointToken value, invalid CredentialToken, invalid CredentialHolderName, invalid External, invalid Reason).
- The DUT did not return at least one Access Point on step 3.
- The DUT did not return valid Topic tns1:AccessControl/Denied/Credential in **GetEventPropertiesResponse**.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** Test will be failed, if no Notification message is received for step 17 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

### 4.9.3 ACCESS CONTROL - ACCESS DENIED WITH CREDENTIAL EVENT (CREDENTIAL NOT FOUND - CARD)

**Test Case ID:** ACCESSCONTROL-9-1-3

**Specification Coverage:** tns1:AccessControl/Denied/CredentialNotFound/Card (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification)

**Feature Under Test:** tns1:AccessControl/Denied/CredentialNotFound/Card

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

**Test Purpose:** To verify tns1:AccessControl/Denied/CredentialNotFound/Card generation and to verify tns1:AccessControl/Denied/CredentialNotFound/Card format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with tns1:AccessControl/Denied/CredentialNotFound/Card event capability is configured and added to the DUT. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of access points from the DUT (see Annex A.1).
4. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
5. Verify the **GetEventPropertiesResponse** message from the DUT.
6. Check if there is an event with Topic tns1:AccessControl/Denied/CredentialNotFound/Card. If there is no event with such Topic fail the test and skip other steps.
7. Check that this event is not a Property event (MessageDescription.IsProperty = "false").
8. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".
9. Check that this event contains Data.SimpleItemDescription item with Name = "Card" and Type = " xs:string".

10. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:AccessControl/Denied/CredentialNotFound/Card Topic as Filter and an InitialTerminationTime of timeout1.

11. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.

12. Test Operator will invoke tns1:AccessControl/Denied/CredentialNotFound/Card event for any Access Points.

13. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

14. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 13.

15. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

16. Verify that TopicExpression is equal to tns1:AccessControl/Denied/CredentialNotFound/Card for the Notification message.

17. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to one of existing Access Point Tokens (e.g. complete list of access points contains Access Point with the same token).

18. Verify that the notification contains Data.SimpleItem item with Name = "Card" and Value with type is equal to xs:string.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**

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

- The DUT did not send valid SubscriptionReference.

- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/Denied/CredentialNotFound/Card event.

- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid AccessPointToken value, invalid Card).

- The DUT did not return at least one Access Point on step 3.
- The DUT did not return valid Topic tns1:AccessControl/Denied/CredentialNotFound/Card in **GetEventPropertiesResponse**.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** Test will be failed, if no Notification message is received for step 14 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

### 4.9.4 ACCESS CONTROL - ACCESS DENIED WITH CREDENTIAL EVENT (CREDENTIAL NOT FOUND)

**Test Case ID:** ACCESSCONTROL-9-1-4

**Specification Coverage:** CredentialNotFound (Access Control Service Specification), GetAccessPointInfoList command (ONVIF Access Control Service Specification)

**Feature Under Test:** tns1:AccessControl/Denied/CredentialNotFound

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

**Test Purpose:** To verify tns1:AccessControl/Denied/CredentialNotFound generation and to verify tns1:AccessControl/Denied/CredentialNotFound format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with tns1:AccessControl/Denied/CredentialNotFound event capability is configured and added to the DUT. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves complete list of access point info items by following the procedure mentioned in Annex A.1 with the following input and output parameters
   - out *accessPointInfoCompleteList* - complete access points info list
   - out *accessPointsNumber* - access points number
4. If *accessPointsNumber* < 1, FAIL the test and skip other steps.
5. ONVIF Client invokes **GetEventProperties** request.
6. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list
7. If *topicSet* does not contain tns1:AccessControl/Denied/CredentialNotFound topic, FAIL the test and skip other steps.
8. Set *topic* := tns1:AccessControl/Denied/CredentialNotFound topic from *topicSet*.
9. If *topic*.MessageDescription.IsProperty is present and equal to true, FAIL the test and skip other steps.
10. If *topic* does not contain MessageDescription.Source.SimpleItemDescription item with Name = "AccessPointToken", FAIL the test and skip other steps.
11. If *topic*.MessageDescription.Source.SimpleItemDescription with Name = "AccessPointToken" does not have Type = "pt:ReferenceToken", FAIL the test and skip other steps.
12. If *topic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "IdentifierType", FAIL the test and skip other steps.

13. If *topic*.MessageDescription.Data.SimpleItemDescription with Name = "IdentifierType" does not have Type = "xs:string", FAIL the test and skip other steps.

14. If *topic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "IdentifierValue", FAIL the test and skip other steps.

15. If *topic*.MessageDescription.Data.SimpleItemDescription with Name = "IdentifierValue" does not have Type = "xs:hexBinary", FAIL the test and skip other steps.

16. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.3 with the following input and output parameters

    - in **tns1:AccessControl/Denied/CredentialNotFound** - Notification Topic
    - out *s* - Subscription Reference
    - out *currentTime* - current time for the DUT
    - out *terminationTime* - Subscription Termination time

17. Test Operator will invoke **tns1:AccessControl/Denied/CredentialNotFound** event for any Access Points.

18. ONVIF Client retrieves and checks **tns1:AccessControl/Denied/CredentialNotFound** event by following the procedure mentioned in Annex A.5 with the following input and output parameters

    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *accessPointInfoCompleteList* - access points list

19. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.4 with the following input and output parameters

    - in *s* - Subscription reference

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**

### 4.9.5 ACCESS CONTROL - ACCESS DENIED TO IDENTIFIER EVENT

**Test Case ID:** ACCESSCONTROL-9-1-5

**Specification Coverage:** Access denied/Identifier (ONVIF Access Control Service Specification)

**Feature Under Test:** tns1:AccessControl/Denied/Identifier

**WSDL Reference:** event.wsdl

**Test Purpose:** To verify tns1:AccessControl/Denied/Identifier event format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. tns1:AccessControl/Denied/Identifier event topic is supported by the DUT. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client 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:AccessControl/Denied/Identifier topic, FAIL the test, restore the DUT state, and skip other steps.
6. ONVIF Client verifies tns1:AccessControl/Denied/Identifier topic (*accessDeniedIdentifierTopic*) from topicSet:

6.1. If *accessDeniedIdentifierTopic*.MessageDescription.IsProperty equals to true, FAIL the test, restore the DUT state, and skip other steps.

6.2. If *accessDeniedIdentifierTopic* does not contain MessageDescription.Source.SimpleItemDescription item with Name = "AccessPointToken", FAIL the test and skip other steps.

6.3. If *accessDeniedIdentifierTopic*.MessageDescription.Source.SimpleItemDescription with Name = "AccessPointToken" does not have Type = "pt:ReferenceToken", FAIL the test and skip other steps.

6.4. If *accessDeniedIdentifierTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "IdentifierType", FAIL the test and skip other steps.

6.5. If *accessDeniedIdentifierTopic*.MessageDescription.Data.SimpleItemDescription with Name = "IdentifierType" does not have Type = "xs:string", FAIL the test and skip other steps.

6.6. If *accessDeniedIdentifierTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "FormatType", FAIL the test and skip other steps.

6.7. If *accessDeniedIdentifierTopic*.MessageDescription.Data.SimpleItemDescription with Name = "FormatType" does not have Type = "xs:string", FAIL the test and skip other steps.

6.8. If *accessDeniedIdentifierTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "IdentifierValue", FAIL the test and skip other steps.

6.9. If *accessDeniedIdentifierTopic*.MessageDescription.Data.SimpleItemDescription with Name = "IdentifierValue" does not have Type = "xs:hexBinary", FAIL the test and skip other steps.

6.10. If *accessDeniedIdentifierTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "Reason", FAIL the test and skip other steps.

6.11. If *accessDeniedIdentifierTopic*.MessageDescription.Data.SimpleItemDescription with Name = "Reason" does not have Type = "xs:string", FAIL the test and skip other steps.

7. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.3 with the following input and output parameters

- in **"tns1:AccessControl/Denied/Identifier"** - Notification Topic
- out *s* - Subscription reference
- out *currentTime* - current time for the DUT
- out *terminationTime* - Subscription termination time

8. Test Operator will invoke **tns1:AccessControl/Denied/Identifier** event for any Access Point.
9. ONVIF Client retrieves and checks **tns1:AccessControl/Denied/Identifier** event for the specified credential identifiers by following the procedure mentioned in Annex A.7 with the following input and output parameters
   - in *s* - Subscription reference
   - in *currentTime* - current time for the DUT
   - in *terminationTime* - subscription termination time
10. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.4 with the following input and output parameters
    - in *s* - Subscription reference

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**

## 4.10 Duress events

### 4.10.1 ACCESS CONTROL - DURESS

**Test Case ID:** ACCESSCONTROL-10-1-2

**Specification Coverage:** tns:AccessControl/Duress (Access Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification)

**Feature Under Test:** tns:AccessControl/Duress

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

**Test Purpose:** To verify tns1:AccessControl/Duress event generation and to verify tns1:AccessControl/Duress event format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point with Duress capability is configured and added to the DUT. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get a complete list of access points from the DUT (see Annex A.1).
4. If there is no Access Point with Capabilities.Duress = "true" on complete list of access points, fail the test and skip other steps.
5. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
6. Verify the **GetEventPropertiesResponse** message from the DUT.
7. Check if there is an event with Topic tns1:AccessControl/Duress. If there is no event with such Topic, skip other steps, fail the test and go to the next test.
8. Check that this event is not a Property event (MessageDescription.IsProperty = "false").
9. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".
10. Check that if this event contains Data.SimpleItemDescription item with Name = "CredentialToken", then it has Type = "pt:ReferenceToken".
11. Check that if this event contains Data.SimpleItemDescription item with Name = "CredentialHolderName", then it has Type = " xs:string".
12. Check that this event contains Data.SimpleItemDescription item with Name = "Reason" and Type = "xs:string".
13. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:AccessControl/Duress Topic as Filter and an InitialTerminationTime of timeout1.

14. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.

15. Test Operator will invoke tns1:AccessControl/Duress event for any Access Point with Capabilities.Duress = "true".

16. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.

17. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. If no NotificationMessage were received repeat step 16.

18. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).

19. Verify that TopicExpression is equal to tns1:AccessControl/Duress for the Notification message.

20. Verify that the notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value is equal to one of existing Access Point Tokens with Capabilities.Duress = "true" (e.g. complete list of access points contains Access Point with the same token and this Access Point has Capabilities.Duress = "true").

21. Verify that the notification contains Data.SimpleItem item with Name = "Reason" and Value with type is equal to xs:string.

22. Verify that the notification which contains Data.SimpleItem item with Name = «CredentialToken» contains Value with type is equal to pt:ReferenceToken.

23. Verify that the notification which contains Data.SimpleItem item with Name = "CredentialHolderName" contains Value with type is equal to xs:string.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a **PullMessagesResponse** message that contains one tns1:AccessControl/Duress event.

- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid AccessPointToken value, invalid CredentialToken, invalid CredentialHolderName, invalid Reason).
- The DUT did not return at least one Access Point at step 3.
- The DUT did not return valid Topic tns1:AccessControl/Duress in **GetEventPropertiesResponse**.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** Test will fail, if no Notification message is received at step 17 for a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

## 4.11 Consistency

### 4.11.1 GET AREA INFO LIST AND GET ACCESS POINT INFO LIST CONSISTENCY

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

**Specification Coverage:** GetAccessPointInfoList (ONVIF Access Control Service Specification), GetAreaInfoList (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAccessPointInfoList, GetAreaInfoList

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify that Area Info List contains all Areas from Access Point Info List.

**Pre-Requisite:** Access Control Service was received from the DUT. Access Point Entity is supported by the DUT. At least one Access Point is configured and added to the DUT. At least one Area is configured and added to the DUT, if the DUT supports Areas.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Get a complete list of access points from the DUT (see Annex A.1)
4. Get a complete list of areas from the DUT (see Annex A.2)
5. Verify that the complete list of areas contains all Areas from AreaTo and AreaFrom elements of a complete access points list.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT returned at least one value in AreaTo or AreaFrom element of complete access points list that was not listed in the complete list of areas.
- The DUT did not return at least one Access Point at step 3.

## 4.11.2 ACCESS POINT CAPABILITIES CONSISTENCY

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

**Specification Coverage:** AccessPointCapabilities (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAccessPointInfoList

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify that AccessPoint capabilities consistency.

**Pre-Requisite:** Access Control Service was received from the DUT. Access Point Entity is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client retrieves a complete access point information list by following the procedure mentioned in Annex A.1 with the following input and output parameters
   - out *accessPointInfoCompleteList* - access point info list
4. For each Access Point Info (*accessPointInfo*) from *accessPointInfoCompleteList*

   4.1. If *accessPointInfo*.Capabilities.IdentifierAccess = true and *accessPointInfo*.Capabilities.ExternalAuthorization != true, FAIL the test and skip other steps.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- DUT does not pass all assertions.

## 4.12 Access Point Configuration

### 4.12.1 ACCESS CONTROL - ADD OR CHANGE ACCESS POINT EVENT

**Test Case ID:** ACCESSCONTROL-12-1-1

**Specification Coverage:** tns1:Configuration/AccessPoint/Changed (Access Control Service Specification), GetAccessPointInfo (ONVIF Access Control Service Specification)

**Feature Under Test:** tns1:Configuration/AccessPoint/Changed

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

**Test Purpose:** To verify tns1:Configuration/AccessPoint/Changed event generation after adding new access point or changing access point configuration to the DUT and to verify: tns1:Configuration/AccessPoint/Changed event format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. The DUT allows adding or changing Access Points. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
4. Verify the **GetEventPropertiesResponse** message from the DUT.
5. Check if there is an event with Topic tns1:Configuration/AccessPoint/Changed. If there is no event with such Topic fail the test and skip other steps.
6. Check that this event isn’t a Property event (MessageDescription.IsProperty = "false").
7. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".
8. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Configuration/AccessPoint/Changed Topic as Filter and an InitialTerminationTime of timeout1.
9. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
10. Test Operator will add Access Point or change Access Point configuration.
11. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.
12. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. Repeat step 11 until Notification received.
13. Verify received Notification messages (correct value for UTC time, TopicExpression and wsnt:Message).
14. Verify that TopicExpression is equal to tns1:Configuration/AccessPoint/Changed for received message.
15. Verify that notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value with type is equal to tdc:ReferenceToken.
16. ONVIF Client will invoke **GetAccessPointInfo** request (Token from Notification message) to retrieve subset of AccessPoint Information from the DUT.
17. Verify the **GetAccessPointInfoResponse** message from the DUT. Verify that requested Access Point was returned.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a **GetAccessPointInfoResponse** with specified Access Point.
- The DUT did not send a Notification message that contains an event tns1:Configuration/AccessPoint/Changed with valid AccessPointToken.
- The DUT sent an invalid Notification message (invalid AccessPointToken value).

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** ONVIF Client on step 12 will wait Notification messages until expected notification is received or Operation Delay has expired.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

### 4.12.2 ACCESS CONTROL - REMOVE ACCESS POINT EVENT

**Test Case ID:** ACCESSCONTROL-12-1-2

**Specification Coverage:** tns1:Configuration/AccessPoint/Removed (Access Control Service Specification), GetAccessPointInfo (ONVIF Access Control Service Specification)

**Feature Under Test:** tns1:Configuration/AccessPoint/Removed

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

**Test Purpose:** To verify tns1:Configuration/AccessPoint/Removed event generation after removing access point configuration to the DUT and to verify tns1:Configuration/AccessPoint/Removed event format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. At least one Access Point is configured and added to the DUT. The DUT supports Access Points remove. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
4. Verify the **GetEventPropertiesResponse** message from the DUT.
5. Check if there is an event with Topic tns1:Configuration/AccessPoint/Removed. If there is no event with such Topic, fail the test and skip other steps.
6. Check that this event isn’t a Property event (MessageDescription.IsProperty = "false").
7. Check that this event contains Source.SimpleItemDescription item with Name = "AccessPointToken" and Type = "pt:ReferenceToken".
8. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Configuration/AccessPoint/Removed Topic as Filter and an InitialTerminationTime of timeout1.
9. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
10. Test Operator will remove Access Point.
11. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.
12. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. Repeat step 11 until Notification received.

13. Verify received Notification messages (correct value for UTC time, TopicExpression and wsnt:Message).

14. Verify that TopicExpression is equal to tns1:Configuration/AccessPoint/Removed for received message.

15. Verify that notification contains Source.SimpleItem item with Name = "AccessPointToken" and Value with type is equal to tdc:ReferenceToken.

16. ONVIF Client will invoke **GetAccessPointInfo** request (Token from Notification message) to retrieve subset of Access Point Information from the DUT.

17. Verify the **GetAccessPointInfoResponse** message from the DUT. Check that empty list was returned.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a **GetAccessPointInfoResponse** with empty list.
- The DUT did not send a Notification message that contains an event tns1:Configuration/AccessPoint/Removed with valid AccessPointToken.
- The DUT sent an invalid Notification message (invalid AccessPointToken value).

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** ONVIF Client on step 12 will wait Notification messages until expected notification is received or Operation Delay has expired.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

## 4.13 Area Configuration

### 4.13.1 ACCESS CONTROL - ADD OR CHANGE AREA EVENT

**Test Case ID:** ACCESSCONTROL-13-1-1

**Specification Coverage:** tns1:Configuration/Area/Change (Access Control Service Specification), GetAreaInfo (ONVIF Access Control Service Specification)

**Feature Under Test:** tns1:Configuration/Area/Change

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

**Test Purpose:** To verify: tns1:Configuration/Area/Change event generation after adding new area or changing area configuration to the DUT and to verify: tns1:Configuration/Area/Change event format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. The DUT allows adding or changing Area. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
4. Verify the **GetEventPropertiesResponse** message from the DUT.
5. Check if there is an event with Topic tns1:Configuration/Area/Change. If there is no event with such Topic fail the test and skip other steps.

6. Check that this event isn't a Property event (MessageDescription.IsProperty = "false").
7. Check that this event contains Source.SimpleItemDescription item with Name = "AreaToken" and Type = "pt:ReferenceToken".
8. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Configuration/Area/Changed Topic as Filter and an InitialTerminationTime of timeout1.
9. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
10. Test Operator will add Area or change Area configuration.
11. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.
12. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. Repeat step 11 until Notification received.
13. Verify received Notification messages (correct value for UTC time, TopicExpression and wsnt:Message).
14. Verify that TopicExpression is equal to tns1:Configuration/Area/Change for received message.
15. Verify that notification contains Source.SimpleItem item with Name = "AreaToken" and Value with type is equal to tdc:ReferenceToken.
16. ONVIF Client will invoke GetAreaInfo request (Token from Notification message) to retrieve subset of Area Information from the DUT.
17. Verify the **GetAreaInfoResponse** message from the DUT. Verify that requested Area was returned.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a **GetAreaInfoResponse** with specified Area.

- The DUT did not send a Notification message that contains an event tns1:Configuration/Area/Change with valid AreaToken.
- The DUT sent an invalid Notification message (invalid AreaToken value).

