ONVIF®

Door 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
14.06	Jun, 2014	The following tests were updated with Test Case ID change: ACCESS DOOR BLOCK DOOR DOUBLE LOCK DOOR LOCK DOOR UNLOCK DOOR LOCK DOWN DOOR LOCK DOWN RELEASE DOOR LOCK OPEN DOOR LOCK OPEN RELEASE DOOR
17.12	Aug 16, 2017	Current document name was changed from Door Control Test Specification to Door Control Device Test Specification. The document formatting was updated.
18.06	Jun 21, 2018	Reformatting document using new template
19.12	Aug 05, 2019	The following test cases and annexes are added according to #1675: DOORCONTROL-8-1-1 GET DOORS DOORCONTROL-8-1-2 GET DOOR LIST - LIMIT DOORCONTROL-8-1-3 GET DOOR LIST - START REFERENCE AND LIMIT DOORCONTROL-8-1-4 GET DOOR LIST - NO LIMIT DOORCONTROL-8-1-5 CREATE DOOR (DOOR CAPABILITIES TRUE) DOORCONTROL-8-1-6 CREATE DOOR (DOOR CAPABILITIES FALSE) DOORCONTROL-8-1-7 MODIFY DOOR DOORCONTROL-8-1-8 DELETE DOOR DOORCONTROL-8-1-9 GET DOORS WITH INVALID TOKEN DOORCONTROL-8-1-10 GET DOORS - TOO MANY ITEMS DOORCONTROL-8-1-11 CREATE DOOR - NOT EMPTY DOOR TOKEN

		DOORCONTROL-8-1-12 MODIFY DOOR WITH INVALID TOKEN DOORCONTROL-8-1-13 DELETE DOOR WITH INVALID TOKEN A.6 Get Door List A.7 Get Service Capabilities A.8 Free Storage for Additional Door A.9 Delete Door A.10 Get Door A.11 Get Door Info A.12 Create Door A.13 Create Pull Point Subscription A.14 Delete Subscription A.15 Retrieve Door Changed Event by PullPoint A.16 Retrieve Door Removed Event by PullPoint Test Policy updated with new Door Management section Scope updated with new Door Management section
19.12	Aug 06, 2019	The following test cases and annexes are added according to #1668: DOORCONTROL-8-1-14 CREATE NEW DOOR WITH SET DOOR (DOOR CAPABILITIES TRUE) DOORCONTROL-8-1-15 CREATE NEW DOOR WITH SET DOOR (DOOR CAPABILITIES FALSE) DOORCONTROL-8-1-16 MODIFY DOOR WITH SET DOOR DOORCONTROL-8-1-17 SET DOOR - EMPTY DOOR TOKEN
20.06	Mar 05, 2020	The following test cases were updated in the scope of #1954: DOORCONTROL-1-1-1 DOOR CONTROL SERVICE CAPABILITIES (reformatted, step 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: DOORCONTROL-6-1-1 DOOR CONTROL – DOOR MODE EVENT DOORCONTROL-6-1-2 DOOR CONTROL – DOOR PHYSICAL STATE EVENT DOORCONTROL-6-1-3 DOOR CONTROL – DOOR PHYSICAL STATE EVENT STATE CHANGE DOORCONTROL-6-1-4 DOOR CONTROL – DOUBLE LOCK PHYSICAL STATE EVENT DOORCONTROL-6-1-5 DOOR CONTROL – DOUBLE LOCK PHYSICAL STATE EVENT STATE CHANGE

		DOORCONTROL-6-1-6 DOOR CONTROL – LOCK PHYSICAL STATE EVENT DOORCONTROL-6-1-7 DOOR CONTROL – LOCK PHYSICAL STATE EVENT STATE CHANGE DOORCONTROL-6-1-8 DOOR CONTROL – DOOR TAMPER EVENT DOORCONTROL-6-1-9 DOOR CONTROL – DOOR TAMPER EVENT STATE CHANGE DOORCONTROL-6-1-10 DOOR CONTROL – DOOR ALARM EVENT DOORCONTROL-6-1-11 DOOR CONTROL – DOOR ALARM EVENT STATE CHANGE DOORCONTROL-6-1-12 DOOR CONTROL – DOOR FAULT EVENT DOORCONTROL-6-1-13 DOOR CONTROL – DOOR FAULT EVENT STATE CHANGE DOORCONTROL-7-1-1 DOOR CONTROL – ADD OR CHANGE DOOR EVENT DOORCONTROL-7-1-2 DOOR CONTROL – REMOVE DOOR EVENT DOORCONTROL-8-1-14 CREATE NEW DOOR WITH SET DOOR (DOOR CAPABILITIES TRUE) DOORCONTROL-8-1-15 CREATE NEW DOOR WITH SET DOOR (DOOR CAPABILITIES FALSE) DOORCONTROL-8-1-16 MODIFY DOOR WITH SET DOOR DOORCONTROL-8-1-17 SET DOOR - EMPTY DOOR TOKEN DOORCONTROL-3-1-28 ACCESS DOOR DOORCONTROL-3-1-29 BLOCK DOOR DOORCONTROL-3-1-30 DOUBLE LOCK DOOR DOORCONTROL-3-1-31 LOCK DOOR DOORCONTROL-3-1-32 UNLOCK DOOR DOORCONTROL-3-1-33 LOCK OPEN DOOR DOORCONTROL-3-1-34 LOCK OPEN RELEASE DOOR DOORCONTROL-3-1-35 LOCK DOWN DOOR DOORCONTROL-3-1-36 LOCK DOWN RELEASE DOOR
20.12	Oct 26, 2020	Access Point Entity added into Pre-Requisite of the following test cases in the scope of #1866: DOORCONTROL-5-1-1 GET ACCESS POINT INFO LIST AND GET DOOR INFO LIST CONSISTENCY
20.12	Oct 26, 2020	Door Entity added into Pre-Requisite of the following test cases in the scope of #1866:

		DOORCONTROL-2-1-1 GET DOOR STATE DOORCONTROL-2-1-2 GET DOOR STATE WITH INVALID TOKEN DOORCONTROL-2-1-3 GET DOOR INFO DOORCONTROL-2-1-7 GET DOOR INFO LIST – START REFERENCE AND LIMIT DOORCONTROL-2-1-10 GET DOOR INFO – TOO MANY ITEMS DOORCONTROL-3-1-10 ACCESS DOOR WITH INVALID TOKEN DOORCONTROL-3-1-11 BLOCK DOOR WITH INVALID TOKEN DOORCONTROL-3-1-12 DOUBLE LOCK DOOR WITH INVALID TOKEN DOORCONTROL-3-1-13 LOCK DOOR WITH INVALID TOKEN DOORCONTROL-3-1-14 UNLOCK DOOR WITH INVALID TOKEN DOORCONTROL-3-1-15 LOCK DOWN DOOR WITH INVALID TOKEN DOORCONTROL-3-1-16 LOCK DOWN RELEASE DOOR WITH INVALID TOKEN DOORCONTROL-3-1-17 LOCK OPEN DOOR WITH INVALID TOKEN DOORCONTROL-3-1-18 LOCK OPEN RELEASE DOOR WITH INVALID TOKEN DOORCONTROL-3-1-19 ACCESS DOOR – COMMAND NOT SUPPORTED DOORCONTROL-3-1-20 BLOCK DOOR – COMMAND NOT SUPPORTED DOORCONTROL-3-1-21 DOUBLE LOCK DOOR – COMMAND NOT SUPPORTED DOORCONTROL-3-1-22 LOCK DOOR – COMMAND NOT SUPPORTED DOORCONTROL-3-1-23 UNLOCK DOOR – COMMAND NOT SUPPORTED DOORCONTROL-3-1-24 LOCK DOWN DOOR – COMMAND NOT SUPPORTED DOORCONTROL-3-1-25 LOCK DOWN RELEASE DOOR – COMMAND NOT SUPPORTED DOORCONTROL-3-1-26 LOCK OPEN DOOR – COMMAND NOT SUPPORTED DOORCONTROL-3-1-27 LOCK OPEN RELEASE DOOR – COMMAND NOT SUPPORTED DOORCONTROL-3-1-28 ACCESS DOOR DOORCONTROL-3-1-29 BLOCK DOOR