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** ONVIF Client on step 12 will wait Notification messages until expected notification is received or Operation Delay after last notification has expired.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

### 4.13.2 ACCESS CONTROL - REMOVE AREA EVENT

**Test Case ID:** ACCESSCONTROL-13-1-2

**Specification Coverage:** tns1:Configuration/Area/Removed (Access Control Service Specification), GetAreaInfo (ONVIF Access Control Service Specification)

**Feature Under Test:** tns1:Configuration/Area/Removed

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

**Test Purpose:** To verify tns1:Configuration/Area/Removed event generation after removing Area configuration to the DUT and to verify tns1:Configuration/Area/Removed event format.

**Pre-Requisite:** Event Service was received from the DUT. Access Control Service was received from the DUT. At least one Area is configured and added to the DUT. The DUT supports Area remove. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
4. Verify the **GetEventPropertiesResponse** message from the DUT.
5. Check if there is an event with Topic tns1:Configuration/Area/Removed. If there is no event with such Topic fail the test and skip other steps.
6. Check that this event isn't a Property event (MessageDescription.IsProperty = "false").
7. Check that this event contains Source.SimpleItemDescription item with Name = "AreaToken" and Type = "pt:ReferenceToken".
8. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Configuration/Area/Removed Topic as Filter and an InitialTerminationTime of timeout1.
9. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
10. Test Operator will remove Area.
11. ONVIF Client will invoke **PullMessages** request with a PullMessagesTimeout of 20s and a MessageLimit of 1.
12. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage. Repeat step 11 until Notification received.
13. Verify received Notification messages (correct value for UTC time, TopicExpression and wsnt:Message).
14. Verify that TopicExpression is equal to tns1:Configuration/Area/Removed for received message.
15. Verify that notification contains Source.SimpleItem item with Name = "AreaToken" and Value with type is equal to tdc:ReferenceToken.
16. ONVIF Client will invoke **GetAreaInfo** request (Token from Notification message) to retrieve subset of Area Information from the DUT.
17. Verify the **GetAreaInfoResponse** message from the DUT. Check that empty list was returned.

**Test Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a **GetAreaInfoResponse** with empty list.
- The DUT did not send a Notification message that contains an event tns1:Configuration/Area/Removed with valid AreaToken.
- The DUT sent an invalid Notification message (invalid AreaToken value).

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** ONVIF Client on step 12 will wait Notification messages until expected notification is received or Operation Delay after last notification has expired.

**Note:** The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

**Note:** The **Renew** has to be used for renew subscription during test, if InitialTerminationTime expires.

**Note:** If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

**Note:** If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the **Renew** request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

**Note:** timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

## 4.14 Access Point Management

### 4.14.1 GET ACCESS POINTS

**Test Case ID:** ACCESSCONTROL-14-1-1

**Specification Coverage:** AccessPointInfo (ONVIF Access Control Service Specification), AccessPoint (ONVIF Access Control Service Specification), GetAccessPoints command (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAccessPoints

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get Access Points.

**Pre-Requisite:** Access Control Service is received from the DUT. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability.

**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 Points (out *accessPointCompleteList*) by following the procedure mentioned in Annex A.12.
4. If *accessPointCompleteList* is empty, skip other steps.
5. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.13.
6. Set the following:
   - *tokenList* := [subset of *accessPointCompleteList*.token values with items number equal to *cap*.MaxLimit]
7. ONVIF client invokes **GetAccessPoints** with parameters
   - Token list := *tokenList*
8. The DUT responds with **GetAccessPointsResponse** message with parameters
   - AccessPoint list =: *accessPointList1*
9. If *accessPointList1* does not contain AccessPoint item for each token from *tokenList*, FAIL the test and skip other steps.
10. If *accessPointList1* contains at least two AccessPoint items with equal token, FAIL the test and skip other steps.
11. If *accessPointList1* contains other AccessPoint items than listed in *tokenList*, FAIL the test and skip other steps.
12. For each AccessPoint.token *token* from *accessPointCompleteList* repeat the following steps:

12.1. ONVIF client invokes **GetAccessPoints** with parameters

- Token[0] := *token*

12.2. The DUT responds with **GetAccessPointsResponse** message with parameters

- AccessPoint list =: *accessPointList2*

12.3. If *accessPointList2* does not contain only one AccessPoint item with token equal to *token*, FAIL the test and skip other steps.

12.4. If *accessPointList2*[0] item does not have equal field values to *accessPointCompleteList*[token = *token*] item, FAIL the test and skip other steps.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

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

**Note:** The following fields are compared at step 12.4:

- AccessPoint:
  - token
  - Name
  - Description
  - AreaFrom
  - AreaTo
  - EntityType
  - Entity
  - Capabilities
    - DisableAccessPoint

- Duress
- AnonymousAccess
- AccessTaken
- ExternalAuthorization
- SupportedRecognitionTypes
- SupportedSecurityLevels

- AuthenticationProfileToken

## 4.14.2 GET ACCESS POINT LIST - LIMIT

**Test Case ID:** ACCESSCONTROL-14-1-2

**Specification Coverage:** AccessPointInfo (ONVIF Access Control Service Specification), AccessPoint (ONVIF Access Control Service Specification), GetAccessPointList command (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAccessPointList

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get Access Point List using Limit.

**Pre-Requisite:** Access Control Service is received from the DUT. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability.

**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.13.
4. ONVIF client invokes **GetAccessPointList** with parameters
    - Limit := 1
    - StartReference skipped

5. The DUT responds with **GetAccessPointListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - AccessPoint list =: *accessPointList1*
6. If *accessPointList1* contains more access point 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 **GetAccessPointList** with parameters
   - Limit := *cap*.MaxLimit
   - StartReference skipped
9. The DUT responds with **GetAccessPointListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - AccessPoint list =: *accessPointList2*
10. If *accessPointList2* contains more AccessPoint 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 **GetAccessPointList** with parameters
    - Limit := *limit*
    - StartReference skipped
14. The DUT responds with **GetAccessPointListResponse** message with parameters
    - NextStartReference =: *nextStartReference*
    - AccessPoint list =: *accessPointList3*
15. If *accessPointList3* contains more AccessPoint items than *limit*, FAIL the test and skip other steps.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 4.14.3 GET ACCESS POINT LIST - START REFERENCE AND LIMIT

**Test Case ID:** ACCESSCONTROL-14-1-3

**Specification Coverage:** AccessPointInfo (ONVIF Access Control Service Specification), AccessPoint (ONVIF Access Control Service Specification), GetAccessPointList command (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAccessPointList

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get AccessPoint List using StartReference and Limit.

**Pre-Requisite:** Access Control Service is received from the DUT. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability.

**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.13.
4. ONVIF client invokes **GetAccessPointList** with parameters
    - Limit := *cap*.MaxLimit
    - StartReference skipped
5. The DUT responds with **GetAccessPointListResponse** message with parameters
    - NextStartReference =: *nextStartReference*
    - AccessPoint list =: *accessPointCompleteList1*

6. If *accessPointCompleteList1* contains more AccessPoint 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 **GetAccessPointList** with parameters
   - Limit := *cap*.MaxLimit
   - StartReference := *nextStartReference*

   7.2. The DUT responds with **GetAccessPointListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - AccessPoint list =: *accessPointListPart*

   7.3. If *accessPointListPart* contains more AccessPoint items than *cap*.MaxLimit, FAIL the test and skip other steps.

   7.4. Set the following:
   - *accessPointCompleteList1* := *accessPointCompleteList1* + *accessPointListPart*
8. If *accessPointCompleteList1* contains at least two AccessPoint items 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 points info (out *accessPointInfoCompleteList*) by following the procedure mentioned in Annex A.1.

   9.2. If *accessPointCompleteList1* does not contain all access points from *accessPointInfoCompleteList*, FAIL the test and skip other steps.

   9.3. If *accessPointCompleteList1* contains access points other than access points from *accessPointInfoCompleteList*, FAIL the test and skip other steps.

   9.4. For each AccessPointInfo.token *token* from *accessPointInfoCompleteList* repeat the following steps:

   9.4.1. If *accessPointCompleteList1*[token = *token*] item does not have equal field values to *accessPointInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.

   9.5. Skip other steps.
10. ONVIF client invokes **GetAccessPointList** with parameters

- Limit := 1
- StartReference skipped

11. The DUT responds with **GetAccessPointListResponse** message with parameters
    - NextStartReference =: *nextStartReference*
    - AccessPoint list =: *accessPointCompleteList2*
12. If *accessPointCompleteList2* contains more AccessPoint 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 **GetAccessPointList** with parameters

    - Limit := 1
    - StartReference := *nextStartReference*

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

    - NextStartReference =: *nextStartReference*
    - AccessPoint list =: *accessPointListPart*

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

    13.4. Set the following:

    - *accessPointCompleteList2* := *accessPointCompleteList2* + *accessPointListPart*
14. If *accessPointCompleteList2* contains at least two AccessPoint items with equal token, FAIL the test and skip other steps.
15. If *accessPointCompleteList2* does not contain all access points from *accessPointCompleteList1*, FAIL the test and skip other steps.
16. If *accessPointCompleteList2* contains access points other than access points from *accessPointCompleteList1*, 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 points info (out *accessPointInfoCompleteList*) by following the procedure mentioned in Annex A.1.

17.2. If *accessPointCompleteList2* does not contain all access points from *accessPointInfoCompleteList*, FAIL the test and skip other steps.

17.3. If *accessPointCompleteList2* contains access points other than access points from *accessPointInfoCompleteList*, FAIL the test and skip other steps.

17.4. For each AccessPointInfo.token *token* from *accessPointInfoCompleteList* repeat the following steps:

17.4.1. If *accessPointCompleteList2*[token = *token*] item does not have equal field values to *accessPointInfoCompleteList*[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 **GetAccessPointList** with parameters

- Limit := *limit*
- StartReference skipped

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

- NextStartReference =: *nextStartReference*
- AccessPoint list =: *accessPointCompleteList3*

21. If *accessPointCompleteList3* contains more AccessPoint 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 **GetAccessPointList** with parameters

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

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

- NextStartReference =: *nextStartReference*
- AccessPoint list =: *accessPointListPart*

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

22.4. Set the following:

- *accessPointCompleteList3* := *accessPointCompleteList3* + *accessPointListPart*

23. If *accessPointCompleteList3* contains at least two AccessPoint items with equal token, FAIL the test and skip other steps.

24. If *accessPointCompleteList3* does not contain all access points from *accessPointCompleteList1*, FAIL the test and skip other steps.

25. If *accessPointCompleteList3* contains access points other than access points from *accessPointCompleteList1*, FAIL the test and skip other steps.

26. ONVIF Client retrieves a complete list of access points info (out *accessPointInfoCompleteList*) by following the procedure mentioned in Annex A.1.

27. If *accessPointCompleteList3* does not contain all access points from *accessPointInfoCompleteList*, FAIL the test and skip other steps.

28. If *accessPointCompleteList3* contains access points other than access points from *accessPointInfoCompleteList*, FAIL the test and skip other steps.

29. For each AccessPointInfo.token *token* from *accessPointInfoCompleteList* repeat the following steps:

29.1. If *accessPointCompleteList3*[token = *token*] item does not have equal field values to *accessPointInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

**Note:** The following fields are compared at step 9.4.1, 17.4.1, and 29.1:

- AccessPointInfo:
  - token
  - Name

- Description
- AreaFrom
- AreaTo
- EntityType
- Entity
- Capabilities
  - DisableAccessPoint
  - Duress
  - AnonymousAccess
  - AccessTaken
  - ExternalAuthorization
  - SupportedRecognitionTypes
  - SupportedSecurityLevels

### 4.14.4 GET ACCESS POINT LIST - NO LIMIT

**Test Case ID:** ACCESSCONTROL-14-1-4

**Specification Coverage:** AccessPointInfo (ONVIF Access Control Service Specification), AccessPoint (ONVIF Access Control Service Specification), GetAccessPointList command (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAccessPointList

**WSDL Reference:** accesscontrol.wsdl

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

**Pre-Requisite:** Access Control Service is received from the DUT. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability.

**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.13.
4. ONVIF client invokes **GetAccessPointList** with parameters
   - Limit skipped
   - StartReference skipped
5. The DUT responds with **GetAccessPointListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - AccessPoint list =: *accessPointCompleteList*
6. If *accessPointCompleteList* contains more AccessPoint 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 **GetAccessPointList** with parameters
   - Limit skipped
   - StartReference := *nextStartReference*

   7.2. The DUT responds with **GetAccessPointListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - AccessPoint list =: *accessPointListPart*

   7.3. If *accessPointListPart* contains more AccessPoint items than *cap*.MaxLimit, FAIL the test and skip other steps.

   7.4. Set the following:
   - *accessPointCompleteList* := *accessPointCompleteList* + *accessPointListPart*
8. If *accessPointCompleteList* contains at least two AccessPoint items with equal token, FAIL the test.
9. ONVIF Client retrieves a complete list of access points (out *accessPointInfoCompleteList*) by following the procedure mentioned in Annex A.1.

10. If *accessPointCompleteList* does not contain all access points from *accessPointInfoCompleteList*, FAIL the test and skip other steps.

11. If *accessPointCompleteList* contains access points other than access points from *accessPointInfoCompleteList*, FAIL the test and skip other steps.

12. For each AccessPointInfo.token *token* from *accessPointInfoCompleteList* repeat the following steps:

    12.1. If *accessPointCompleteList*[token = *token*] item does not have equal field values to *accessPointInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

**Note:** The following fields are compared at step 12.1:

- AccessPointInfo:
  - token
  - Name
  - Description
  - AreaFrom
  - AreaTo
  - EntityType
  - Entity
  - Capabilities
    - DisableAccessPoint
    - Duress
    - AnonymousAccess
    - AccessTaken

- ExternalAuthorization
- SupportedRecognitionTypes
- SupportedSecurityLevels

### 4.14.5 CREATE ACCESS POINT (ACCESS POINT CAPABILITIES TRUE)

**Test Case ID:** ACCESSCONTROL-14-1-5

**Specification Coverage:** CreateAccessPoint command (ONVIF Access Control Service Specification), AccessPoint (ONVIF Access Control Service Specification)

**Feature Under Test:** CreateAccessPoint

**WSDL Reference:** accesscontrol.wsdl, door.wsdl, event.wsdl

**Test Purpose:** To verify creation of access point and generating of access point changed notifications.

**Pre-Requisite:** Access Control Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional access point. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves a complete list of access points by following the procedure mentioned in Annex A.12 with the following output parameters
    - out *accessPointCompleteList1* - complete access point list
4. ONVIF Client checks free storage for additional access point by following the procedure mentioned in Annex A.15 with the following input and output parameters
    - in *accessPointCompleteList1* - complete access point list
    - out (optional) *accessPointToRestore* - deleted access point

5. ONVIF Client retrieves references to Areas existing on the DUT by following the procedure mentioned in Annex A.18 with the following input and output parameters

   - out (optional) *areaToken1* - token of the 1st Area
   - out (optional) *areaToken2* - token of the 2nd Area

6. If Door Control Service is supported by the DUT

   6.1. ONVIF Client gets the Door Control service capabilities (out *cap*) by following the procedure mentioned in Annex A.22.

   6.2. If *cap*.MaxDoors >0 or skipped:

   6.2.1. ONVIF Client retrieves references to a Door existing on the DUT by following the procedure mentioned in Annex A.20 with the following input and output parameters

   - out (optional) *doorToken* - token of the door

7. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.3 with the following input and output parameters

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

8. ONVIF client invokes **CreateAccessPoint** with parameters

   - AccessPoint.token := ""
   - AccessPoint.Name := "Test Name"
   - AccessPoint.Description := "Test Description"
   - AccessPoint.AreaFrom := *areaToken1* if it was returned at step 5, otherwise skipped
   - AccessPoint.AreaTo := *areaToken2* if it was returned at step 5, otherwise skipped
   - AccessPoint.EntityType := "tdc:Door" if door token was returned at step 6.2.1, otherwise value taken from UI.
   - AccessPoint.Entity := *doorToken* if it was returned at step 6.2.1, otherwise value taken from UI.

- AccessPoint.Capabilities.DisableAccessPoint := true
- AccessPoint.Capabilities.Duress := true
- AccessPoint.Capabilities.AnonymousAccess := true
- AccessPoint.Capabilities.AccessTaken := true
- AccessPoint.Capabilities.ExternalAuthorization := true
- AccessPoint.Capabilities.SupportedRecognitionTypes skipped
- AccessPoint.Capabilities.IdentifierAccess := true
- AccessPoint.Capabilities.SupportedSecurityLevels skipped
- AccessPoint.AuthenticationProfileToken skipped

9. The DUT responds with **CreateAccessPointResponse** message with parameters
   - Token =: *accessPointToken*
10. ONVIF Client retrieves and checks **tns1:Configuration/AccessPoint/Changed** event for the specified AccessPoint token by following the procedure mentioned in Annex A.17 with the following input and output parameters
    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *accessPointToken* - Access Point token
11. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.4 with the following input and output parameters
    - in *s* - Subscription reference
12. ONVIF Client retrieves an access point by following the procedure mentioned in Annex A.24 with the following input and output parameters
    - in *accessPointToken* - access point token
    - out *accessPointList* - access point list
13. If *accessPointList*[0] item does not have equal field values to values from step 8, FAIL the test and go step 20.

14. ONVIF Client retrieves an access point info by following the procedure mentioned in Annex A.25 with the following input and output parameters

    - in *accessPointToken* - access point token
    - out *accessPointInfoList* - access point info list

15. If *accessPointInfoList*[0] item does not have equal field values to values from step 8, FAIL the test and go step 20.

16. ONVIF Client retrieves a complete access point information list by following the procedure mentioned in Annex A.1 with the following input and output parameters

    - out *accessPointInfoCompleteList* - access point info list

17. If *accessPointInfoCompleteList* does not have AccessPointInfo[token = *accessPointToken*] item with equal field values to values from step 8, FAIL the test and go step 20.

18. ONVIF Client retrieves a complete list of access points by following the procedure mentioned in Annex A.12 with the following input and output parameters

    - out *accessPointCompleteList2* - access point list

19. If *accessPointCompleteList2* does not have AccessPoint[token = *accessPointToken*] item with equal field values to values from step 8, FAIL the test and go step 20.

20. ONVIF Client deletes the access point by following the procedure mentioned in Annex A.26 to restore DUT configuration with the following input and output parameters

    - in *accessPointToken* - access point token

21. ONVIF Client restores access point deleted at step 4 if any.

22. ONVIF Client deletes door created at step 6.2.1 if any.

23. ONVIF Client deletes areas created at step 5 if any.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

**Note:** The following fields are compared at steps 13, 19:

- AccessPoint:
  - token
  - Name
  - Description
  - AreaFrom
  - AreaTo
  - EntityType
  - Entity
  - Capabilities
    - DisableAccessPoint
    - Duress
    - AnonymousAccess
    - AccessTaken
    - ExternalAuthorization
    - SupportedRecognitionTypes
    - IdentifierAccess
    - SupportedSecurityLevels
  - AuthenticationProfileToken

**Note:** The following fields are compared at steps 15, 17:

- AccessPoint:
  - token
  - Name
  - Description
  - AreaFrom
  - AreaTo

- EntityType
- Entity
- Capabilities
  - DisableAccessPoint
  - Duress
  - AnonymousAccess
  - AccessTaken
  - ExternalAuthorization
  - SupportedRecognitionTypes
  - IdentifierAccess
  - SupportedSecurityLevels

### 4.14.6 CREATE ACCESS POINT (ACCESS POINT CAPABILITIES FALSE)

**Test Case ID:** ACCESSCONTROL-14-1-6

**Specification Coverage:** CreateAccessPoint command (ONVIF Access Control Service Specification), AccessPoint (ONVIF Access Control Service Specification)

**Feature Under Test:** CreateAccessPoint

**WSDL Reference:** accesscontrol.wsdl, door.wsdl, event.wsdl

**Test Purpose:** To verify creation of access point and generating of access point changed notifications.

**Pre-Requisite:** Access Control Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional access point. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of access points by following the procedure mentioned in Annex A.12 with the following output parameters

   - out *accessPointCompleteList1* - complete access point list

4. ONVIF Client checks free storage for additional access point by following the procedure mentioned in Annex A.15 with the following input and output parameters

   - in *accessPointCompleteList1* - complete access point list

   - out (optional) *accessPointToRestore* - deleted access point

5. If Door Control Service is supported by the DUT

   5.1. ONVIF Client gets the Door Control service capabilities (out *cap*) by following the procedure mentioned in Annex A.22.

   5.2. If *cap*.MaxDoors >0 or skipped:

      5.2.1. ONVIF Client retrieves references to a Door existing on the DUT by following the procedure mentioned in Annex A.20 with the following input and output parameters

      - out (optional) *doorToken* - token of the door

6. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.3 with the following input and output parameters

   - in **tns1:Configuration/AccessPoint/Changed** - Notification Topic

   - out *s* - Subscription reference

   - out *currentTime* - current time for the DUT

   - out *terminationTime* - Subscription termination time

7. ONVIF client invokes **CreateAccessPoint** with parameters

   - AccessPoint.token := ""

   - AccessPoint.Name := "Test Name"

   - AccessPoint.Description skipped

- AccessPoint.AreaFrom skipped
- AccessPoint.AreaTo skipped
- AccessPoint.EntityType := "tdc:Door" if door token was returned at step 6.2.1, otherwise value taken from UI.
- AccessPoint.Entity := *doorToken* if it was returned at step 6.2.1, otherwise value taken from UI.
- AccessPoint.Capabilities.DisableAccessPoint := false
- AccessPoint.Capabilities.Duress := false
- AccessPoint.Capabilities.AnonymousAccess := false
- AccessPoint.Capabilities.AccessTaken := false
- AccessPoint.Capabilities.ExternalAuthorization := false
- AccessPoint.Capabilities.SupportedRecognitionTypes skipped
- AccessPoint.Capabilities.IdentifierAccess := false
- AccessPoint.Capabilities.SupportedSecurityLevels skipped
- AccessPoint.AuthenticationProfileToken skipped

8. The DUT responds with **CreateAccessPointResponse** message with parameters
   - Token =: *accessPointToken*
9. ONVIF Client retrieves and checks **tns1:Configuration/AccessPoint/Changed** event for the specified AccessPoint token by following the procedure mentioned in Annex A.17 with the following input and output parameters
   - in *s* - Subscription reference
   - in *currentTime* - current time for the DUT
   - in *terminationTime* - subscription termination time
   - in *accessPointToken* - Access Point token
10. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.4 with the following input and output parameters

- in *s* - Subscription reference

11. ONVIF Client retrieves an access point by following the procedure mentioned in Annex A.24 with the following input and output parameters

    - in *accessPointToken* - access point token
    - out *accessPointList* - access point list

12. If *accessPointList*[0] item does not have equal field values to values from step 7, FAIL the test and go step 19.

13. ONVIF Client retrieves an access point info by following the procedure mentioned in Annex A.25 with the following input and output parameters

    - in *accessPointToken* - access point token
    - out *accessPointInfoList* - access point info list

14. If *accessPointInfoList*[0] item does not have equal field values to values from step 7, FAIL the test and go step 19.

15. ONVIF Client retrieves a complete access point information list by following the procedure mentioned in Annex A.1 with the following input and output parameters

    - out *accessPointInfoCompleteList* - access point info list

16. If *accessPointInfoCompleteList* does not have AccessPointInfo[token = *accessPointToken*] item with equal field values to values from step 7, FAIL the test and go step 19.

17. ONVIF Client retrieves a complete list of access points by following the procedure mentioned in Annex A.12 with the following input and output parameters

    - out *accessPointCompleteList2* - access point list

18. If *accessPointCompleteList2* does not have AccessPoint[token = *accessPointToken*] item with equal field values to values from step 7, FAIL the test and go step 19.

19. ONVIF Client deletes the access point by following the procedure mentioned in Annex A.26 to restore DUT configuration with the following input and output parameters

    - in *accessPointToken* - access point token

20. ONVIF Client restores access point deleted at step 4 if any.

21. ONVIF Client deletes door created at step 6.2.1 if any.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

**Note:** The following fields are compared at steps 12, 18:

- AccessPoint:
  - token
  - Name
  - Description
  - AreaFrom
  - AreaTo
  - EntityType
  - Entity
  - Capabilities
    - DisableAccessPoint
    - Duress
    - AnonymousAccess
    - AccessTaken
    - ExternalAuthorization
    - SupportedRecognitionTypes
    - IdentifierAccess
    - SupportedSecurityLevels
  - AuthenticationProfileToken

**Note:** The following fields are compared at steps 14, 16:

- AccessPoint:
  - token
  - Name
  - Description
  - AreaFrom
  - AreaTo
  - EntityType
  - Entity
  - Capabilities
    - DisableAccessPoint
    - Duress
    - AnonymousAccess
    - AccessTaken
    - ExternalAuthorization
    - SupportedRecognitionTypes
    - IdentifierAccess
    - SupportedSecurityLevels

### 4.14.7 MODIFY ACCESS POINT

**Test Case ID:** ACCESSCONTROL-14-1-7

**Specification Coverage:** AccessPointInfo (ONVIF Access Control Service Specification), AccessPoint (ONVIF Access Control Service Specification), ModifyAccessPoint command (ONVIF Access Control Service Specification)

**Feature Under Test:** ModifyAccessPoint

**WSDL Reference:** accesscontrol.wsdl, door.wsdl, event.wsdl

**Test Purpose:** To verify modifying of access point and generating of access point changed notifications.

**Pre-Requisite:** Access Control Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional Access Point. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves a complete list of access points by following the procedure mentioned in Annex A.12 with the following output parameters
   - out *accessPointCompleteList1* - complete access point list
4. ONVIF Client checks free storage for additional access point by following the procedure mentioned in Annex A.15 with the following input and output parameters
   - in *accessPointCompleteList1* - complete access point list
   - out (optional) *accessPointToRestore* - deleted access point
5. ONVIF Client retrieves references to Areas existing on the DUT by following the procedure mentioned in Annex A.18 with the following input and output parameters
   - out (optional) *areaToken1* - token of the 1st Area
   - out (optional) *areaToken2* - token of the 2nd Area
6. If Door Control Service is supported by the DUT

   6.1. ONVIF Client gets the Door Control service capabilities (out *cap*) by following the procedure mentioned in Annex A.22.

   6.2. If *cap*.MaxDoors >0 or skipped:

   6.2.1. ONVIF Client retrieves references to a Door existing on the DUT by following the procedure mentioned in Annex A.20 with the following input and output parameters

   - out (optional) *doorToken* - token of the door
7. ONVIF Client creates access point by following the procedure mentioned in Annex A.27 with the following input and output parameters

- in "Test Name1" - Access Point Name
- in "Test Description1" - Access Point Description
- in *areaToken1* if it was returned at step 5, otherwise skipped - Area From Token
- in *areaToken2* if it was returned at step 5, otherwise skipped - Area To Token
- in "tdc:Door" if door token was returned at step 6.2.1, otherwise value taken from UI - Access Point Entity Type
- in *doorToken* if it was returned at step 6.2.1, otherwise value taken from UI - Access Point Entity Token
- in false - Disable Access Point Capabilities
- in false - Access Point Duress Capabilities
- in false - Access Point Anonymous Access Capabilities
- in false - Access Point Access Taken Capabilities
- in false - Access Point External Authorization Capabilities
- in false - Access Point Identifier Access Capabilities
- out *accessPointToken* - Access Point Token

8. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.3 with the following input and output parameters
   - in **tns1:Configuration/AccessPoint/Changed** - Notification Topic
   - out *s* - Subscription reference
   - out *currentTime* - current time for the DUT
   - out *terminationTime* - Subscription termination time
9. ONVIF client invokes **ModifyAccessPoint** with parameters
   - AccessPoint.token := *accessPointToken*
   - AccessPoint.Name := "Test Name2"
   - AccessPoint.Description := "Test Description2"
   - AccessPoint.AreaFrom := *areaToken2* if it was returned at step 5, otherwise skipped

- AccessPoint.AreaTo := *areaToken1* if it was returned at step 5, otherwise skipped
- AccessPoint.EntityType := "tdc:Door" if door token was returned at step 6.2.1, otherwise value taken from UI
- AccessPoint.Entity := *doorToken* if it was returned at step 6.2.1, otherwise value taken from UI
- AccessPoint.Capabilities.DisableAccessPoint := true
- AccessPoint.Capabilities.Duress := true
- AccessPoint.Capabilities.AnonymousAccess := true
- AccessPoint.Capabilities.AccessTaken := true
- AccessPoint.Capabilities.ExternalAuthorization := true
- AccessPoint.Capabilities.SupportedRecognitionTypes skipped
- AccessPoint.Capabilities.IdentifierAccess := true
- AccessPoint.Capabilities.SupportedSecurityLevels skipped
- AccessPoint.AuthenticationProfileToken skipped

10. The DUT responds with empty **ModifyAccessPointResponse** message.

11. ONVIF Client retrieves and checks **tns1:Configuration/AccessPoint/Changed** event for the specified Access Point token by following the procedure mentioned in Annex A.17 with the following input and output parameters

    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *accessPointToken* - access point token

12. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.4 with the following input and output parameters

    - in *s* - Subscription reference

13. ONVIF Client retrieves an access point by following the procedure mentioned in Annex A.24 with the following input and output parameters

- in *accessPointToken* - access point token
- out *accessPointList* - access point list

14. If *accessPointList*[0] item does not have equal field values to values from step 9, FAIL the test and go step 21.
15. ONVIF Client retrieves an access point info by following the procedure mentioned in Annex A.25 with the following input and output parameters
    - in *accessPointToken* - access point token
    - out *accessPointInfoList* - access point info list
16. If *accessPointInfoList*[0] item does not have equal field values to values from step 9, FAIL the test and go step 21.
17. ONVIF Client retrieves a complete access point information list by following the procedure mentioned in Annex A.1 with the following input and output parameters
    - out *accessPointInfoCompleteList* - access point info list
18. If *accessPointInfoCompleteList* does not have AccessPointInfo[token:= *accessPointToken*] item with equal field values to values from step 9, FAIL the test and go step 21.
19. ONVIF Client retrieves a complete list of access points by following the procedure mentioned in Annex A.12 with the following input and output parameters
    - out *accessPointCompleteList2* - access point list
20. If *accessPointCompleteList2* does not have AccessPoint[token = *accessPointToken*] item with equal field values to values from step 9, FAIL the test and go step 21.
21. ONVIF Client deletes the access point by following the procedure mentioned in Annex A.26 to restore DUT configuration with the following input and output parameters
    - in *accessPointToken* - access point token
22. ONVIF Client restores access point deleted at step 4 if any.
23. ONVIF Client deletes door created at step 6.2.1 if any.
24. ONVIF Client deletes areas created at step 5 if any.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

**Note:** The following fields are compared at steps 14, 20:

- AccessPoint:
  - token
  - Name
  - Description
  - AreaFrom
  - AreaTo
  - EntityType
  - Entity
  - Capabilities
    - DisableAccessPoint
    - Duress
    - AnonymousAccess
    - AccessTaken
    - ExternalAuthorization
    - SupportedRecognitionTypes
    - IdentifierAccess
    - SupportedSecurityLevels
  - AuthenticationProfileToken

**Note:** The following fields are compared at steps 16, 18:

- AccessPoint:
  - token
  - Name

- Description
- AreaFrom
- AreaTo
- EntityType
- Entity
- Capabilities
  - DisableAccessPoint
  - Duress
  - AnonymousAccess
  - AccessTaken
  - ExternalAuthorization
  - SupportedRecognitionTypes
  - IdentifierAccess
  - SupportedSecurityLevels

### 4.14.8 DELETE ACCESS POINT

**Test Case ID:** ACCESSCONTROL-14-1-8

**Specification Coverage:** DeleteAccessPoint command (ONVIF Access Control Service Specification)

**Feature Under Test:** DeleteAccessPoint

**WSDL Reference:** accesscontrol.wsdl, door.wsdl and event.wsdl

**Test Purpose:** To verify deleting of access point and generating of access point removed notifications.

**Pre-Requisite:** Access Control Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional access point. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of access points by following the procedure mentioned in Annex A.12 with the following output parameters

   - out *accessPointCompleteList1* - complete access point list

4. ONVIF Client checks free storage for additional access point by following the procedure mentioned in Annex A.15 with the following input and output parameters

   - in *accessPointCompleteList1* - complete access point list

   - out (optional) *accessPointToRestore* - deleted access point

5. If Door Control Service is supported by the DUT

   5.1. ONVIF Client gets the Door Control service capabilities (out *cap*) by following the procedure mentioned in Annex A.22.

   5.2. If *cap*.MaxDoors >0 or skipped:

      5.2.1. ONVIF Client retrieves references to a Door existing on the DUT by following the procedure mentioned in Annex A.20 with the following input and output parameters

      - out (optional) *doorToken* - token of the door

6. ONVIF Client creates access point by following the procedure mentioned in Annex A.27 with the following input and output parameters

   - in "Test Name1" - Access Point Name

   - in "Test Description1" - Access Point Description

   - in "tdc:Door" if door token was returned at step 6.2.1, otherwise value taken from UI - Access Point Entity Type

   - in *doorToken* if it was returned at step 6.2.1, otherwise value taken from UI - Access Point Entity Token

   - in false - Disable Access Point Capabilities

- out *accessPointToken* - Access Point Token

7. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.3 with the following input and output parameters

    - in **tns1:Configuration/AccessPoint/Removed** - Notification Topic
    - out *s* - Subscription reference
    - out *currentTime* - current time for the DUT
    - out *terminationTime* - Subscription termination time

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

    - Token := *accessPointToken*

9. The DUT responds with empty **DeleteAccessPointResponse** message.

10. ONVIF Client retrieves and checks **tns1:Configuration/AccessPoint/Removed** event for the specified access point token by following the procedure mentioned in Annex A.28 with the following input and output parameters

    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *accessPointToken* - access point token

11. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.4 with the following input and output parameters

    - in *s* - Subscription reference

12. ONVIF Client retrieves an access point by following the procedure mentioned in Annex A.24 with the following input and output parameters

    - in *accessPointToken* - access point token
    - out *accessPointList* - access point list

13. If *accessPointList* is not empty, FAIL the test and go step 20.

14. ONVIF Client retrieves an access point info by following the procedure mentioned in Annex A.25 with the following input and output parameters

- in *accessPointToken* - access point token
- out *accessPointInfoList* - access point info list

15. If *accessPointInfoList* is not empty, FAIL the test and go step 20.
16. ONVIF Client retrieves a complete access point information list by following the procedure mentioned in Annex A.1 with the following input and output parameters
    - out *accessPointInfoCompleteList* - access point info list
17. If *accessPointInfoCompleteList* contains AccessPointInfo[token:= *accessPointToken*] item, FAIL the test and go step 20.
18. ONVIF Client retrieves a complete list of access points by following the procedure mentioned in Annex A.12 with the following input and output parameters
    - out *accessPointCompleteList2* - access point list
19. If *accessPointCompleteList2* contains AccessPoint[token:= *accessPointToken*] item, FAIL the test and go step 20.
20. ONVIF Client restores access point deleted at step 4 if any.
21. ONVIF Client deletes door created at step 6.2.1 if any.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

### 4.14.9 GET ACCESS POINTS WITH INVALID TOKEN

**Test Case ID:** ACCESSCONTROL-14-1-9

**Specification Coverage:**GetAccessPoints command (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAccessPoints

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get Access Points with invalid token.

**Pre-Requisite:** Access Control Service is received from the DUT. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability.

**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 points info by following the procedure mentioned in Annex A.1 with the following output parameters
   - out *accessPointInfoCompleteList* - complete access point info list
4. Set the following:
   - *invalidToken* := value not equal to any *accessPointInfoCompleteList*.token
5. ONVIF client invokes **GetAccessPoints** with parameters
   - Token[0] := *invalidToken*
6. The DUT responds with **GetAccessPointsResponse** message with parameters
   - AccessPoint list =: *accessPointList*
7. If *accessPointList* is not empty, FAIL the test.
8. If *accessPointInfoCompleteList* is empty, skip other steps.
9. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.13 with the following output parameters
   - out *cap* - access control capabilities
10. If *cap*.MaxLimit is less than 2, skip other steps.
11. ONVIF client invokes **GetAccessPoints** with parameters
    - Token[0] := *invalidToken*
    - Token[1] := *accessPointInfoCompleteList*[0].token
12. The DUT responds with **GetAccessPointsResponse** message with parameters

- AccessPointInfo list =: *accessPointList*

13. If *accessPointList* is empty, FAIL the test.

14. If *accessPointList* contains more than one item, FAIL the test.

15. If *accessPointList*[0].token does not equal to *accessPointInfoCompleteList*[0].token, FAIL the test.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 4.14.10 GET ACCESS POINTS - TOO MANY ITEMS

**Test Case ID:** ACCESSCONTROL-14-1-10

**Specification Coverage:** GetAccessPoints command (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAccessPoints

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get Access Points if there are more items than MaxLimit in request.

**Pre-Requisite:** Access Control Service is received from the DUT. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability.

**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 points by following the procedure mentioned in Annex A.12 with the following output parameters

- out *accessPointCompleteList* - complete access point list

4. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.13 with the following output parameters

   - out *cap* - access control capabilities

5. If *accessPointCompleteList*.token items number is less than *cap*.MaxLimit or equal to *cap*.MaxLimit, skip other steps.

6. Set the following:

   - *tokenList* := [subset of *accessPointCompleteList*.token values with items number equal to *cap*.MaxLimit + 1]

7. ONVIF client invokes **GetAccessPoints** 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

## 4.14.11 CREATE ACCESS POINT - NOT EMPTY ACCESS POINT TOKEN

**Test Case ID:** ACCESSCONTROL-14-1-11

**Specification Coverage:** CreateAccessPoint command (ONVIF Access Control Service Specification), AccessPoint (ONVIF Access Control Service Specification)

**Feature Under Test:** CreateAccessPoint

**WSDL Reference:** accesscontrol.wsdl, door.wsdl

**Test Purpose:** Create Access Point with not Empty Token Verification.

**Pre-Requisite:** Access Control Service is received from the DUT. The DUT shall have enough free storage capacity for one additional access point. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability.