		DOORCONTROL-3-1-30 DOUBLE LOCK DOOR DOORCONTROL-3-1-31 LOCK DOOR DOORCONTROL-3-1-32 UNLOCK DOOR DOORCONTROL-3-1-33 LOCK OPEN DOOR DOORCONTROL-3-1-34 LOCK OPEN RELEASE DOOR DOORCONTROL-3-1-35 LOCK DOWN DOOR DOORCONTROL-3-1-36 LOCK DOWN RELEASE DOOR DOORCONTROL-6-1-1 DOOR CONTROL – DOOR MODE EVENT DOORCONTROL-6-1-2 DOOR CONTROL – DOOR PHYSICAL STATE EVENT DOORCONTROL-6-1-3 DOOR CONTROL – DOOR PHYSICAL STATE EVENT STATE CHANGE DOORCONTROL-6-1-4 DOOR CONTROL – DOUBLE LOCK PHYSICAL STATE EVENT DOORCONTROL-6-1-5 DOOR CONTROL – DOUBLE LOCK PHYSICAL STATE EVENT STATE CHANGE DOORCONTROL-6-1-6 DOOR CONTROL – LOCK PHYSICAL STATE EVENT DOORCONTROL-6-1-7 DOOR CONTROL – LOCK PHYSICAL STATE EVENT STATE CHANGE DOORCONTROL-6-1-8 DOOR CONTROL – DOOR TAMPER EVENT DOORCONTROL-6-1-9 DOOR CONTROL – DOOR TAMPER EVENT STATE CHANGE DOORCONTROL-6-1-10 DOOR CONTROL – DOOR ALARM EVENT DOORCONTROL-6-1-11 DOOR CONTROL – DOOR ALARM EVENT STATE CHANGE DOORCONTROL-6-1-12 DOOR CONTROL – DOOR FAULT EVENT DOORCONTROL-6-1-13 DOOR CONTROL – DOOR FAULT EVENT STATE CHANGE
20.12	Oct 26, 2020	The following test cases were updated in the scope of #1866: DOORCONTROL-2-1-4 GET DOOR INFO WITH INVALID TOKEN (step 8 updated) DOORCONTROL-2-1-6 GET DOOR INFO LIST – LIMIT (steps 4, 5, 8.4, 12, 13.3 added, step 8 and 13: condition added) DOORCONTROL-2-1-8 GET DOOR INFO LIST – NO LIMIT (step 7 added)
20.12	Nov 12, 2020	The following test cases were updated in the scope of #2095:

		DOORCONTROL-5-1-1 GET ACCESS POINT INFO LIST AND GET DOOR INFO LIST CONSISTENCY (Pre-Requisite updated)

Table of Contents

1 Introduction

The goal of the ONVIF test specification set is to make it possible to realize fully interoperable IP physical security implementation from different vendors. The set of ONVIF test specification describes the test cases need to verify the [ONVIF 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 Door Control Test Specification acts as a supplementary document to the [ONVIF Network Interface Specs], illustrating test cases need to be executed and passed. And also this specification acts as an input document to the development of test tool which will be used to test the ONVIF device implementation conformance towards ONVIF standard. This test tool is referred as ONVIF Client hereafter.

1.1 Scope

This ONVIF Door 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 the test cases to test individual requirements of ONVIF devices according to ONVIF Door 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 Door Control Test Specification covers ONVIF Door Control service and Door 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 to get Door Control Service capabilities. It means that the following commands are covered by these test cases:

- GetServices (Device Management Service);
- GetServiceCapabilities.

1.1.2 General

General test cases cover verification to get Door list, information and state. It means that the following commands are covered by these test cases:

- GetDoorInfo;
- GetDoorInfoList;
- GetDoorState.

1.1.3 Door Control

Door Control test cases cover verification of door control commands and door state change. It means that the following commands and events are covered by these test cases:

- tns1:Door/State/DoorMode;
- AccessDoor;
- BlockDoor;
- DoubleLockDoor;
- LockDoor;
- UnlockDoor;
- LockDownDoor;
- LockDownReleaseDoor;
- LockOpenDoor;
- LockOpenReleaseDoor.

1.1.4 Consistency

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

- Access Point Info and Door Info.

1.1.5 Property Events

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

- tns1:Door/State/DoorMode;
- tns1:Door/State/DoorPhysicalState;
- tns1:Door/State/LockPhysicalState;
- tns1:Door/State/DoubleLockPhysicalState;
- tns1:Door/State/DoorAlarm;
- tns1:Door/State/DoorTamper;
- tns1:Door/State/DoorFault.

1.1.6 Door Change Events

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

- tns1:Configuration/Door/Change;
- tns1:Configuration/Door/Removed.

1.1.7 Door Management

Door Management test cases cover verification of getting of door list by GetDoors and GetDoorList commands, creation and modifying of door by CreateDoor, SetDoor, and ModifyDoor commands, and deleting of door by DeleteDoor command.