**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 points by following the procedure mentioned in Annex A.12 with the following output parameters

   - out *accessPointCompleteList1* - complete access point list

4. ONVIF Client checks free storage for additional access point by following the procedure mentioned in Annex A.15 with the following input and output parameters

   - in *accessPointCompleteList1* - complete access point list

   - out (optional) *accessPointToRestore* - deleted access point

5. If Door Control Service is supported by the DUT

   5.1. ONVIF Client gets the Door Control service capabilities (out *cap*) by following the procedure mentioned in Annex A.22.

   5.2. If *cap*.MaxDoors >0 or skipped:

      5.2.1. ONVIF Client retrieves references to a Door existing on the DUT by following the procedure mentioned in Annex A.20 with the following input and output parameters

      - out (optional) *doorToken* - token of the door

6. ONVIF client invokes **CreateAccessPoint** with parameters

   - AccessPoint.token := "AccessPointToken"

   - AccessPoint.Name := "Test Name"

   - AccessPoint.Description skipped

   - AccessPoint.AreaFrom skipped

   - AccessPoint.AreaTo skipped

- AccessPoint.EntityType := "tdc:Door" if door token was returned at step 6.2.1, otherwise value taken from UI.
- AccessPoint.Entity := *doorToken* if it was returned at step 6.2.1, otherwise value taken from UI.
- AccessPoint.Capabilities.DisableAccessPoint := false
- AccessPoint.Capabilities.Duress skipped
- AccessPoint.Capabilities.AnonymousAccess skipped
- AccessPoint.Capabilities.AccessTaken skipped
- AccessPoint.Capabilities.ExternalAuthorization skipped
- AccessPoint.Capabilities.SupportedRecognitionTypes skipped
- AccessPoint.Capabilities.IdentifierAccess skipped
- AccessPoint.Capabilities.SupportedSecurityLevels skipped
- AccessPoint.AuthenticationProfileToken skipped

7. The DUT responds with **env:Sender\ter:InvalidArgVal** SOAP 1.2 fault.
8. ONVIF Client restores access point deleted at step 4 if any.
9. ONVIF Client deletes door created at step 6.2.1 if any.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

### 4.14.12 MODIFY ACCESS POINT WITH INVALID TOKEN

**Test Case ID:** ACCESSCONTROL-14-1-12

**Specification Coverage:** AccessPointInfo (ONVIF Access Control Service Specification), AccessPoint (ONVIF Access Control Service Specification), ModifyAccessPoint command (ONVIF Access Control Service Specification)

**Feature Under Test:** ModifyAccessPoint

**WSDL Reference:** accesscontrol.wsdl, door.wsdl

**Test Purpose:** To verify modifying of access point with invalid token.

**Pre-Requisite:** Access Control Service is received from the DUT. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability.

**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 point info by following the procedure mentioned in Annex A.1 with the following output parameters
   - out *accessPointInfoCompleteList* - complete access point info list
4. Set the following:
   - *invalidToken* := value not equal to any *accessPointInfoCompleteList*.token
5. If Door Control Service is supported by the DUT

   5.1. ONVIF Client gets the Door Control service capabilities (out *cap*) by following the procedure mentioned in Annex A.22.

   5.2. If *cap*.MaxDoors >0 or skipped:

   5.2.1. ONVIF Client retrieves references to a Door existing on the DUT by following the procedure mentioned in Annex A.20 with the following input and output parameters

   - out (optional) *doorToken* - token of the door
6. ONVIF client invokes **ModifyAccessPoint** with parameters
   - AccessPoint.token := *invalidToken*
   - AccessPoint.Name := "Test Name"
   - AccessPoint.Description skipped
   - AccessPoint.AreaFrom skipped

- AccessPoint.AreaTo skipped
- AccessPoint.EntityType := "tdc:Door" if door token was returned at step 6.2.1, otherwise value taken from UI
- AccessPoint.Entity := *doorToken* if it was returned at step 6.2.1, otherwise value taken from UI
- AccessPoint.Capabilities.DisableAccessPoint := false
- AccessPoint.Capabilities.Duress skipped
- AccessPoint.Capabilities.AnonymousAccess skipped
- AccessPoint.Capabilities.AccessTaken skipped
- AccessPoint.Capabilities.ExternalAuthorization skipped
- AccessPoint.Capabilities.SupportedRecognitionTypes skipped
- AccessPoint.Capabilities.IdentifierAccess := true
- AccessPoint.Capabilities.SupportedSecurityLevels skipped
- AccessPoint.AuthenticationProfileToken skipped

7. The DUT returns **env:Sender\ter:InvalidArgVal\ter:NotFound** SOAP 1.2 fault.
8. ONVIF Client deletes door created at step 6.2.1 if any.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **env:Sender\ter:InvalidArgVal\ter:NotFound** SOAP 1.2 fault

### 4.14.13 DELETE ACCESS POINT WITH INVALID TOKEN

**Test Case ID:** ACCESSCONTROL-14-1-13

**Specification Coverage:** DeleteAccessPoint command (ONVIF Access Control Service Specification)

**Feature Under Test:** DeleteAccessPoint

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify deleting of access point with invalid token.

**Pre-Requisite:** Access Control Service is received from the DUT. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability.

**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 point info by following the procedure mentioned in Annex A.1 with the following output parameters
   - out *accessPointInfoCompleteList* - complete access point info list
4. Set the following:
   - *invalidToken* := value not equal to any *accessPointInfoCompleteList*.token
5. ONVIF Client invokes **DeleteAccessPoint** with parameters
   - Token := *invalidToken*
6. The DUT returns **env:Sender\ter:InvalidArgVal\ter:NotFound** SOAP 1.2 fault.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **env:Sender\ter:InvalidArgVal\ter:NotFound** SOAP 1.2 fault

## 4.14.14 CREATE NEW ACCESS POINT WITH SET ACCESS POINT (ACCESS POINT CAPABILITIES TRUE)

**Test Case ID:** ACCESSCONTROL-14-1-14

**Specification Coverage:** SetAccessPoint command (ONVIF Access Control Service Specification), AccessPoint (ONVIF Access Control Service Specification)

**Feature Under Test:** SetAccessPoint

**WSDL Reference:** accesscontrol.wsdl, door.wsdl, event.wsdl

**Test Purpose:** To verify creation of access point using SetAccessPoint command and generating of appropriate notifications.

**Pre-Requisite:** Access Control Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional access point. Client Supplied Token is supported by the DUT as indicated by ClientSuppliedTokenSupported capability.

**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 points by following the procedure mentioned in Annex A.12 with the following output parameters
   - out *accessPointCompleteList1* - complete access point list
4. Set *acccsPointToken* := token that differs from tokens listed in *accessPointCompleteList1*.
5. ONVIF Client checks free storage for additional access point by following the procedure mentioned in Annex A.15 with the following input and output parameters
   - in *accessPointCompleteList1* - complete access point list
   - out (optional) *accessPointToRestore* - deleted access point
6. ONVIF Client retrieves references to Areas existing on the DUT by following the procedure mentioned in Annex A.18 with the following input and output parameters
   - out (optional) *areaToken1* - token of the 1st Area
   - out (optional) *areaToken2* - token of the 2nd Area
7. If Door Control Service is supported by the DUT

   7.1. ONVIF Client gets the Door Control service capabilities (out *cap*) by following the procedure mentioned in Annex A.22.

   7.2. If *cap*.MaxDoors >0 or skipped:

   7.2.1. ONVIF Client retrieves references to a Door existing on the DUT by following the procedure mentioned in Annex A.20 with the following input and output parameters

  - out (optional) *doorToken* - token of the door

8. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.3 with the following input and output parameters

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

9. ONVIF client invokes **SetAccessPoint** with parameters

   - AccessPoint.token := *accssPointToken*
   - AccessPoint.Name := "Test Name"
   - AccessPoint.Description := "Test Description"
   - AccessPoint.AreaFrom := *areaToken1* if it was returned at step 6, otherwise skipped
   - AccessPoint.AreaTo := *areaToken2* if it was returned at step 6, otherwise skipped
   - AccessPoint.EntityType := "tdc:Door" if door token was returned at step 6.2.1, otherwise value taken from UI.
   - AccessPoint.Entity := *doorToken* if it was returned at step 6.2.1, otherwise value taken from UI.
   - AccessPoint.Capabilities.DisableAccessPoint := true
   - AccessPoint.Capabilities.Duress := true
   - AccessPoint.Capabilities.AnonymousAccess := true
   - AccessPoint.Capabilities.AccessTaken := true
   - AccessPoint.Capabilities.ExternalAuthorization := true
   - AccessPoint.Capabilities.SupportedRecognitionTypes skipped
   - AccessPoint.Capabilities.IdentifierAccess := true
   - AccessPoint.Capabilities.SupportedSecurityLevels skipped
   - AccessPoint.AuthenticationProfileToken skipped

10. The DUT responds with **SetAccessPointResponse** message

11. ONVIF Client retrieves and checks **tns1:Configuration/AccessPoint/Changed** event for the specified AccessPoint token by following the procedure mentioned in Annex A.17 with the following input and output parameters

    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *accessPointToken* - Access Point token

12. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.4 with the following input and output parameters

    - in *s* - Subscription reference

13. ONVIF Client retrieves an access point by following the procedure mentioned in Annex A.24 with the following input and output parameters

    - in *accessPointToken* - access point token
    - out *accessPointList* - access point list

14. If *accessPointList*[0] item does not have equal field values to values from step 9, FAIL the test and go step 21.

15. ONVIF Client retrieves an access point info by following the procedure mentioned in Annex A.25 with the following input and output parameters

    - in *accessPointToken* - access point token
    - out *accessPointInfoList* - access point info list

16. If *accessPointInfoList*[0] item does not have equal field values to values from step 9, FAIL the test and go step 21.

17. ONVIF Client retrieves a complete access point information list by following the procedure mentioned in Annex A.1 with the following input and output parameters

    - out *accessPointInfoCompleteList* - access point info list

18. If *accessPointInfoCompleteList* does not have AccessPointInfo[token = *accessPointToken*] item with equal field values to values from step 9, FAIL the test and go step 21.

19. ONVIF Client retrieves a complete list of access points by following the procedure mentioned in Annex A.12 with the following input and output parameters

    • out *accessPointCompleteList2* - access point list

20. If *accessPointCompleteList2* does not have AccessPoint[token = *accessPointToken*] item with equal field values to values from step 9, FAIL the test and go step 21.

21. ONVIF Client deletes the access point by following the procedure mentioned in Annex A.26 to restore DUT configuration with the following input and output parameters

    • in *accessPointToken* - access point token

22. ONVIF Client restores access point deleted at step 5 if any.

23. ONVIF Client deletes door created at step 6.2.1 if any.

24. ONVIF Client deletes areas created at step 6 if any.

**Test Result:**

**PASS –**

• DUT passes all assertions.

**FAIL –**

• The DUT did not send **CreateAccessPointResponse** message.

**Note:** The following fields are compared at steps 14, 20:

• AccessPoint:

  • token

  • Name

  • Description

  • AreaFrom

  • AreaTo

  • EntityType

  • Entity

  • Capabilities

    - DisableAccessPoint
    - Duress
    - AnonymousAccess
    - AccessTaken
    - ExternalAuthorization
    - SupportedRecognitionTypes
    - IdentifierAccess
    - SupportedSecurityLevels
  - AuthenticationProfileToken

**Note:** The following fields are compared at steps 16, 18:

- AccessPoint:
  - token
  - Name
  - Description
  - AreaFrom
  - AreaTo
  - EntityType
  - Entity
  - Capabilities
    - DisableAccessPoint
    - Duress
    - AnonymousAccess
    - AccessTaken
    - ExternalAuthorization

- SupportedRecognitionTypes
- IdentifierAccess
- SupportedSecurityLevels

## 4.14.15 CREATE NEW ACCESS POINT WITH SET ACCESS POINT (ACCESS POINT CAPABILITIES FALSE)

**Test Case ID:** ACCESSCONTROL-14-1-15

**Specification Coverage:** SetAccessPoint command (ONVIF Access Control Service Specification), AccessPoint (ONVIF Access Control Service Specification)

**Feature Under Test:** SetAccessPoint

**WSDL Reference:** accesscontrol.wsdl, door.wsdl, event.wsdl

**Test Purpose:** To verify creation of access point using SetAccessPoint command and generating of appropriate notifications.

**Pre-Requisite:** Access Control Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional access point. Client Supplied Token is supported by the DUT as indicated by ClientSuppliedTokenSupported capability.

**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 points by following the procedure mentioned in Annex A.12 with the following output parameters
   - out *accessPointCompleteList1* - complete access point list
4. Set *accssPointToken* := token that differs from tokens listed in *accessPointCompleteList1*.
5. ONVIF Client checks free storage for additional access point by following the procedure mentioned in Annex A.15 with the following input and output parameters
   - in *accessPointCompleteList1* - complete access point list

- out (optional) *accessPointToRestore* - deleted access point

6. If Door Control Service is supported by the DUT

    6.1. ONVIF Client gets the Door Control service capabilities (out *cap*) by following the procedure mentioned in Annex A.22.

    6.2. If *cap*.MaxDoors >0 or skipped:

        6.2.1. ONVIF Client retrieves references to a Door existing on the DUT by following the procedure mentioned in Annex A.20 with the following input and output parameters

        - out (optional) *doorToken* - token of the door

7. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.3 with the following input and output parameters

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

8. ONVIF client invokes **SetAccessPoint** with parameters

    - AccessPoint.token := *accssPointToken*
    - AccessPoint.Name := "Test Name"
    - AccessPoint.Description skipped
    - AccessPoint.AreaFrom skipped
    - AccessPoint.AreaTo skipped
    - AccessPoint.EntityType := "tdc:Door" if door token was returned at step 6.2.1, otherwise value taken from UI.
    - AccessPoint.Entity := *doorToken* if it was returned at step 6.2.1, otherwise value taken from UI.
    - AccessPoint.Capabilities.DisableAccessPoint := false
    - AccessPoint.Capabilities.Duress := false

- AccessPoint.Capabilities.AnonymousAccess := false
- AccessPoint.Capabilities.AccessTaken := false
- AccessPoint.Capabilities.ExternalAuthorization := false
- AccessPoint.Capabilities.SupportedRecognitionTypes skipped
- AccessPoint.Capabilities.IdentifierAccess := false
- AccessPoint.Capabilities.SupportedSecurityLevels skipped
- AccessPoint.AuthenticationProfileToken skipped

9. The DUT responds with **SetAccessPointResponse** message

10. ONVIF Client retrieves and checks **tns1:Configuration/AccessPoint/Changed** event for the specified AccessPoint token by following the procedure mentioned in Annex A.17 with the following input and output parameters

    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *accessPointToken* - Access Point token

11. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.4 with the following input and output parameters

    - in *s* - Subscription reference

12. ONVIF Client retrieves an access point by following the procedure mentioned in Annex A.24 with the following input and output parameters

    - in *accessPointToken* - access point token
    - out *accessPointList* - access point list

13. If *accessPointList*[0] item does not have equal field values to values from step 8, FAIL the test and go step 20.

14. ONVIF Client retrieves an access point info by following the procedure mentioned in Annex A.25 with the following input and output parameters

    - in *accessPointToken* - access point token

- out *accessPointInfoList* - access point info list

15. If *accessPointInfoList*[0] item does not have equal field values to values from step 8, FAIL the test and go step 20.

16. ONVIF Client retrieves a complete access point information list by following the procedure mentioned in Annex A.1 with the following input and output parameters

    - out *accessPointInfoCompleteList* - access point info list

17. If *accessPointInfoCompleteList* does not have AccessPointInfo[token = *accessPointToken*] item with equal field values to values from step 8, FAIL the test and go step 20.

18. ONVIF Client retrieves a complete list of access points by following the procedure mentioned in Annex A.12 with the following input and output parameters

    - out *accessPointCompleteList2* - access point list

19. If *accessPointCompleteList2* does not have AccessPoint[token = *accessPointToken*] item with equal field values to values from step 8, FAIL the test and go step 20.

20. ONVIF Client deletes the access point by following the procedure mentioned in Annex A.26 to restore DUT configuration with the following input and output parameters

    - in *accessPointToken* - access point token

21. ONVIF Client restores access point deleted at step 5 if any.

22. ONVIF Client deletes door created at step 6.2.1 if any.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

**Note:** The following fields are compared at steps 13, 19:

- AccessPoint:

  - token

  - Name

  - Description
  - AreaFrom
  - AreaTo
  - EntityType
  - Entity
  - Capabilities
    - DisableAccessPoint
    - Duress
    - AnonymousAccess
    - AccessTaken
    - ExternalAuthorization
    - SupportedRecognitionTypes
    - IdentifierAccess
    - SupportedSecurityLevels
  - AuthenticationProfileToken

**Note:** The following fields are compared at steps 15, 17:

- AccessPoint:
  - token
  - Name
  - Description
  - AreaFrom
  - AreaTo
  - EntityType
  - Entity

- Capabilities
  - DisableAccessPoint
  - Duress
  - AnonymousAccess
  - AccessTaken
  - ExternalAuthorization
  - SupportedRecognitionTypes
  - IdentifierAccess
  - SupportedSecurityLevels

### 4.14.16 MODIFY ACCESS POINT WITH SET ACCESS POINT

**Test Case ID:** ACCESSCONTROL-14-1-16

**Specification Coverage:** SetAccessPoint command (ONVIF Access Control Service Specification), AccessPointInfo (ONVIF Access Control Service Specification), AccessPoint (ONVIF Access Control Service Specification), ModifyAccessPoint command (ONVIF Access Control Service Specification)

**Feature Under Test:** SetAccessPoint

**WSDL Reference:** accesscontrol.wsdl, door.wsdl, event.wsdl

**Test Purpose:** To verify modifying of access point using SetAccessPoint and generating of access point changed notifications.

**Pre-Requisite:** Access Control Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional access point. Client Supplied Token is supported by the DUT as indicated by ClientSuppliedTokenSupported capability.

**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 points by following the procedure mentioned in Annex A.12 with the following output parameters

   - out *accessPointCompleteList1* - complete access point list

4. ONVIF Client checks free storage for additional access point by following the procedure mentioned in Annex A.15 with the following input and output parameters

   - in *accessPointCompleteList1* - complete access point list
   - out (optional) *accessPointToRestore* - deleted access point

5. ONVIF Client retrieves references to Areas existing on the DUT by following the procedure mentioned in Annex A.18 with the following input and output parameters

   - out (optional) *areaToken1* - token of the 1st Area
   - out (optional) *areaToken2* - token of the 2nd Area

6. If Door Control Service is supported by the DUT

   6.1. ONVIF Client gets the Door Control service capabilities (out *cap*) by following the procedure mentioned in Annex A.22.

   6.2. If *cap*.MaxDoors >0 or skipped:

   6.2.1. ONVIF Client retrieves references to a Door existing on the DUT by following the procedure mentioned in Annex A.20 with the following input and output parameters

   - out (optional) *doorToken* - token of the door

7. ONVIF Client creates access point by following the procedure mentioned in Annex A.27 with the following input and output parameters

   - in "Test Name1" - Access Point Name
   - in "Test Description1" - Access Point Description
   - in *areaToken1* if it was returned at step 5, otherwise skipped - Area From Token
   - in *areaToken2* if it was returned at step 5, otherwise skipped - Area To Token
   - in "tdc:Door" if door token was returned at step 6.2.1, otherwise value taken from UI - Access Point Entity Type
   - in *doorToken* if it was returned at step 6.2.1, otherwise value taken from UI - Access Point Entity Token

- in false - Disable Access Point Capabilities
- in false - Access Point Duress Capabilities
- in false - Access Point Anonymous Access Capabilities
- in false - Access Point Access Taken Capabilities
- in false - Access Point External Authorization Capabilities
- in false - Access Point Identifier Access Capabilities
- out *accessPointToken* - Access Point Token

8. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.3 with the following input and output parameters

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

9. ONVIF client invokes **SetAccessPoint** with parameters

   - AccessPoint.token := *accessPointToken*
   - AccessPoint.Name := "Test Name2"
   - AccessPoint.Description := "Test Description2"
   - AccessPoint.AreaFrom := *areaToken2* if it was returned at step 5, otherwise skipped
   - AccessPoint.AreaTo := *areaToken1* if it was returned at step 5, otherwise skipped
   - AccessPoint.EntityType := "tdc:Door" if door token was returned at step 6.2.1, otherwise value taken from UI
   - AccessPoint.Entity := *doorToken* if it was returned at step 6.2.1, otherwise value taken from UI
   - AccessPoint.Capabilities.DisableAccessPoint := true
   - AccessPoint.Capabilities.Duress := true
   - AccessPoint.Capabilities.AnonymousAccess := true

- AccessPoint.Capabilities.AccessTaken := true
- AccessPoint.Capabilities.ExternalAuthorization := true
- AccessPoint.Capabilities.SupportedRecognitionTypes skipped
- AccessPoint.Capabilities.IdentifierAccess := true
- AccessPoint.Capabilities.SupportedSecurityLevels skipped
- AccessPoint.AuthenticationProfileToken skipped

10. The DUT responds with empty **SetAccessPointResponse** message.

11. ONVIF Client retrieves and checks **tns1:Configuration/AccessPoint/Changed** event for the specified Access Point token by following the procedure mentioned in Annex A.17 with the following input and output parameters

    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *accessPointToken* - access point token

12. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.4 with the following input and output parameters

    - in *s* - Subscription reference

13. ONVIF Client retrieves an access point by following the procedure mentioned in Annex A.24 with the following input and output parameters

    - in *accessPointToken* - access point token
    - out *accessPointList* - access point list

14. If *accessPointList*[0] item does not have equal field values to values from step 9, FAIL the test and go step 21.

15. ONVIF Client retrieves an access point info by following the procedure mentioned in Annex A.25 with the following input and output parameters

    - in *accessPointToken* - access point token
    - out *accessPointInfoList* - access point info list

16. If *accessPointInfoList*[0] item does not have equal field values to values from step 9, FAIL the test and go step 21.

17. ONVIF Client retrieves a complete access point information list by following the procedure mentioned in Annex A.1 with the following input and output parameters

    - out *accessPointInfoCompleteList* - access point info list

18. If *accessPointInfoCompleteList* does not have AccessPointInfo[token:= *accessPointToken*] item with equal field values to values from step 9, FAIL the test and go step 21.

19. ONVIF Client retrieves a complete list of access points by following the procedure mentioned in Annex A.12 with the following input and output parameters

    - out *accessPointCompleteList2* - access point list

20. If *accessPointCompleteList2* does not have AccessPoint[token = *accessPointToken*] item with equal field values to values from step 9, FAIL the test and go step 21.

21. ONVIF Client deletes the access point by following the procedure mentioned in Annex A.26 to restore DUT configuration with the following input and output parameters

    - in *accessPointToken* - access point token

22. ONVIF Client restores access point deleted at step 4 if any.

23. ONVIF Client deletes door created at step 6.2.1 if any.

24. ONVIF Client deletes areas created at step 5 if any.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

**Note:** The following fields are compared at steps 14, 20:

- AccessPoint:
  - token
  - Name
  - Description

- AreaFrom
- AreaTo
- EntityType
- Entity
- Capabilities
  - DisableAccessPoint
  - Duress
  - AnonymousAccess
  - AccessTaken
  - ExternalAuthorization
  - SupportedRecognitionTypes
  - IdentifierAccess
  - SupportedSecurityLevels
- AuthenticationProfileToken

**Note:** The following fields are compared at steps 16, 18:

- AccessPoint:
  - token
  - Name
  - Description
  - AreaFrom
  - AreaTo
  - EntityType
  - Entity
  - Capabilities

- DisableAccessPoint
- Duress
- AnonymousAccess
- AccessTaken
- ExternalAuthorization
- SupportedRecognitionTypes
- IdentifierAccess
- SupportedSecurityLevels

## 4.14.17 SET ACCESS POINT - EMPTY ACCESS POINT TOKEN

**Test Case ID:** ACCESSCONTROL-14-1-17

**Specification Coverage:** SetAccessPoint command (ONVIF Access Control Service Specification)

**Feature Under Test:** SetAccessPoint

**WSDL Reference:** accesscontrol.wsdl, door.wsdl

**Test Purpose:** To verify SetAccessPoint command with empty access point token.

**Pre-Requisite:** Access Control Service is received from the DUT. The DUT shall have enough free storage capacity for one additional access point. Client Supplied Token is supported by the DUT as indicated by ClientSuppliedTokenSupported capability.

**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 points by following the procedure mentioned in Annex A.12 with the following output parameters
   - out *accessPointCompleteList1* - complete access point list

4. ONVIF Client checks free storage for additional access point by following the procedure mentioned in Annex A.15 with the following input and output parameters

   - in *accessPointCompleteList1* - complete access point list

   - out (optional) *accessPointToRestore* - deleted access point

5. If Door Control Service is supported by the DUT

   5.1. ONVIF Client gets the Door Control service capabilities (out *cap*) by following the procedure mentioned in Annex A.22.

   5.2. If *cap*.MaxDoors >0 or skipped:

   5.2.1.ONVIF Client retrieves references to a Door existing on the DUT by following the procedure mentioned in Annex A.20 with the following input and output parameters

   - out (optional) *doorToken* - token of the door

6. ONVIF client invokes **SetAccessPoint** with parameters

   - AccessPoint.token := ""

   - AccessPoint.Name := "Test Name"

   - AccessPoint.Description skipped

   - AccessPoint.AreaFrom skipped

   - AccessPoint.AreaTo skipped

   - AccessPoint.EntityType := "tdc:Door" if door token was returned at step 6.2.1, otherwise value taken from UI.

   - AccessPoint.Entity := *doorToken* if it was returned at step 6.2.1, otherwise value taken from UI.

   - AccessPoint.Capabilities.DisableAccessPoint := false

   - AccessPoint.Capabilities.Duress := false

   - AccessPoint.Capabilities.AnonymousAccess := false

   - AccessPoint.Capabilities.AccessTaken := false

   - AccessPoint.Capabilities.ExternalAuthorization := false

- AccessPoint.Capabilities.SupportedRecognitionTypes skipped
- AccessPoint.Capabilities.IdentifierAccess := false
- AccessPoint.Capabilities.SupportedSecurityLevels skipped
- AccessPoint.AuthenticationProfileToken skipped

7. The DUT responds with **env:Sender\ter:InvalidArgVal** SOAP 1.2 fault.
8. ONVIF Client restores access point deleted at step 4 if any.
9. ONVIF Client deletes door created at step 6.2.1 if any.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 4.15 Area Management

### 4.15.1 GET AREAS

**Test Case ID:** ACCESSCONTROL-15-1-1

**Specification Coverage:** AreaInfo (ONVIF Access Control Service Specification), Area (ONVIF Access Control Service Specification), GetAreas command (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAreas

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get Areas.

**Pre-Requisite:** Access Control Service is received from the DUT. Area Management is supported by the DUT as indicated by AreaManagementSupported capability.

**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 Areas (out *areaCompleteList*) by following the procedure mentioned in Annex A.14.

4. If *areaCompleteList* is empty, skip other steps.

5. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.13.

6. Set the following:

   - *tokenList* := [subset of *areaCompleteList*.token values with items number equal to *cap*.MaxLimit]

7. ONVIF client invokes **GetAreas** with parameters

   - Token list := *tokenList*

8. The DUT responds with **GetAreasResponse** message with parameters

   - Area list =: *areaList1*

9. If *areaList1* does not contain Area item for each token from *tokenList*, FAIL the test and skip other steps.

10. If *areaList1* contains at least two Area items with equal token, FAIL the test and skip other steps.

11. If *areaList1* contains other Area items than listed in *tokenList*, FAIL the test and skip other steps.

12. For each Area.token *token* from *areaCompleteList* repeat the following steps:

    12.1. ONVIF client invokes **GetAreas** with parameters

    - Token[0] := *token*

    12.2. The DUT responds with **GetAreasResponse** message with parameters

    - Area list =: *areaList2*

    12.3. If *areaList2* does not contain only one Area item with token equal to *token*, FAIL the test and skip other steps.

12.4. If *areaList2*[0] item does not have equal field values to *areaCompleteList*[token = *token*] item, FAIL the test and skip other steps.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

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

**Note:** The following fields are compared at step 12.4:

- Area:
  - token
  - Name
  - Description

### 4.15.2 GET AREA LIST - LIMIT

**Test Case ID:** ACCESSCONTROL-15-1-2

**Specification Coverage:** AreaInfo (ONVIF Access Control Service Specification), Area (ONVIF Access Control Service Specification), GetAreaList command (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAreaList

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get Area List using Limit.

**Pre-Requisite:** Access Control Service is received from the DUT. Area Management is supported by the DUT as indicated by AreaManagementSupported capability.

**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.13.
4. ONVIF client invokes **GetAreaList** with parameters
    - Limit := 1
    - StartReference skipped
5. The DUT responds with **GetAreaListResponse** message with parameters
    - NextStartReference =: *nextStartReference*
    - Area list =: *areaList1*
6. If *areaList1* contains more area 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 **GetAreaList** with parameters
    - Limit := *cap*.MaxLimit
    - StartReference skipped
9. The DUT responds with **GetAreaListResponse** message with parameters
    - NextStartReference =: *nextStartReference*
    - Area list =: *areaList2*
10. If *areaList2* contains more Area 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 **GetAreaList** with parameters
    - Limit := *limit*
    - StartReference skipped
14. The DUT responds with **GetAreaListResponse** message with parameters

- NextStartReference =: *nextStartReference*
- Area list =: *areaList3*

15. If *areaList3* contains more Area items than *limit*, FAIL the test and skip other steps.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

### 4.15.3 GET AREA LIST - START REFERENCE AND LIMIT

**Test Case ID:** ACCESSCONTROL-15-1-3

**Specification Coverage:** AreaInfo (ONVIF Access Control Service Specification), Area (ONVIF Access Control Service Specification), GetAreaList command (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAreaList

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get Area List using StartReference and Limit.

**Pre-Requisite:** Access Control Service is received from the DUT. Area Management is supported by the DUT as indicated by AreaManagementSupported capability.

**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.13.
4. ONVIF client invokes **GetAreaList** with parameters
    - Limit := *cap*.MaxLimit

- StartReference skipped

5. The DUT responds with **GetAreaListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - Area list =: *areaCompleteList1*
6. If *areaCompleteList1* contains more Area 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 **GetAreaList** with parameters
   - Limit := *cap*.MaxLimit
   - StartReference := *nextStartReference*

   7.2. The DUT responds with **GetAreaListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - Area list =: *areaListPart*

   7.3. If *areaListPart* contains more Area items than *cap*.MaxLimit, FAIL the test and skip other steps.

   7.4. Set the following:
   - *areaCompleteList1* := *areaCompleteList1* + *areaListPart*
8. If *areaCompleteList1* contains at least two Area items 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 areas info (out *areaInfoCompleteList*) by following the procedure mentioned in Annex A.2.

   9.2. If *areaCompleteList1* does not contain all areas from *areaInfoCompleteList*, FAIL the test and skip other steps.

   9.3. If *areaCompleteList1* contains areas other than areas from *areaInfoCompleteList*, FAIL the test and skip other steps.

   9.4. For each AreaInfo.token *token* from *areaInfoCompleteList* repeat the following steps:

9.4.1. If *areaCompleteList1*[token = *token*] item does not have equal field values to *areaInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.

9.5. Skip other steps.

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

- Limit := 1
- StartReference skipped

11. The DUT responds with **GetAreaListResponse** message with parameters

- NextStartReference =: *nextStartReference*
- Area list =: *areaCompleteList2*

12. If *areaCompleteList2* contains more Area 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 **GetAreaList** with parameters

- Limit := 1
- StartReference := *nextStartReference*

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

- NextStartReference =: *nextStartReference*
- Area list =: *areaListPart*

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

13.4. Set the following:

- *areaCompleteList2* := *areaCompleteList2* + *areaListPart*

14. If *areaCompleteList2* contains at least two Area items with equal token, FAIL the test and skip other steps.

15. If *areaCompleteList2* does not contain all areas from *areaCompleteList1*, FAIL the test and skip other steps.

16. If *areaCompleteList2* contains areas other than areas from *areaCompleteList1*, 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 areas info (out *areaInfoCompleteList*) by following the procedure mentioned in Annex A.2.

    17.2. If *areaCompleteList2* does not contain all areas from *areaInfoCompleteList*, FAIL the test and skip other steps.

    17.3. If *areaCompleteList2* contains areas other than areas from *areaInfoCompleteList*, FAIL the test and skip other steps.

    17.4. For each AreaInfo.token *token* from *areaInfoCompleteList* repeat the following steps:

        17.4.1. If *areaCompleteList2*[token = *token*] item does not have equal field values to *areaInfoCompleteList*[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 **GetAreaList** with parameters

    • Limit := *limit*

    • StartReference skipped

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

    • NextStartReference =: *nextStartReference*

    • Area list =: *areaCompleteList3*

21. If *areaCompleteList3* contains more Area 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 **GetAreaList** with parameters

        • Limit := *limit*

        • StartReference := *nextStartReference*

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

        • NextStartReference =: *nextStartReference*

        • Area list =: *areaListPart*

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

22.4. Set the following:

- *areaCompleteList3* := *areaCompleteList3* + *areaListPart*

23. If *areaCompleteList3* contains at least two Area items with equal token, FAIL the test and skip other steps.

24. If *areaCompleteList3* does not contain all areas from *areaCompleteList1*, FAIL the test and skip other steps.

25. If *areaCompleteList3* contains areas other than areas from *areaCompleteList1*, FAIL the test and skip other steps.

26. ONVIF Client retrieves a complete list of areas info (out *areaInfoCompleteList*) by following the procedure mentioned in Annex A.2.

27. If *areaCompleteList3* does not contain all areas from *areaInfoCompleteList*, FAIL the test and skip other steps.

28. If *areaCompleteList3* contains areas other than areas from *areaInfoCompleteList*, FAIL the test and skip other steps.

29. For each AreaInfo.token *token* from *areaInfoCompleteList* repeat the following steps:

29.1. If *areaCompleteList3*[token = *token*] item does not have equal field values to *areaInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

**Note:** The following fields are compared at step 9.4.1, 17.4.1, and 29.1:

- AreaInfo:
  - token
  - Name
  - Description

### 4.15.4 GET AREA LIST - NO LIMIT

**Test Case ID:** ACCESSCONTROL-15-1-4

**Specification Coverage:** AreaInfo (ONVIF Access Control Service Specification), Area (ONVIF Access Control Service Specification), GetAreaList command (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAreaList

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get Area List without using Limit.

**Pre-Requisite:** Access Control Service is received from the DUT. Area Management is supported by the DUT as indicated by AreaManagementSupported capability.

**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.13.
4. ONVIF client invokes **GetAreaList** with parameters
   - Limit skipped
   - StartReference skipped
5. The DUT responds with **GetAreaListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - Area list =: *areaCompleteList*
6. If *areaCompleteList* contains more Area 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 **GetAreaList** with parameters
   - Limit skipped

- StartReference := *nextStartReference*

7.2. The DUT responds with **GetAreaListResponse** message with parameters

- NextStartReference =: *nextStartReference*
- Area list =: *areaListPart*

7.3. If *areaListPart* contains more Area items than *cap*.MaxLimit, FAIL the test and skip other steps.

7.4. Set the following:

- *areaCompleteList* := *areaCompleteList* + *areaListPart*

8. If *areaCompleteList* contains at least two area items with equal token, FAIL the test.

9. ONVIF Client retrieves a complete list of areas (out *areaInfoCompleteList*) by following the procedure mentioned in Annex A.2.

10. If *areaCompleteList* does not contain all areas from *areaInfoCompleteList*, FAIL the test and skip other steps.

11. If *areaCompleteList* contains areas other than areas from *areaInfoCompleteList*, FAIL the test and skip other steps.

12. For each AreaInfo.token *token* from *areaInfoCompleteList* repeat the following steps:

12.1. If *areaCompleteList*[token = *token*] item does not have equal field values to *areaInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

**Note:** The following fields are compared at step 12.1:

- token
- Name
- Description

## 4.15.5 CREATE AREA

**Test Case ID:** ACCESSCONTROL-15-1-5

**Specification Coverage:** CreateArea command (ONVIF Access Control Service Specification), Area (ONVIF Access Control Service Specification)

**Feature Under Test:** CreateArea

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

**Test Purpose:** To verify creation of area and generating of area changed notifications.

**Pre-Requisite:** Access Control Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional area. Area Management is supported by the DUT as indicated by AreaManagementSupported capability. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves a complete list of areas by following the procedure mentioned in Annex A.14 with the following output parameters
   - out *areaCompleteList1* - complete area list
4. ONVIF Client checks free storage for additional area by following the procedure mentioned in Annex A.29 with the following input and output parameters
   - in *areaCompleteList1* - complete area list
   - out (optional) *areaToRestore* - deleted area
5. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.3 with the following input and output parameters
   - in **tns1:Configuration/Area/Changed** - Notification Topic
   - out *s* - Subscription reference
   - out *currentTime* - current time for the DUT
   - out *terminationTime* - Subscription termination time

6. ONVIF client invokes **CreateArea** with parameters
   - Area.token := ""
   - Area.Name := "Test Name"
   - Area.Description := "Test Description"
7. The DUT responds with **CreateAreaResponse** message with parameters
   - Token =: *areaToken*
8. ONVIF Client retrieves and checks **tns1:Configuration/Area/Changed** event for the specified Area token by following the procedure mentioned in Annex A.31 with the following input and output parameters
   - in *s* - Subscription reference
   - in *currentTime* - current time for the DUT
   - in *terminationTime* - subscription termination time
   - in *areaToken* - Area token
9. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - in *s* - Subscription reference
10. ONVIF Client retrieves an area by following the procedure mentioned in Annex A.32 with the following input and output parameters
    - in *areaToken* - area token
    - out *areaList* - area list
11. If *areaList*[0] item does not have equal field values to values from step 6, FAIL the test and go step 18.
12. ONVIF Client retrieves an area info by following the procedure mentioned in Annex A.33 with the following input and output parameters
    - in *areaToken* - area token
    - out *areaInfoList* - area info list
13. If *areaInfoList*[0] item does not have equal field values to values from step 6, FAIL the test and go step 18.