2 Terms and Definitions

2.1 Definitions

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

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 sensor.
Door Alarm	An abnormal state of the door where door is forced open or held open beyond the permitted time duration
Door Lock	A device that secures a door to prevent access, except when explicitly allowed by the access control system. Lock types include electromagnet, electric strike, etc.
Door Mode	Logical state of the door indicating whether the door is locked, unlocked, blocked, locked down or locked open etc.
Door Monitor	Also known as a Door Contact Sensor
Lock	An operation after which a door is locked and alarm is unmasked.
Momentary Access	An operation which invokes the same logic as upon normal access being granted to a credential.
Tamper Detector	Mechanism commonly available for doors, access points and controllers to detect physical tamper
Unlock	An operation to allow a door to be freely used for passage without any door alarms being triggered.

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.

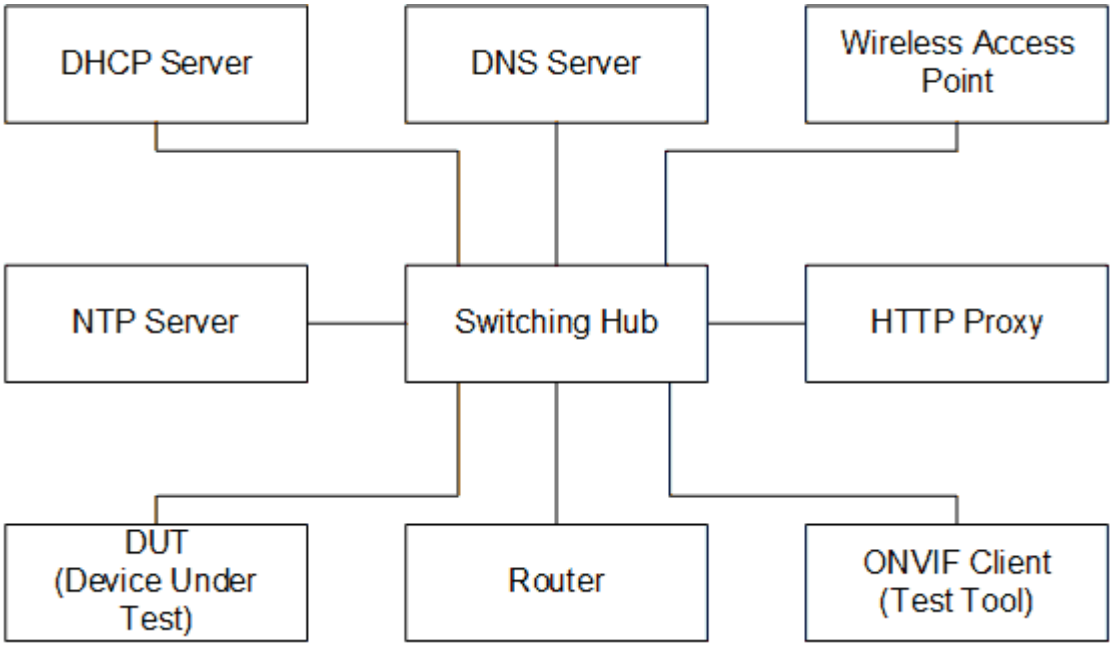
3.1 Test Setup

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 DUT shall be configured with an IPv4 address.
- The DUT shall be IP reachable [in the test configuration].
- The DUT shall be able to be discovered by the Test Tool.
- The DUT shall be configured with the time, i.e. manual configuration of UTC time and if NTP is supported by the DUT then NTP time shall be synchronized with NTP Server.
- The DUT time and Test tool time shall be synchronized with each other either manually or by a common NTP server.
- 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):
  - 4.3.19 ACCESS DOOR
  - 4.3.20 BLOCK DOOR
  - 4.3.21 DOUBLE LOCK DOOR
  - 4.3.22 LOCK DOOR
  - 4.3.23 UNLOCK DOOR
  - 4.3.24 LOCK OPEN DOOR
  - 4.3.25 LOCK OPEN RELEASE DOOR
  - 4.3.26 LOCK DOWN DOOR