14. ONVIF Client retrieves a complete area information list by following the procedure mentioned in Annex A.2 with the following input and output parameters

    - out *areaInfoCompleteList* - area info list

15. If *areaInfoCompleteList* does not have AreaInfo[token = *areaToken*] item with equal field values to values from step 6, FAIL the test and go step 18.

16. ONVIF Client retrieves a complete list of areas by following the procedure mentioned in Annex A.14 with the following input and output parameters

    - out *areaCompleteList2* - area list

17. If *areaCompleteList2* does not have Area[token = *areaToken*] item with equal field values to values from step 6, FAIL the test and go step 18.

18. ONVIF Client deletes the area by following the procedure mentioned in Annex A.34 to restore DUT configuration with the following input and output parameters

    - in *areaToken* - area token

19. ONVIF Client restores area deleted at step 4 if any.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

**Note:** The following fields are compared at steps 11, 13, 15, and 17:

- Area:
    - token
    - Name
    - Description

### 4.15.6 MODIFY AREA

**Test Case ID:** ACCESSCONTROL-15-1-6

**Specification Coverage:** ModifyArea command (ONVIF Access Control Service Specification), Area (ONVIF Access Control Service Specification)

**Feature Under Test:** ModifyArea

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

**Test Purpose:** To verify modifying of area and generating of area changed notifications.

**Pre-Requisite:** Access Control Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional area. Area Management is supported by the DUT as indicated by AreaManagementSupported capability. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of areas by following the procedure mentioned in Annex A.14 with the following output parameters

   - out *areaCompleteList1* - complete area list

4. ONVIF Client checks free storage for additional area by following the procedure mentioned in Annex A.29 with the following input and output parameters

   - in *areaCompleteList1* - complete area list

   - out (optional) *areaToRestore* - deleted area

5. ONVIF Client creates an area by following the procedure mentioned in Annex A.19 with the following input and output parameters

   - in "Test Name1" - Access Point Name

   - in "Test Description1" - Access Point Description

   - out *areaToken* - area Token

6. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.3 with the following input and output parameters

   - in **tns1:Configuration/Area/Changed** - Notification Topic

- out *s* - Subscription reference
- out *currentTime* - current time for the DUT
- out *terminationTime* - Subscription termination time

7. ONVIF client invokes **ModifyArea** with parameters
   - Area.token := *areaToken*
   - Area.Name := "Test Name2"
   - Area.Description := "Test Description2"
8. The DUT responds with empty **ModifyAreaResponse** message.
9. ONVIF Client retrieves and checks **tns1:Configuration/Area/Changed** event for the specified Area token by following the procedure mentioned in Annex A.31 with the following input and output parameters
   - in *s* - Subscription reference
   - in *currentTime* - current time for the DUT
   - in *terminationTime* - subscription termination time
   - in *areaToken* - Area token
10. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.4 with the following input and output parameters
    - in *s* - Subscription reference
11. ONVIF Client retrieves an area by following the procedure mentioned in Annex A.32 with the following input and output parameters
    - in *areaToken* - area token
    - out *areaList* - area list
12. If *areaList*[0] item does not have equal field values to values from step 7, FAIL the test and go step 19.
13. ONVIF Client retrieves an area info by following the procedure mentioned in Annex A.33 with the following input and output parameters
    - in *areaToken* - area token

- out *areaInfoList* - area info list

14. If *areaInfoList*[0] item does not have equal field values to values from step 7, FAIL the test and go step 19.

15. ONVIF Client retrieves a complete area information list by following the procedure mentioned in Annex A.2 with the following input and output parameters

    - out *areaInfoCompleteList* - area info list

16. If *areaInfoCompleteList* does not have AreaInfo[token = *areaToken*] item with equal field values to values from step 7, FAIL the test and go step 19.

17. ONVIF Client retrieves a complete list of areas by following the procedure mentioned in Annex A.14 with the following input and output parameters

    - out *areaCompleteList2* - area list

18. If *areaCompleteList2* does not have Area[token = *areaToken*] item with equal field values to values from step 7, FAIL the test and go step 19.

19. ONVIF Client deletes the area by following the procedure mentioned in Annex A.34 to restore DUT configuration with the following input and output parameters

    - in *areaToken* - area token

20. ONVIF Client restores area deleted at step 4 if any.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

**Note:** The following fields are compared at steps 12, 14, 16, and 18:

- Area:
    - token
    - Name
    - Description

### 4.15.7 DELETE AREA

**Test Case ID:** ACCESSCONTROL-15-1-7

**Specification Coverage:** DeleteArea command (ONVIF Access Control Service Specification), Area (ONVIF Access Control Service Specification)

**Feature Under Test:** DeleteArea

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

**Test Purpose:** To verify modifying of area and generating of area removed notifications.

**Pre-Requisite:** Access Control Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional area. Area Management is supported by the DUT as indicated by AreaManagementSupported capability. Device supports Pull-Point Notification feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves a complete list of areas by following the procedure mentioned in Annex A.14 with the following output parameters
   - out *areaCompleteList1* - complete area list
4. ONVIF Client checks free storage for additional area by following the procedure mentioned in Annex A.29 with the following input and output parameters
   - in *areaCompleteList1* - complete area list
   - out (optional) *areaToRestore* - deleted area
5. ONVIF Client creates an area by following the procedure mentioned in Annex A.19 with the following input and output parameters
   - in "Test Name" - Access Point Name
   - out *areaToken* - area Token
6. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.3 with the following input and output parameters

- in **tns1:Configuration/Area/Removed** - Notification Topic
- out *s* - Subscription reference
- out *currentTime* - current time for the DUT
- out *terminationTime* - Subscription termination time

7. ONVIF client invokes **DeleteArea** with parameters
   - Area.token := *areaToken*
8. The DUT responds with empty **DeleteAreaResponse** message.
9. ONVIF Client retrieves and checks **tns1:Configuration/Area/Removed** event for the specified Area token by following the procedure mentioned in Annex A.35 with the following input and output parameters
   - in *s* - Subscription reference
   - in *currentTime* - current time for the DUT
   - in *terminationTime* - subscription termination time
   - in *areaToken* - Area token
10. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.4 with the following input and output parameters
    - in *s* - Subscription reference
11. ONVIF Client retrieves an area by following the procedure mentioned in Annex A.32 with the following input and output parameters
    - in *areaToken* - area token
    - out *areaList* - area list
12. If *areaList* list is not empty, FAIL the test and restore the DUT settings.
13. ONVIF Client retrieves an area info by following the procedure mentioned in Annex A.33 with the following input and output parameters
    - in *areaToken* - area token
    - out *areaInfoList* - area info list
14. If *areaInfoList* list is not empty, FAIL the test and restore the DUT settings.

15. ONVIF Client retrieves a complete area information list by following the procedure mentioned in Annex A.2 with the following input and output parameters
    - out *areaInfoCompleteList* - area info list
16. If *areaInfoCompleteList* contains AreaInfo[token = *areaToken*] item, FAIL the test and restore the DUT settings.
17. ONVIF Client retrieves a complete list of areas by following the procedure mentioned in Annex A.14 with the following input and output parameters
    - out *areaCompleteList2* - area list
18. If *areaCompleteList2* contains Area[token = *areaToken*] item, FAIL the test and restore the DUT settings.
19. ONVIF Client restores area deleted at step 4 if any.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

### 4.15.8 GET AREAS WITH INVALID TOKEN

**Test Case ID:** ACCESSCONTROL-15-1-8

**Specification Coverage:**GetAreas command (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAreas

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get Areas with invalid token.

**Pre-Requisite:** Access Control Service is received from the DUT. Area Management is supported by the DUT as indicated by AreaManagementSupported capability.

**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 areas info by following the procedure mentioned in Annex A.2 with the following output parameters
   - out *areaInfoCompleteList* - complete area info list
4. Set the following:
   - *invalidToken* := value not equal to any *areaInfoCompleteList*.token
5. ONVIF client invokes **GetAreas** with parameters
   - Token[0] := *invalidToken*
6. The DUT responds with **GetAreasResponse** message with parameters
   - Area list =: *areaList*
7. If *areaList* is not empty, FAIL the test.
8. If *areaInfoCompleteList* is empty, skip other steps.
9. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.13 with the following output parameters
   - out *cap* - access control capabilities
10. If *cap*.MaxLimit is less than 2, skip other steps.
11. ONVIF client invokes **GetAreas** with parameters
    - Token[0] := *invalidToken*
    - Token[1] := *areaInfoCompleteList*[0].token
12. The DUT responds with **GetAreasResponse** message with parameters
    - AreaInfo list =: *areaList*
13. If *areaList* is empty, FAIL the test.
14. If *areaList* contains more than one item, FAIL the test.
15. If *areaList*[0].token does not equal to *areaInfoCompleteList*[0].token, FAIL the test.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 4.15.9 GET AREAS - TOO MANY ITEMS

**Test Case ID:** ACCESSCONTROL-15-1-9

**Specification Coverage:** GetAreas command (ONVIF Access Control Service Specification)

**Feature Under Test:** GetAreas

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify Get Areas if there are more items than MaxLimit in request.

**Pre-Requisite:** Access Control Service is received from the DUT. Area Management is supported by the DUT as indicated by AreaManagementSupported capability.

**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 areas by following the procedure mentioned in Annex A.14 with the following output parameters
   - out *areaCompleteList* - complete area list
4. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.13 with the following output parameters
   - out *cap* - access control capabilities
5. If *areaCompleteList*.token items number is less than *cap*.MaxLimit or equal to *cap*.MaxLimit, skip other steps.
6. Set the following:
   - *tokenList* := [subset of *areaCompleteList*.token values with items number equal to *cap*.MaxLimit + 1]
7. ONVIF client invokes **GetAreas** 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

## 4.15.10 CREATE AREA - NOT EMPTY AREA TOKEN

**Test Case ID:** ACCESSCONTROL-15-1-10

**Specification Coverage:** CreateArea command (ONVIF Access Control Service Specification), Area (ONVIF Access Control Service Specification)

**Feature Under Test:** CreateArea

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** Create Area with not Empty Token Verification.

**Pre-Requisite:** Access Control Service is received from the DUT. The DUT shall have enough free storage capacity for one additional area. Area Management is supported by the DUT as indicated by AreaManagementSupported capability.

**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 areas by following the procedure mentioned in Annex A.14 with the following output parameters

   - out *areaCompleteList1* - complete area list

4. ONVIF Client checks free storage for additional area by following the procedure mentioned in Annex A.29 with the following input and output parameters

   - in *areaCompleteList1* - complete area list

   - out (optional) *areaToRestore* - deleted area

5. ONVIF client invokes **CreateArea** with parameters

   - Area.token := "AreaToken"
   - Area.Name := "Test Name"
   - Area.Description := "Test Description"

6. The DUT sends **env:Sender\ter:InvalidArgVal** SOAP 1.2 fault.

7. ONVIF Client restores area deleted at step 4 if any.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 4.15.11 MODIFY AREA WITH INVALID TOKEN

**Test Case ID:** ACCESSCONTROL-15-1-11

**Specification Coverage:** AreaInfo (ONVIF Access Control Service Specification), Area (ONVIF Access Control Service Specification), ModifyArea command (ONVIF Access Control Service Specification)

**Feature Under Test:** ModifyArea

**WSDL Reference:** accesscontrol.wsdl, door.wsdl

**Test Purpose:** To verify modifying of area with invalid token.

**Pre-Requisite:** Access Control Service is received from the DUT. Area Management is supported by the DUT as indicated by AreaManagementSupported capability.

**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 area info by following the procedure mentioned in Annex A.2 with the following output parameters

- out *areaInfoCompleteList* - complete area info list

4. Set the following:
    - *invalidToken* := value not equal to any *areaInfoCompleteList*.token
5. ONVIF client invokes **ModifyArea** with parameters
    - Area.token := *invalidToken*
    - Area.Name := "Test Name"
    - Area.Description skipped
6. The DUT returns **env:Sender\ter:InvalidArgVal\ter:NotFound** SOAP 1.2 fault.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **env:Sender\ter:InvalidArgVal\ter:NotFound** SOAP 1.2 fault

### 4.15.12 DELETE AREA WITH INVALID TOKEN

**Test Case ID:** ACCESSCONTROL-15-1-12

**Specification Coverage:** DeleteArea command (ONVIF Access Control Service Specification)

**Feature Under Test:** DeleteArea

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify deleting of area with invalid token.

**Pre-Requisite:** Access Control Service is received from the DUT. Area Management is supported by the DUT as indicated by AreaManagementSupported capability.

**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 area info by following the procedure mentioned in Annex A.23 with the following output parameters

    - out *areaInfoCompleteList* - complete area info list

4. Set the following:

    - *invalidToken* := value not equal to any *areaInfoCompleteList*.token

5. ONVIF Client invokes **DeleteArea** with parameters

    - Token := *invalidToken*

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

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **env:Sender\ter:InvalidArgVal\ter:NotFound** SOAP 1.2 fault

### 4.15.13 CREATE NEW AREA WITH SET AREA

**Test Case ID:** ACCESSCONTROL-15-1-13

**Specification Coverage:** SetArea command (ONVIF Access Control Service Specification), Area (ONVIF Access Control Service Specification)

**Feature Under Test:** SetArea

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

**Test Purpose:** To verify creation of area using SetArea command and generating of appropriate notifications.

**Pre-Requisite:** Access Control Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional area. Client Supplied Token is supported by the DUT as indicated by ClientSuppliedTokenSupported capability.

**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 areas by following the procedure mentioned in Annex A.14 with the following output parameters

   - out *areaCompleteList1* - complete area list

4. Set *areaToken* := token that differs from tokens listed in *areaCompleteList1*.

5. ONVIF Client checks free storage for additional area by following the procedure mentioned in Annex A.29 with the following input and output parameters

   - in *areaCompleteList1* - complete area list
   - out (optional) *areaToRestore* - deleted area

6. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.3 with the following input and output parameters

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

7. ONVIF client invokes **SetArea** with parameters

   - Area.token := *areaToken*
   - Area.Name := "Test Name"
   - Area.Description := "Test Description"

8. The DUT responds with empty **SetAreaResponse** message

9. ONVIF Client retrieves and checks **tns1:Configuration/Area/Changed** event for the specified Area token by following the procedure mentioned in Annex A.31 with the following input and output parameters

   - in *s* - Subscription reference
   - in *currentTime* - current time for the DUT
   - in *terminationTime* - subscription termination time
   - in *areaToken* - Area token

10. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.4 with the following input and output parameters

    - in *s* - Subscription reference

11. ONVIF Client retrieves an area by following the procedure mentioned in Annex A.32 with the following input and output parameters

    - in *areaToken* - area token

    - out *areaList* - area list

12. If *areaList*[0] item does not have equal field values to values from step 7, FAIL the test and go step 19.

13. ONVIF Client retrieves an area info by following the procedure mentioned in Annex A.33 with the following input and output parameters

    - in *areaToken* - area token

    - out *areaInfoList* - area info list

14. If *areaInfoList*[0] item does not have equal field values to values from step 7, FAIL the test and go step 19.

15. ONVIF Client retrieves a complete area information list by following the procedure mentioned in Annex A.2 with the following input and output parameters

    - out *areaInfoCompleteList* - area info list

16. If *areaInfoCompleteList* does not have AreaInfo[token = *areaToken*] item with equal field values to values from step 7, FAIL the test and go step 19.

17. ONVIF Client retrieves a complete list of areas by following the procedure mentioned in Annex A.14 with the following input and output parameters

    - out *areaCompleteList2* - area list

18. If *areaCompleteList2* does not have Area[token = *areaToken*] item with equal field values to values from step 7, FAIL the test and go step 19.

19. ONVIF Client deletes the area by following the procedure mentioned in Annex A.34 to restore DUT configuration with the following input and output parameters

    - in *areaToken* - area token

20. ONVIF Client restores area deleted at step 5 if any.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

**Note:** The following fields are compared at steps 12, 14, 16, and 18:

- Area:
  - token
  - Name
  - Description

## 4.15.14 MODIFY AREA WITH SET AREA

**Test Case ID:** ACCESSCONTROL-15-1-14

**Specification Coverage:** SetArea command (ONVIF Access Control Service Specification), Area (ONVIF Access Control Service Specification)

**Feature Under Test:** SetArea

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

**Test Purpose:** To verify modifying of area using SetArea and generating of access point changed notifications.

**Pre-Requisite:** Access Control Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional area. Client Supplied Token is supported by the DUT as indicated by ClientSuppliedTokenSupported capability.

**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 areas by following the procedure mentioned in Annex A.14 with the following output parameters

- out *areaCompleteList1* - complete area list

4. ONVIF Client checks free storage for additional area by following the procedure mentioned in Annex A.29 with the following input and output parameters

   - in *areaCompleteList1* - complete area list
   - out (optional) *areaToRestore* - deleted area

5. ONVIF Client creates an area by following the procedure mentioned in Annex A.19 with the following input and output parameters

   - in "Test Name1" - Access Point Name
   - in "Test Description1" - Access Point Description
   - out *areaToken* - area Token

6. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.3 with the following input and output parameters

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

7. ONVIF client invokes **SetArea** with parameters

   - Area.token := *areaToken*
   - Area.Name := "Test Name2"
   - Area.Description := "Test Description2"

8. The DUT responds with empty **SetAreaResponse** message.

9. ONVIF Client retrieves and checks **tns1:Configuration/Area/Changed** event for the specified Area token by following the procedure mentioned in Annex A.31 with the following input and output parameters

   - in *s* - Subscription reference
   - in *currentTime* - current time for the DUT
   - in *terminationTime* - subscription termination time

- in *areaToken* - Area token

10. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.4 with the following input and output parameters

    - in *s* - Subscription reference

11. ONVIF Client retrieves an area by following the procedure mentioned in Annex A.32 with the following input and output parameters

    - in *areaToken* - area token

    - out *areaList* - area list

12. If *areaList*[0] item does not have equal field values to values from step 7, FAIL the test and go step 19.

13. ONVIF Client retrieves an area info by following the procedure mentioned in Annex A.33 with the following input and output parameters

    - in *areaToken* - area token

    - out *areaInfoList* - area info list

14. If *areaInfoList*[0] item does not have equal field values to values from step 7, FAIL the test and go step 19.

15. ONVIF Client retrieves a complete area information list by following the procedure mentioned in Annex A.2 with the following input and output parameters

    - out *areaInfoCompleteList* - area info list

16. If *areaInfoCompleteList* does not have AreaInfo[token = *areaToken*] item with equal field values to values from step 7, FAIL the test and go step 19.

17. ONVIF Client retrieves a complete list of areas by following the procedure mentioned in Annex A.14 with the following input and output parameters

    - out *areaCompleteList2* - area list

18. If *areaCompleteList2* does not have Area[token = *areaToken*] item with equal field values to values from step 7, FAIL the test and go step 19.

19. ONVIF Client deletes the area by following the procedure mentioned in Annex A.34 to restore DUT configuration with the following input and output parameters

    - in *areaToken* - area token

20. ONVIF Client restores area deleted at step 4 if any.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

**Note:** The following fields are compared at steps 12, 14, 16, and 18:

- Area:
  - token
  - Name
  - Description

## 4.15.15 SET AREA - EMPTY AREA TOKEN

**Test Case ID:** ACCESSCONTROL-15-1-15

**Specification Coverage:** SetArea command (ONVIF Access Control Service Specification)

**Feature Under Test:** SetArea

**WSDL Reference:** accesscontrol.wsdl

**Test Purpose:** To verify SetArea command with empty area token.

**Pre-Requisite:** Access Control Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional area. Client Supplied Token is supported by the DUT as indicated by ClientSuppliedTokenSupported capability.

**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 areas by following the procedure mentioned in Annex A.14 with the following output parameters

- out *areaCompleteList1* - complete area list

4. ONVIF Client checks free storage for additional area by following the procedure mentioned in Annex A.29 with the following input and output parameters

    - in *areaCompleteList1* - complete area list
    - out (optional) *areaToRestore* - deleted area

5. ONVIF client invokes **SetArea** with parameters

    - Area.token := ""
    - Area.Name := "Test Name"
    - Area.Description skipped

6. The DUT responds with **env:Sender\ter:InvalidArgVal** SOAP 1.2 fault.

7. ONVIF Client restores area deleted at step 4 if any.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

# Annex A Helper Procedures and Additional Notes

## A.1 Get Complete Access Point Info List

**Name:** HelperGetCompleteAccessPointInfoList

**Procedure Purpose:** Helper procedure to retrieve complete access points info list.

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

**Input:** None

**Returns:** Complete access points info list (*accessPointInfoCompleteList*). Number of access points (*accessPointsNumber*).

**Procedure:**

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

   3.1. ONVIF client invokes **GetAccessPointInfoList** with parameters
   - Limit is skipped
   - StartReference := *nextStartReference*

   3.2. The DUT responds with **GetAccessPointInfoListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - AccessPointInfo list =: *accessPointInfoListPart*

   3.3. Set *accessPointInfoCompleteList* := *accessPointInfoCompleteList* + *accessPointInfoListPart*.

4. Set *accessPointsNumber* := number of AccessPointInfo items in *accessPointInfoCompleteList*.

**Procedure Result:**

**PASS -**

- The DUT passed all assertions.

**FAIL -**

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

## A.2 Get Complete Area Info List

The following algorithm will be used to get a complete list of Areas:

1. ONVIF Client will invoke **GetAreaInfoList** request (no Limit, no StartReference) to retrieve the first part of Area Information list from the DUT.
2. Verify the **GetAreaInfoListResponse** message from the DUT.
3. If **GetAreaInfoListResponse** message contains NextStartReference, repeat steps 1-2 with StartReference = [current NextStartReference]. Otherwise, skip other steps and finalize getting complete area list.

The complete ordered list of areas with information will be made by the means of uniting all **GetAreaInfoListResponse** messages. Also the total number of areas will be calculated.

## A.3 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.4 Delete Subscription

**Name:** HelperDeleteSubscription

**Procedure Purpose:** Helper procedure to delete supscribtion.

**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.5 Retrieve CredentialNotFound Event by PullPoint

**Name:** HelperPullDeniedCredentialNotFound

**Procedure Purpose:** Helper procedure to retrieve and check tns1:AccessControl/Denied/CredentialNotFound 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 points list (*accessPointsList*).

**Returns:** None

**Procedure:**

1. Until *oprationDelay* timeout expires, repeat the following steps:

   1.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.

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

   - Timeout := PT60S
   - MessageLimit := 1

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

   - CurrentTime =: *ct*
   - TerminationTime =: *tt*
   - NotificationMessage list =: *notificationMessageList*

   1.4. If *notificationMessageList* is not empty:

   1.4.1. If *notification*.Topic is not equal to **tns1:AccessControl/Denied/CredentialNotFound**, FAIL the test and skip other steps.

   1.4.2. If *notification*.Message.Message does not have Source.SimpleItem with Name = "AccessPointToken" and Value which is equal to one of AccessPointInfo.token attribute from *accessPointsList*, FAIL the test and skip other steps.

   1.4.3. If *notification*.Message.Message does not have Data.SimpleItem with Name = "IdentifierType" and Value with type is equal to "xs:string", FAIL the test and skip other steps.

   1.4.4. If *notification*.Message.Message does not have Data.SimpleItem with Name = "IdentifierValue" and Value with type is equal to "xs:hexBinary", FAIL the test and skip other steps.

1.4.5. Skip other steps and finish the procedure.

1.5. If *oprationDelay* timeout expires for step 1 without Notification, FAIL the test and skip other steps.

**Procedure Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

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

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

## A.6 Retrieve Access Granted Identifier Event by PullPoint

**Name:** HelperPullAccessGrantedIdentifier

**Procedure Purpose:** Helper procedure to retrieve and check tns1:AccessControl/AccessGranted/Identifier 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*).

**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 second}.

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

   - Timeout := PT60S
   - MessageLimit := 1

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

   - CurrentTime =: *ct*
   - TerminationTime =: *tt*
   - NotificationMessage list =: *notificationMessageList*

1.4. If *notificationMessageList* is not empty:

1.4.1. Set *message* := *notificationMessageList*[0].Topic.Message.Message.

1.4.2. If *message* contains PropertyOperation attribute, FAIL the test, restore the DUT state, and skip other steps.

1.4.3. If *message*.Source does not contain SimpleItem item with Name = "AccessPointToken", FAIL the test, restore the DUT state, and skip other steps.

1.4.4. If *message*.Source.SimpleItem with Name = "AccessPointToken" does not have Value value with type "pt:ReferenceToken", FAIL the test, restore the DUT state, and skip other steps.

1.4.5. If *message*.Data does not contain SimpleItem item with Name = "IdentifierType", FAIL the test, restore the DUT state, and skip other steps.

1.4.6. If *message*.Source.SimpleItem with Name = "IdentifierType" does not have Value value with type "xs:string", FAIL the test, restore the DUT state, and skip other steps.

1.4.7. If *message*.Data does not contain SimpleItem item with Name = "FormatType", FAIL the test, restore the DUT state, and skip other steps.

1.4.8. If *message*.Source.SimpleItem with Name = "FormatType" does not have Value value with type "xs:string", FAIL the test, restore the DUT state, and skip other steps.

1.4.9. If *message*.Data does not contain SimpleItem item with Name = "IdentifierValue", FAIL the test, restore the DUT state, and skip other steps.

1.4.10. If *message*.Source.SimpleItem with Name = "IdentifierValue" does not have Value value with type "xs:hexBinary", FAIL the test, restore the DUT state, and skip other steps.

1.4.11. Skip other steps of the procedure.

2. If *operationDelay* timeout expires for step 1 and no notifications were recieved, FAIL the test, restore the DUT state, and skip other steps.

**Procedure Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

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

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

## A.7 Retrieve Denied Identifier Event by PullPoint

**Name:** HelperPullDeniedIdentifier

**Procedure Purpose:** Helper procedure to retrieve and check tns1:AccessControl/Denied/Identifier 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*).

**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 second}.

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

   - Timeout := PT60S
   - MessageLimit := 1

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

   - CurrentTime =: *ct*
   - TerminationTime =: *tt*
   - NotificationMessage list =: *notificationMessageList*

   1.4. If *notificationMessageList* is not empty:

      1.4.1. Set *message* := *notificationMessageList*[0].Topic.Message.Message.

      1.4.2. If *message* contains PropertyOperation attribute, FAIL the test, restore the DUT state, and skip other steps.

      1.4.3. If *message*.Source does not contain SimpleItem item with Name = "AccessPointToken", FAIL the test, restore the DUT state, and skip other steps.

1.4.4. If *message*.Source.SimpleItem with Name = "AccessPointToken" does not have Value value with type "pt:ReferenceToken", FAIL the test, restore the DUT state, and skip other steps.

1.4.5. If *message*.Data does not contain SimpleItem item with Name = "IdentifierType", FAIL the test, restore the DUT state, and skip other steps.

1.4.6. If *message*.Source.SimpleItem with Name = "IdentifierType" does not have Value value with type "xs:string", FAIL the test, restore the DUT state, and skip other steps.

1.4.7. If *message*.Data does not contain SimpleItem item with Name = "FormatType", FAIL the test, restore the DUT state, and skip other steps.

1.4.8. If *message*.Source.SimpleItem with Name = "FormatType" does not have Value value with type "xs:string", FAIL the test, restore the DUT state, and skip other steps.

1.4.9. If *message*.Data does not contain SimpleItem item with Name = "IdentifierValue", FAIL the test, restore the DUT state, and skip other steps.

1.4.10. If *message*.Source.SimpleItem with Name = "IdentifierValue" does not have Value value with type "xs:hexBinary", FAIL the test, restore the DUT state, and skip other steps.

1.4.11. If *message*.Data does not contain SimpleItem item with Name = "Reason", FAIL the test, restore the DUT state, and skip other steps.

1.4.12. If *message*.Source.SimpleItem with Name = "Reason" does not have Value value with type "xs:string", FAIL the test, restore the DUT state, and skip other steps.

1.4.13. Skip other steps of the procedure.

2. If *operationDelay* timeout expires for step 1 and no notifications were recieved, FAIL the test, restore the DUT state, and skip other steps.

**Procedure Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

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

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

## A.8 Retrieve Request Identifier Event by PullPoint

**Name:** HelperPullRequestIdentifier

**Procedure Purpose:** Helper procedure to retrieve and check tns1:AccessControl/Request/Identifier 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*).

**Returns:** Token of access point for which **tns1:AccessControl/Request/Identifier** notification was recieved (*accessPointToken*).

**Procedure:**

1. Until *operationDelay* timeout expires, repeat the following steps:

   1.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.

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

   - Timeout := PT60S
   - MessageLimit := 1

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

   - CurrentTime =: *ct*
   - TerminationTime =: *tt*
   - NotificationMessage list =: *notificationMessageList*

   1.4. If *notificationMessageList* is not empty:

      1.4.1. Set *message* := *notificationMessageList*[0].Topic.Message.Message.

      1.4.2. If *message* contains PropertyOperation attribute, FAIL the test, restore the DUT state, and skip other steps.

      1.4.3. If *message*.Source does not contain SimpleItem item with Name = "AccessPointToken", FAIL the test, restore the DUT state, and skip other steps.

1.4.4. If *message*.Source.SimpleItem with Name = "AccessPointToken" does not have Value value with type "pt:ReferenceToken", FAIL the test, restore the DUT state, and skip other steps.

1.4.5. If *message*.Data does not contain SimpleItem item with Name = "IdentifierType", FAIL the test, restore the DUT state, and skip other steps.

1.4.6. If *message*.Source.SimpleItem with Name = "IdentifierType" does not have Value value with type "xs:string", FAIL the test, restore the DUT state, and skip other steps.

1.4.7. If *message*.Data does not contain SimpleItem item with Name = "FormatType", FAIL the test, restore the DUT state, and skip other steps.

1.4.8. If *message*.Source.SimpleItem with Name = "FormatType" does not have Value value with type "xs:string", FAIL the test, restore the DUT state, and skip other steps.

1.4.9. If *message*.Data does not contain SimpleItem item with Name = "IdentifierValue", FAIL the test, restore the DUT state, and skip other steps.

1.4.10. If *message*.Source.SimpleItem with Name = "IdentifierValue" does not have Value value with type "xs:hexBinary", FAIL the test, restore the DUT state, and skip other steps.

1.4.11. Skip other steps of the procedure.

2. If *operationDelay* timeout expires for step 1 and no notifications were recieved, FAIL the test, restore the DUT state, and skip other steps.

**Procedure Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

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

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

## A.9 Retrieve Request Timeout Event by PullPoint

**Name:** HelperPullRequestTimeout

**Procedure Purpose:** Helper procedure to retrieve and check tns1:AccessControl/Request/Timeout 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*). Token of access point for which **tns1:AccessControl/Request/Identifier** notification was recieved (*accessPointToken*).

**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 second}.

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

   - Timeout := PT60S
   - MessageLimit := 1

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

   - CurrentTime =: *ct*
   - TerminationTime =: *tt*
   - NotificationMessage list =: *notificationMessageList*

   1.4. If *notificationMessageList* is not empty:

   1.4.1. Set *message* := *notificationMessageList*[0].Topic.Message.Message.

   1.4.2. If *message* contains PropertyOperation attribute, FAIL the test, restore the DUT state, and skip other steps.

   1.4.3. If *message*.Source does not contain SimpleItem item with Name = "AccessPointToken", FAIL the test, restore the DUT state, and skip other steps.

   1.4.4. If *message*.Source.SimpleItem with Name = "AccessPointToken" does not have Value = *accessPointToken*, FAIL the test, restore the DUT state, and skip other steps.

   1.4.5. Skip other steps of the procedure.

2. If *operationDelay* timeout expires for step 1 and no notifications were recieved, FAIL the test, restore the DUT state, and skip other steps.

**Procedure Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

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

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

## A.10 Retrieve AccessGranted Event by PullPoint for Identifier External Authorization

**Name:** HelperPullExternalAuthAccessGrantedIdentifier

**Procedure Purpose:** Helper procedure to retrieve and check tns1:AccessControl/AccessGranted/Anonymous or tns1:AccessControl/AccessGranted/Credential 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*). Token of access point for which **tns1:AccessControl/Request/Identifier** notification was recieved (*accessPointToken*).

**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 second}.

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

   - Timeout := PT60S

   - MessageLimit := 1

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

   - CurrentTime =: *ct*

- TerminationTime =: *tt*
- NotificationMessage list =: *notificationMessageList*

1.4. If *notificationMessageList* is not empty:

1.4.1. Set *message* := *notificationMessageList*[0].Topic.Message.Message.

1.4.2. If *message* contains PropertyOperation attribute, FAIL the test, restore the DUT state, and skip other steps.

1.4.3. If *message*.Source does not contain SimpleItem item with Name = "AccessPointToken", FAIL the test, restore the DUT state, and skip other steps.

1.4.4. If *message*.Source.SimpleItem with Name = "AccessPointToken" does not have Value = *accessPointToken*, FAIL the test, restore the DUT state, and skip other steps.

1.4.5. If *message*.Source does not contain SimpleItem item with Name = "External", FAIL the test, restore the DUT state, and skip other steps.

1.4.6. If *message*.Source.SimpleItem with Name = "External" does not have Value = true, FAIL the test, restore the DUT state, and skip other steps.

1.4.7. Skip other steps of the procedure.

2. If *operationDelay* timeout expires for step 1 and no notifications were recieved, FAIL the test, restore the DUT state, and skip other steps.

**Procedure Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

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

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

## A.11 Retrieve Denied Event by PullPoint for Identifier External Authorization

**Name:** HelperPullExternalAuthAccessDeniedIdentifier

**Procedure Purpose:** Helper procedure to retrieve and check tns1:AccessControl/Denied/Anonymous or tns1:AccessControl/Denied/Credential 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*). Token of access point for which **tns1:AccessControl/Request/Identifier** notification was recieved (*accessPointToken*).

**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 second}.

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

   - Timeout := PT60S
   - MessageLimit := 1

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

   - CurrentTime =: *ct*
   - TerminationTime =: *tt*
   - NotificationMessage list =: *notificationMessageList*

   1.4. If *notificationMessageList* is not empty:

   1.4.1. Set *message* := *notificationMessageList*[0].Topic.Message.Message.

   1.4.2. If *message* contains PropertyOperation attribute, FAIL the test, restore the DUT state, and skip other steps.

   1.4.3. If *message*.Source does not contain SimpleItem item with Name = "AccessPointToken", FAIL the test, restore the DUT state, and skip other steps.

   1.4.4. If *message*.Source.SimpleItem with Name = "AccessPointToken" does not have Value = *accessPointToken*, FAIL the test, restore the DUT state, and skip other steps.

   1.4.5. If *message*.Source does not contain SimpleItem item with Name = "External", FAIL the test, restore the DUT state, and skip other steps.

1.4.6. If *message*.Source.SimpleItem with Name = "External" does not have Value = true, FAIL the test, restore the DUT state, and skip other steps.

1.4.7. If *message*.Source does not contain SimpleItem item with Name = "Reason", FAIL the test, restore the DUT state, and skip other steps.

1.4.8. If *message*.Source.SimpleItem with Name = "Reason" does not have Value with type "xs:string", FAIL the test, restore the DUT state, and skip other steps.

1.4.9. Skip other steps of the procedure.

2. If *operationDelay* timeout expires for step 1 and no notifications were recieved, FAIL the test, restore the DUT state, and skip other steps.

**Procedure Result:**

**PASS -**

- DUT passes all assertions.

**FAIL -**

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

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

## A.12 Get Access Points List

**Name:** HelperGetAccessPointList

**Procedure Purpose:** Helper procedure to get complete access points list.

**Pre-requisite:** Access Control Service is received from the DUT. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability.

**Input:** None.

**Returns:** The complete list of access points (*accessPointCompleteList*).

**Procedure:**

1. ONVIF client invokes **GetAccessPointList** with parameters

    - Limit skipped

    - StartReference skipped

2. The DUT responds with **GetAccessPointListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - AccessPoint list =: *accessPointCompleteList*
3. Until *nextStartReference* is not null, repeat the following steps:

   3.1. ONVIF client invokes **GetAccessPointList** with parameters
   - Limit skipped
   - StartReference := *nextStartReference*

   3.2. The DUT responds with **GetAccessPointListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - AccessPoint list =: *accessPointListPart*

   3.3. Set the following:
   - *accessPointCompleteList* := *accessPointCompleteList* + *accessPointListPart*

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAccessPointListResponse** message

## A.13 Get Service Capabilities

**Name:** HelperGetServiceCapabilities

**Procedure Purpose:** Helper procedure to get service capabilities.

**Pre-requisite:** Access Control 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 –**

- DUT passes all assertions.

**FAIL –**

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

## A.14 Get Areas List

**Name:** HelperGetAreaList

**Procedure Purpose:** Helper procedure to get complete areas list.

**Pre-requisite:** Access Control Service is received from the DUT. Area Management is supported by the DUT as indicated by AreaManagementSupported capability.

**Input:** None.

**Returns:** The complete list of areas (*areaCompleteList*).

**Procedure:**

1. ONVIF client invokes **GetAreaList** with parameters

   - Limit skipped

   - StartReference skipped

2. The DUT responds with **GetAreaListResponse** message with parameters

   - NextStartReference =: *nextStartReference*

   - Area list =: *areaCompleteList*

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

   3.1. ONVIF client invokes **GetAreaList** with parameters

      - Limit skipped

- StartReference := *nextStartReference*

3.2. The DUT responds with **GetAreaListResponse** message with parameters

- NextStartReference =: *nextStartReference*
- Area list =: *areaListPart*

3.3. Set the following:

- *areaCompleteList* := *areaCompleteList* + *areaListPart*

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAreaListResponse** message

## A.15 Free Storage for Additional Access Point

**Name:** HelperCheckFreeStorageForAccessPoint

**Procedure Purpose:** Helper procedure to provide possibility to add an access point.

**Pre-requisite:** Access Control Service is received from the DUT. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability.