  - 4.3.27 LOCK DOWN RELEASE DOOR
  - 4.5.1 DOOR CONTROL – DOOR MODE EVENT
  - 4.5.2 DOOR CONTROL – DOOR PHYSICAL STATE EVENT
  - 4.5.3 DOOR CONTROL – DOOR PHYSICAL STATE EVENT STATE CHANGE
  - 4.5.4 DOOR CONTROL – DOUBLE LOCK PHYSICAL STATE EVENT
  - 4.5.5 DOOR CONTROL – DOUBLE LOCK PHYSICAL STATE EVENT STATE CHANGE
  - 4.5.6 DOOR CONTROL – LOCK PHYSICAL STATE EVENT
  - 4.5.7 DOOR CONTROL – LOCK PHYSICAL STATE EVENT STATE CHANGE
  - 4.5.8 DOOR CONTROL – DOOR TAMPER EVENT
  - 4.5.9 DOOR CONTROL – DOOR TAMPER EVENT STATE CHANGE
  - 4.5.10 DOOR CONTROL – DOOR ALARM EVENT
  - 4.5.11 DOOR CONTROL – DOOR ALARM EVENT STATE CHANGE
  - 4.5.12 DOOR CONTROL – DOOR FAULT EVENT
  - 4.5.13 DOOR CONTROL – DOOR FAULT EVENT STATE CHANGE
  - 4.6.1 DOOR CONTROL – ADD OR CHANGE DOOR EVENT
  - 4.6.2 DOOR CONTROL – REMOVE DOOR EVENT
- At least one Door is configured and added to the DUT.
- At least one Door with Access capability is configured and added to the DUT, if Access capability is supported by the DUT.
- At least one Door with Block capability is configured and added to the DUT, if Block capability is supported by the DUT.
- At least one Door with Double Lock capability is configured and added to the DUT, if Double Lock capability is supported by the DUT.
- At least one Door with Lock capability is configured and added to the DUT, if Lock capability is supported by the DUT.
- At least one Door with Unlock capability is configured and added to the DUT, if Unlock capability is supported by the DUT.

- At least one Door with Lock Down capability is configured and added to the DUT, if Lock Down capability is supported by the DUT.
- At least one Door with Lock Open capability is configured and added to the DUT, if Lock Open capability is supported by the DUT.
- At least one Door with Door Monitor capability is configured and added to the DUT, if Door Monitor capability is supported by the DUT.
- At least one Door with Lock Monitor capability is configured and added to the DUT, if Lock Monitor capability is supported by the DUT.
- At least one Door with Double Lock Monitor capability is configured and added to the DUT, if Double Lock Monitor capability is supported by the DUT.
- At least one Door with Tamper capability is configured and added to the DUT, if Tamper capability is supported by the DUT.
- At least one Door with Alarm capability is configured and added to the DUT, if Alarm capability is supported by the DUT.
- At least one Door with Fault capability is configured and added to the DUT, if Fault capability is supported by 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 Door 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 General

DUT shall give the Door 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 Door. And also at least one Door shall be added and configured on device.

DUT shall support the following commands:

- GetDoorInfo
- GetDoorInfoList
- GetDoorState

DUT shall not return any fault, if GetDoorInfo was invoked for non-exciting Door token. Such tokens shall be ignored.

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

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

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

DUT should return SOAP 1.2 fault message (InvalidArgVal/NotFound), if GetDoorState command was invoked for non-exciting Door token.

Please, refer to Section 4.2 for General Test Cases.

3.3.3 Door Control

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

DUT shall support GetServiceCapabilities command.

DUT shall support following property events:

- tns1:Door/State/DoorMode

DUT shall return capabilities for each specific door.

If DUT returns Access capability as supported by door, then DUT shall support AccessDoor command for this door. Otherwise, DUT should return SOAP 1.2 fault message.

If DUT returns Block capability as supported by door, then DUT shall support BlockDoor command for this door. Otherwise, DUT should return SOAP 1.2 fault message.

If DUT returns DoubleLock capability as supported by door, then DUT shall support DoubleLockDoor command for this door. Otherwise DUT should return SOAP 1.2 fault message.

If DUT returns Lock capability as supported by door, then DUT shall support LockDoor command for this door. Otherwise, DUT should return SOAP 1.2 fault message.

If DUT returns Unlock capability as supported by door, then DUT shall support UnlockDoor command for this door. Otherwise, DUT should return SOAP 1.2 fault message.

If DUT returns LockDown capability as supported by door, then DUT shall support LockDownDoor and LockDownReleaseDoor command for this door. Otherwise, DUT should return SOAP 1.2 fault message.

If DUT returns LockOpen capability as supported by door, then DUT shall support LockOpenDoor and LockOpenReleaseDoor command for this door. Otherwise, DUT should return SOAP 1.2 fault message.

DUT should return SOAP 1.2 fault message (InvalidArgVal/NotFound), if AccessDoor, BlockDoor, DoubleLockDoor, LockDoor, UnlockDoor, LockDownDoor, LockDownReleaseDoor, LockOpenDoor, or LockOpenReleaseDoor command was invoked for non-exciting Door token.

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 4.3 for Door Control Test Cases.