**Input:** The complete list of access points (*accessPointCompleteList*).

**Returns:** Removed access point (optional) (*accessPointToRestore*).

**Procedure:**

1. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.13.
2. If *cap*.MaxAccessPoints is skipped set *cap*.MaxAccessPoints := 10.
3. If number of items of *accessPointCompleteList* less than cap.MaxAccessPoints, skip other steps.
4. ONVIF Client find an access point to delete by following the procedure mentioned in Annex A.17 with the following input parameters

- in *accessPointCompleteList* - access point list.
- out (optional) *accessPointToRestore* - deleted access point to restore.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- None.

## A.16 Find Access Point To Delete

**Name:** HelperFindAccessPointToDelete

**Procedure Purpose:** Helper procedure to find an access point in the access point list that can be deleted.

**Pre-requisite:** Access Control Service is received from the DUT. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability.

**Input:** Access Point list (*accessPointList*).

**Returns:** Deleted Access Point (*deletedAccessPoint*).

**Procedure:**

1. For each access point *accessPoint* from *accessPointList* repeat the following steps:

   1.1. ONVIF Client invokes **DeleteAccessPoint** request with parameters

   - Token =: *accessPoint*.@token

   1.2. The DUT responds with SOAP fault message or with **DeleteAccessPointResponse** message.

   1.3. If DUT returns **DeleteAccessPointResponse** message, set *deletedAccessPoint* := *accessPoint* and skip other steps.

2. Fail the test.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **DeleteAccessPointResponse** message or SOAP fault message.

## A.17 Retrieve Access Point Changed Event by PullPoint

**Name:** HelperPullAccessPointChanged

**Procedure Purpose:** Helper procedure to retrieve and check tns1:Configuration/AccessPoint/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 Point token (*accessPointToken*).

**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 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* contains at least one NotificationMessage with Topic value is not equal to "tns1:Configuration/AccessPoint/Changed", FAIL the test, restore the DUT state, and skip other steps.

1.5. If *notificationMessageList* is not empty and the AccessPointToken source simple item in *notificationMessageList* is equal to *accessPointToken*, skip other steps and finish the procedure.

1.6. If *operationDelay* timeout expires for step 1 without Notification with Token source simple item equal to *accessPointToken*, FAIL the test, restore the DUT state, and skip other steps.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

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

## A.18 Get Area Token

**Name:** HelperGetAreaToken

**Procedure Purpose:** Helper procedure to retrieve tokens of existing areas or to create new ares.

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

**Input:** None.

**Returns:** Token of the 1st Area (optional) (*areaToken1*), token of the 2nd Area (optional) (*areaToken2*)

**Procedure:**

1. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.13.

2. If *cap*.MaxAreas is skipped set *cap*.MaxAreas := 10.

3. If *cap*.MaxAreas > 0

   3.1. ONVIF Client retrieves complete list of areas info from the DUT by following the procedure mentioned in Annex A.2 with the following input and output parameters

- out *areaInfoList* - list of area info

3.2. If *areaInfoList* contains at least one item, set *areaToken1* := *areaInfoList*[0].@token

3.3. If *areaInfoList* contains at least two items, set *areaToken2* := *areaInfoList*[1].@token

3.4. If *areaToken1* != NULL and *areaToken2* != NULL, skip other steps and return to test procedure.

3.5. If *cap*.AreaManagementSupported != true, skip other steps and return to test procedure.

3.6. If *areaToken1* = NULL

3.6.1. ONVIF Client creates the 1st area on the DUT by following the procedure mentioned in Annex A.19 with the following input and output parameters

- in "Test Name1" - Access Point Name
- out *areaToken1* - area token

3.7. If *areaToken2* = NULL

3.7.1. If *cap*.MaxAreas < 2, skip other steps and return to test procedure.

3.7.2. ONVIF Client creates the 2nd area on the DUT by following the procedure mentioned in Annex A.19 with the following input and output parameters

- in "Test Name2" - Access Point Name
- out *areaToken2* - area token

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **DeleteAccessPointResponse** message or SOAP fault message.

## A.19 Create Area

**Name:** HelperCreateArea

**Procedure Purpose:** Helper procedure to create an area on the DUT.

**Pre-requisite:** Access Control Service is received from the DUT. Area Management is supported by the DUT as indicated by AreaManagementSupported capability.

**Input:** Area Name *areaName*, Area Description (optional) *areaDescription*,

**Returns:** Area (*areaToken*).

**Procedure:**

1. ONVIF client invokes **CreateArea** with parameters
   - token := ""
   - Name := *areaName*
   - Description := *areaDescription*
2. The DUT responds with **CreateAreaResponse** message with parameters
   - Token =: *areaToken*

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **CreateAreaResponse** message

**Note:** If optional input parameter is skipped, corresponding field of CreateArea request will be skipped.

## A.20 Get Door Token

**Name:** HelperGetDoorToken

**Procedure Purpose:** Helper procedure to retrieve tokens of existing door or to create new door.

**Pre-requisite:** Door Control Service is received from the DUT. Door Entity is supported by the DUT.

**Input:** None.

**Returns:** Token of the door (optional) (*doorToken*)

**Procedure:**

1. ONVIF Client gets the Door Control service capabilities (out *cap*) by following the procedure mentioned in Annex A.22.

2. If *cap*.MaxDoors is skipped set *cap*.MaxDoors := 10.

3. If *cap*.MaxDoors > 0

   3.1. ONVIF Client retrieves complete list of door info from the DUT by following the procedure mentioned in Annex A.23 with the following input and output parameters

   - out *doorInfoList* - list of door info

   3.2. If *doorInfoList* contains at least one item, set *doorToken* := *doorInfoList*[0].@token, skipp other steps and return to test procedure.

   3.3. If *cap*.DoorManagementSupported != true, skip other steps and return to test procedure.

   3.4. ONVIF Client creates door by following the procedure mentioned in Annex A.21 with the following input and output parameters

   - in false - for all door capabilities
   - in "pt:Door" - Door Type
   - in PT60S - ReleaseTime
   - in PT120S - OpenTime
   - out *doorToken* - door token

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- None.

## A.21 Create Door

**Name:** HelperCreateDoor

**Procedure Purpose:** Helper procedure to create door.

**Pre-requisite:** Door Control Service is received from the DUT. Door Management is supported by the DUT as indicated by DoorManagementSupported capability.

**Input:** Door Capabilities (*doorCapabilities*), Door type (*doorType*), Release Time (*releaseTime*), Open Time (*openTime*).

**Returns:** Door Token (*doorToken*).

**Procedure:**

1. ONVIF client invokes **CreateDoor** with parameters

   - Door.token := ""
   - Door.Name := "Test Name"
   - Door.Description := "Test Description"
   - Door.Capabilities.Access := *doorCapabilities*.Access
   - Door.Capabilities.AccessTimingOverride := *doorCapabilities*.AccessTimingOverride
   - Door.Capabilities.Lock := *doorCapabilities*.Lock
   - Door.Capabilities.Unlock := *doorCapabilities*.Unlock
   - Door.Capabilities.Block := *doorCapabilities*.Block
   - Door.Capabilities.DoubleLock := *doorCapabilities*.DoubleLock
   - Door.Capabilities.LockDown := *doorCapabilities*.LockDown
   - Door.Capabilities.LockOpen := *doorCapabilities*.LockOpen
   - Door.Capabilities.DoorMonitor := *doorCapabilities*.DoorMonitor
   - Door.Capabilities.LockMonitor := *doorCapabilities*.LockMonitor
   - Door.Capabilities.DoubleLockMonitor := *doorCapabilities*.DoubleLockMonitor
   - Door.Capabilities.Alarm := *doorCapabilities*.Alarm
   - Door.Capabilities.Tamper := *doorCapabilities*.Tamper
   - Door.Capabilities.Fault := *doorCapabilities*.Fault
   - Door.DoorType := *doorType*
   - Door.Timings.ReleaseTime := *releaseTime*

- Door.Timings.OpenTime := *openTime*
- Door.Timings.ExtendedReleaseTime skipped
- Door.Timings.DelayTimeBeforeRelock skipped
- Door.Timings.ExtendedOpenTime skipped
- Door.Timings.PreAlarmTime skipped

2. The DUT responds with **CreateDoorResponse** message with parameters
   - Token =: *doorToken*

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **CreateDoorResponse** message

## A.22 Get Door Control Service Capabilities

**Name:** HelperGetDoorControlServiceCapabilities

**Procedure Purpose:** Helper procedure to get Door Control service capabilities.

**Pre-requisite:** Door 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 –**

- DUT passes all assertions.

**FAIL –**

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

## A.23 Get Complete Door Info List

The following algorithm will be used to get a complete list of Doors:

1. ONVIF Client will invoke **GetDoorInfoList** request (no Limit, no StartReference) to retrieve the first part of Door Information list from the DUT.
2. Verify the **GetDoorInfoListResponse** message from the DUT.
3. If **GetDoorInfoListResponse** message contains NextStartReference, repeat steps 1-2 with StartReference = [current NextStartReference]. Otherwise, skip other steps and finalize getting complete door list.

The complete ordered list of doors with information will be made by the means of uniting all **GetDoorInfoListResponse** messages. Also, the total number of doors will be calculated.

## A.24 Get Access Point

**Name:** HelperGetAccessPoint

**Procedure Purpose:** Helper procedure to get access point.

**Pre-requisite:** Access Control Control Service is received from the DUT. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability.

**Input:** Access Point Token (*accessPointToken*).

**Returns:** Access Point List (*accessPointList*).

**Procedure:**

1. ONVIF Client invokes **GetAccessPoints** with parameters
    - Token[0] := *accessPointToken*
2. The DUT responds with **GetAccessPointsResponse** message with parameters:
    - AccessPoint list =: *accessPointList*

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

## A.25 Get Access Point Info

**Name:** HelperGetAccessPointInfo

**Procedure Purpose:** Helper procedure to get access point info.

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

**Input:** Access Point Token (*accessPointToken*).

**Returns:** Access Point Info List (*accessPointInfoList*).

**Procedure:**

1. ONVIF Client invokes **GetAccessPointInfo** with parameters
    - Token[0] := *accessPointToken*
2. The DUT sends the **GetAccessPointInfoResponse** message with parameters
    - AccessPointInfo =: *accessPointInfoList*

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAccessPointInfoResponse** message

## A.26 Delete Access Point

**Name:** HelperDeleteAccessPoint

**Procedure Purpose:** Helper procedure to delete access point.

**Pre-requisite:** Access Control Service is received from the DUT. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability.

**Input:** Access Point Token (*accessPointToken*).

**Returns:** None.

**Procedure:**

1. ONVIF Client invokes **DeleteAccessPoint** request with parameters
   - Token =: *accessPointToken*
2. The DUT responds with **DeleteAccessPointResponse** message.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **DeleteAccessPointResponse** message

## A.27 Create Access Point

**Name:** HelperCreateAccessPoint

**Procedure Purpose:** Helper procedure to create access point.

**Pre-requisite:** Access Control Service is received from the DUT. Access Point Management is supported by the DUT as indicated by AccessPointManagementSupported capability.

**Input:** Access Point Name *accessPointName*, Access Point Description (optional) *accessPointDescription*, Area From Token (optional) *areaFrom*, Area To Token (optional) *areaTo*, Access Point Entity Type (optional) *entityType*, Access Point Entity Token *entityToken*, Disable Access Point Capabilities *disableAccessPoint*, Access Point Duress Capabilities (optional) *duress*, Access Point Anonymous Access Capabilities (optional) *anonymousAccess*, Access Point Access Taken Capabilities (optional) *accessTaken*, Access Point External Authorization Capabilities (optional) *externalAuthorization*, Access Point Supported Recognition Types Capabilities (optional) *supportedRecognitionTypes*, Access Point Identifier Access Capabilities (optional) *identifierAccess*, Access Point Supported Security Levels Capabilities (optional) *supportedSecurityLevels*, Access Point Authentication Profile Token (optional) *authenticationProfileToken*.

**Returns:** Access Point Token (*accessPointToken*).

**Procedure:**

1. ONVIF client invokes **CreateAccessPoint** with parameters

- AccessPoint.token := ""
- AccessPoint.Name := *accessPointName*
- AccessPoint.Description := *accessPointDescription*
- AccessPoint.AreaFrom := *areaFrom*
- AccessPoint.AreaTo := *areaTo*
- AccessPoint.EntityType := *entityType*
- AccessPoint.Entity := *entityToken*
- AccessPoint.Capabilities.DisableAccessPoint := *disableAccessPoint*
- AccessPoint.Capabilities.Duress := *duress*
- AccessPoint.Capabilities.AnonymousAccess := *anonymousAccess*
- AccessPoint.Capabilities.AccessTaken := *accessTaken*
- AccessPoint.Capabilities.ExternalAuthorization := *externalAuthorization*
- AccessPoint.Capabilities.SupportedRecognitionTypes := *supportedRecognitionTypes*
- AccessPoint.Capabilities.IdentifierAccess := *identifierAccess*
- AccessPoint.Capabilities.SupportedSecurityLevels := *supportedSecurityLevels*
- AccessPoint.AuthenticationProfileToken := *authenticationProfileToken*

2. The DUT responds with **CreateAccessPointResponse** message with parameters
    - Token =: *accessPointToken*

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **CreateAccessPointResponse** message

**Note:** If optional input parameter is skipped, corresponding field of CreateAccessPoint request will be skipped.

## A.28 Retrieve Access Point Removed Event by PullPoint

**Name:** HelperPullAccessPointRemoved

**Procedure Purpose:** Helper procedure to retrieve and check tns1:Configuration/AccessPoint/ Removed event with PullMessages.

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

**Input:** Subscription reference (*s*), current time for the DUT (*ct*), Subscription termination time (*tt*) and Access Point token (*accessPointToken*).

**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 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* contains at least one NotificationMessage with Topic value is not equal to "tns1:Configuration/AccessPoint/Removed", FAIL the test, restore the DUT state, and skip other steps.

   1.5. If *notificationMessageList* is not empty and the AccessPointToken source simple item in *notificationMessageList* is equal to *accessPointToken*, skip other steps and finish the procedure.

   1.6. If *operationDelay* timeout expires for step 1 without Notification with Token source simple item equal to *accessPointToken*, FAIL the test, restore the DUT state, and skip other steps.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

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

## A.29 Free Storage for Additional Area

**Name:** HelperCheckFreeStorageForArea

**Procedure Purpose:** Helper procedure to provide possibility to add an area.

**Pre-requisite:** Access Control Service is received from the DUT. Area Management is supported by the DUT as indicated by AreaManagementSupported capability.

**Input:** The complete list of areas (*areaCompleteList*).

**Returns:** Removed area (optional) (*areaToRestore*).

**Procedure:**

1. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.13.
2. If *cap*.MaxAreas is skipped set *cap*.MaxAreas := 10.
3. If number of items of *areaCompleteList* less than cap.MaxAreas, skip other steps.
4. ONVIF Client find an area to delete by following the procedure mentioned in Annex A.30 with the following input parameters
   - in *areaCompleteList* - area list.
   - out (optional) *areaToRestore* - deleted area to restore.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- None.

## A.30 Find Area To Delete

**Name:** HelperFindAreaToDelete

**Procedure Purpose:** Helper procedure to find an area in the area list that can be deleted.

**Pre-requisite:** Access Control Service is received from the DUT. Area Management is supported by the DUT as indicated by AreaManagementSupported capability.

**Input:** Area list (*areaList*).

**Returns:** Deleted Area (*deletedArea*).

**Procedure:**

1. For each area *area* from *areaList* repeat the following steps:

   1.1. ONVIF Client invokes **DeleteArea** request with parameters

   - Token =: *area*.@token

   1.2. The DUT responds with SOAP fault message or with **DeleteAreaResponse** message.

   1.3. If DUT returns **DeleteAreaResponse** message, set *deletedArea* := *area* and skip other steps.

2. Fail the test.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **DeleteAreaResponse** message or SOAP fault message.

## A.31 Retrieve Area Changed Event by PullPoint

**Name:** HelperPullAreaChanged

**Procedure Purpose:** Helper procedure to retrieve and check tns1:Configuration/Area/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 Area token (*areaToken*).

**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 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* contains at least one NotificationMessage with Topic value is not equal to "tns1:Configuration/Area/Changed", FAIL the test, restore the DUT state, and skip other steps.

   1.5. If *notificationMessageList* is not empty and the AreaToken source simple item in *notificationMessageList* is equal to *areaToken*, skip other steps and finish the procedure.

   1.6. If *operationDelay* timeout expires for step 1 without Notification with Token source simple item equal to *areaToken*, FAIL the test, restore the DUT state, and skip other steps.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

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

## A.32 Get Area

**Name:** HelperGetArea

**Procedure Purpose:** Helper procedure to get area.

**Pre-requisite:** Access Control Control Service is received from the DUT. Area Management is supported by the DUT as indicated by AreaManagementSupported capability.

**Input:** Area Token (*areaToken*).

**Returns:** Area List (*areaList*).

**Procedure:**

1. ONVIF Client invokes **GetAreas** with parameters
   - Token[0] := *areaToken*
2. The DUT responds with **GetAreasResponse** message with parameters:
   - Area list =: *areaList*

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

## A.33 Get Area Info

**Name:** HelperGetAreaInfo

**Procedure Purpose:** Helper procedure to get area info.

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

**Input:** Area Token (*areaToken*).

**Returns:** Area Info List (*areaInfoList*).

**Procedure:**

1. ONVIF Client invokes **GetAreaInfo** with parameters
   - Token[0] := *areaToken*
2. The DUT sends the **GetAreaInfoResponse** message with parameters
   - AreaInfo =: *areaInfoList*

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAreaInfoResponse** message

## A.34 Delete Area

**Name:** HelperDeleteArea

**Procedure Purpose:** Helper procedure to delete area.

**Pre-requisite:** Access Control Service is received from the DUT. Area Management is supported by the DUT as indicated by AreaManagementSupported capability.

**Input:** Area Token (*areaToken*).

**Returns:** None.

**Procedure:**

1. ONVIF Client invokes **DeleteArea** request with parameters
   - Token =: *areaToken*
2. The DUT responds with **DeleteAreaResponse** message.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **DeleteAreaResponse** message

## A.35 Retrieve Area Removed Event by PullPoint

**Name:** HelperPullAreaRemoved

**Procedure Purpose:** Helper procedure to retrieve and check tns1:Configuration/Area/Removed event with PullMessages.

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

**Input:** Subscription reference (*s*), current time for the DUT (*ct*), Subscription termination time (*tt*) and Area token (*areaToken*).

**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 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* contains at least one NotificationMessage with Topic value is not equal to "tns1:Configuration/Area/Removed", FAIL the test, restore the DUT state, and skip other steps.

   1.5. If *notificationMessageList* is not empty and the AreaToken source simple item in *notificationMessageList* is equal to *areaToken*, skip other steps and finish the procedure.

1.6. If *operationDelay* timeout expires for step 1 without Notification with Token source simple item equal to *areaToken*, 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.