3.3.4 Consistency

DUT shall give the Door 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 a device.

DUT shall support at least one Door. And also at least one Door shall be added and configured on a device.

DUT shall support the following commands:

- GetAccessPointInfoList
- GetDoorInfoList

- DUT shall not return other Door tokens, than listed in GetAccessPointInfoList responses, in Entity field of GetAccessPointInfo's if Type is skipped or equal to tdc:Door.

Please, refer to Section 4.4 for Consistency Test Cases.

3.3.5 Property Events

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

DUT shall support GetServiceCapabilities command.

DUT shall support following property events:

- tns1:Door/State/DoorMode

DUT shall return capabilities for each specific door.

If DUT returns DoorMonitor capability as supported by door, then DUT shall support the following event for this door:

- tns1:Door/State/DoorPhysicalState

If DUT returns DoubleLockMonitor capability as supported by door, then DUT shall support the following event for this door:

- tns1:Door/State/DoubleLockPhysicalState

If DUT returns LockMonitor capability as supported by door, then DUT shall support the following event for this door:

- tns1:Door/State/LockPhysicalState

If DUT returns Tamper capability as supported by door, then DUT shall support the following event for this door:

- tns1:Door/State/DoorTamper

If DUT returns Alarm capability as supported by door, then DUT shall support the following event for this door:

- tns1:Door/State/DoorAlarm

If DUT returns Fault capability as supported by door, then DUT shall support the following event for this door:

- tns1:Door/State/DoorFault

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.

A test operator shall manually change the following properties if required:

- DoorPhysicalState
- LockPhysicalState
- DoubleLockPhysicalState
- Alarm
- Tamper
- Fault

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

3.3.6 Door Change Events

DUT shall give the Door Control Service entry point and Event Service entry point by GetServices command, if DUT supports Door 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 doors, then DUT shall support the following events:

- tns1:Configuration/Door/Change.

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

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

- tns1:Configuration/Door/Removed

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

A test operator shall manually invoke following events if required:

- tns1:Configuration/Door/Change
- tns1:Configuration/Door/Removed

Please, refer to Section 4.6 for Door Change events Test Cases.

3.3.7 Door Management

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

- DUT shall give the Door Control Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.
- DUT shall support the following commands:
  - GetServices
  - GetServiceCapabilities
- If DUT supports Door Management as indicated by the Capabilities.DoorManagementSupported, then DUT shall support the following commands:
  - GetDoors
  - GetDoorList
  - CreateDoor
  - ModifyDoor
  - DeleteDoor
- If DUT supports Client Supplied Token as indicated by the Capabilities.ClientSuppliedTokenSupported, then DUT shall support the following commands:
  - SetDoor
- DUT shall not return more items in GetDoorList responses than specified in service capabilities by MaxLimit.
- DUT shall not return more items in GetDoorList response than specified by Limit parameter in a request.
- DUT shall not return items with the same tokens in GetDoorList responses.

- DUT shall not return any fault if GetDoors was invoked for non-exciting Door token. Such tokens shall be ignored.
- DUT shall return SOAP 1.2 fault message (InvalidArgs/TooManyItems) if more items than MaxLimit was requested by GetDoors command.
- DUT shall return SOAP 1.2 fault message (InvalidArgVal) if CreateDoor command was invoked with non empty Door token.
- DUT shall return SOAP 1.2 fault message (InvalidArgs) if ModifyDoor command was invoked with non-exciting Door token.
- DUT shall return SOAP 1.2 fault message (InvalidArgs) if DeleteDoor command was invoked with non-exciting Door token.
- DUT shall return SOAP 1.2 fault message (InvalidArgVal) if SetDoor command was invoked with empty Door token.
- The following tests are performed if DUT supports Door Management:
  - Getting doors with GetDoors command
  - Getting door list with GetDoorList command with using different Limit and NextReference values
  - Create new door with CreateDoor command
  - Modify door with ModifyDoor command
  - Delete door with DeleteDoor command
  - Getting doors with invalid door token
  - Getting doors with number of requested items is greater than MaxLimit
  - Create door with non empty door token in CreateDoor request
  - Modify door with non existing door token in request
  - Delete door with non existing door token in request
- The following tests are performed if DUT supports Client Supplied Token:
  - Create new door with SetDoor command
  - Modify door with SetDoor command
  - Set door with empty door token in request

Please, refer to Section 4.6 for Door Management Test Cases.

4 Door Control Test Cases

4.1 Capabilities

4.1.1 DOOR CONTROL SERVICE CAPABILITIES

Test Case ID: DOORCONTROL-1-1-1

Specification Coverage: Capability exchange (Specification Coverage), GetServiceCapabilities (ONVIF Door Control Service Specification)

Feature Under Test: GetServiceCapabilities (for Door Control Service)

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify DUT Door Control Service Capabilities.

Pre-Requisite: Door 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*.@MaxLimit is not greater than zero, FAIL the test and skip other steps.
6. If *cap*.@DoorManagementSupported = true and *cap*.@MaxDoors = 0, FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

4.1.2 GET SERVICES AND GET DOOR CONTROL SERVICE CAPABILITIES CONSISTENCY

Test Case ID: DOORCONTROL-1-1-2

Specification Coverage: Capability exchange (Specification Coverage), GetServiceCapabilities (ONVIF Door Control Service Specification)

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

WSDL Reference: devicemgmt.wsdl, doorcontrol.wsdl

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

Pre-Requisite: None.

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

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client 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 Door Control service capabilities of the DUT.
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 Door Controller service if it has Namespace element that is equal to “http://www.onvif.org/ver10/doorcontrol/wsdl”.

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

- MaxLimit attribute values are different;
- MaxDoors attribute values are different;
- MaxDoors attribute is skipped only for GetServices or onle for GetServiceCapabilities request;
- ClientSuppliedTokenSupported attribute values are different;
- DoorManagementSupported attribute values are different.

4.2 General

4.2.1 GET DOOR STATE

Test Case ID: DOORCONTROL-2-1-1

Specification Coverage: GetDoorState (ONVIF Door Control Service Specification)

Feature Under Test: GetDoorState

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify Get Door State.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door 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 doors from the DUT (see Annex A.1).
4. ONVIF Client will invoke **GetDoorState** request (Token = “Token1”, where Token1 is the first DoorInfo.token from the complete list of doors at step 3) to retrieve current state of the door.
5. Verify the **GetDoorStateResponse** message from the DUT.
6. Check that if DoorInfo.Capabilities.DoorMonitor = "true", than DoorState.DoorPhysicalState is present. Otherwise, if DoorInfo.Capabilities.DoorMonitor = "false", then check that DoorState.DoorPhysicalState is skipped.
7. Check that if DoorInfo.Capabilities.LockMonitor = "true", than DoorState.LockPhysicalState is present. Otherwise, if DoorInfo.Capabilities.LockMonitor = "false", then check that DoorState.LockPhysicalState is skipped.
8. Check that if DoorInfo.Capabilities.DoubleLockMonitor = "true", than DoorState.DoubleLockPhysicalState is present. Otherwise, if DoorInfo.Capabilities.DoubleLockMonitor = "false", then check that DoorState.DoubleLockPhysicalState is skipped.
9. Check that if DoorInfo.Capabilities.Alarm = "true", than DoorState.Alarm is present. Otherwise if DoorInfo.Capabilities.Alarm = "false", then check that DoorState.Alarm is skipped.
10. Check that if DoorInfo.Capabilities.Tamper = "true", than DoorState.Tamper is present. Otherwise, if DoorInfo.Capabilities.Tamper = "false", than check that DoorState.Tamper is skipped.
11. Check that if DoorInfo.Capabilities.Fault = "true", than DoorState.Fault is present. Otherwise, if DoorInfo.Capabilities.Fault = "false", then check that DoorState.Fault is skipped.
12. Repeat steps 4-11 for all other tokens from the complete list of doors at step 3.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send valid **GetDoorStateResponse** message.
- The DUT returned DoorState which contains unsupported monitors (DoorPhysicalState, LockPhysicalState, DoubleLockPhysicalState, Alarm, Tamper, and Fault).

- The DUT returned DoorState which does not contain supported monitors (DoorPhysicalState, LockPhysicalState, DoubleLockPhysicalState, Alarm, Tamper, and Fault).
- The DUT did not return at least one Door at step 3.

Note: If the DUT does not return any DoorInfo on a complete list of doors at step 3, skip steps 4-12, fail the test and go to the next test.

4.2.2 GET DOOR STATE WITH INVALID TOKEN

Test Case ID: DOORCONTROL-2-1-2

Specification Coverage: GetDoorState (ONVIF Door Control Service Specification)

Feature Under Test: GetDoorState

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify Get Door State List with invalid Token.

Pre-Requisite: Door Control Service address was received from the DUT. Door 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 **GetDoorState** 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.

4.2.3 GET DOOR INFO

Test Case ID: DOORCONTROL-2-1-3

Specification Coverage: GetDoorInfo (ONVIF Door Control Service Specification)

Feature Under Test: GetDoorInfo

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify Get Door Info.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door 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 doors 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 **GetDoorInfo** request (Token list with subset of DoorInfo.token values from the complete list of doors at step 3 with token number equal to MaxLimit) to retrieve subset of Door Information from the DUT.
7. Verify the **GetDoorInfoResponse** message from the DUT.
8. Verify that all requested DoorInfo items are in **GetDoorInfoResponse** message.
9. ONVIF Client will invoke **GetDoorInfo** request (Token = Token1, where Token1 is the first token from the complete list of doors at step 3) to retrieve Door Information for the specified token from the DUT.
10. Verify the **GetDoorInfoResponse** message from the DUT.
11. Verify that **GetDoorInfoResponse** message contains DoorInfo only for specified token.

12. Repeat steps 9-11 for all other tokens from the complete list of doors at step 3.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Note: these limitations will not be used, if ONVIF Device Test Tool reuses values that were received from the DUT.

4.2.4 GET DOOR INFO WITH INVALID TOKEN

Test Case ID: DOORCONTROL-2-1-4

Specification Coverage: GetDoorInfo (ONVIF Door Control Service Specification)

Feature Under Test: GetDoorInfoList

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify Get Door Info with invalid Token.

Pre-Requisite: Door Control Service address was received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of doors 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 **GetDoorInfo** request (invalid Token).
7. Verify the **GetDoorInfoResponse** message from the DUT. Check that empty list was returned.

8. Door Entity is supported by the DUT and if MaxLimit > 1

   8.1. ONVIF Client will invoke **GetDoorInfo** request (invalid Token, valid Token).

   8.2. Verify the **GetDoorInfoResponse** message from the DUT. Check that list which contains Door for valid Token only was returned.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send valid **GetDoorInfoResponse** message.
- The DUT returned **GetDoorInfoResponse** message that contains at least two DoorInfo items with equal tokens.
- The DUT did not send valid **GetDoorInfoResponse** message.
- The DUT returned **GetDoorInfoResponse** message that contains any DoorInfo items for step 7.
- The DUT returned **GetDoorInfoResponse** message that contains any DoorInfo items other than item for valid Token or does not contains DoorInfo item for valid token step 9.
- The DUT did not return at least one Door at step 3.

Note: If MaxLimit less than 2, skip steps 8.

4.2.5 GET DOOR INFO LIST – LIMIT

Test Case ID: DOORCONTROL-2-1-6

Specification Coverage: GetDoorInfoList (ONVIF Door Control Service Specification)

Feature Under Test: GetDoorInfoList

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify Get Door Info List using Limit.

Pre-Requisite: Door Control Service was received from the DUT. At least one Door is configured and added to the DUT if Door 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 list of door info by following the procedure mentioned in Annex A.1 with the following output parameters
   - out *doorInfoCompleteList* - complete door info list
   - out *doorsNumber* - doors number
4. If DUT supports Door Entity feature and *doorsNumber* < = 0, FAIL the test and skip other steps.
5. If DUT does not support Door Entity feature and *doorsNumber* != 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 **GetDoorInfoList** request (Limit = 1, no StartReference) to retrieve the first item in the list of Door Information from the DUT.
   8.2. The DUT responds with a **GetDoorInfoListResponse** message with parameters
      - NextStartReference
      - List of DoorInfo := *doorInfoList1*
   8.3. If *doorInfoList1* contains number of items greater than 1, FAIL the test and skip other steps.
   8.4. If DUT does not support Door Entity and *doorInfoList1* is not empty, FAIL the test and skip other steps.
9. ONVIF Client will invoke **GetDoorInfoList** request (Limit = MaxLimit, no StartReference) to retrieve the first part of Door Information list from the DUT and is limited by MaxLimit.
10. The DUT responds with a **GetDoorInfoListResponse** message with parameters
    - NextStartReference

- List of DoorInfo := *doorInfoList2*

11. If *doorInfoList2* contains number of items greater than MaxLimit, FAIL the test and skip other steps.
12. If DUT does not support Door Entity and *doorInfoList2* is not empty, FAIL the test and skip other steps.
13. If MaxLimit > 0
    13.1 ONVIF Client invokes **GetDoorInfoList** request (Limit = ItemNumber, no StartReference, where ItemNumber is between 1 and minimal value of MaxLimit and the total number of doors) to retrieve Door Information sublist limited by ItemNumber from the DUT.
    13.2 The DUT responds with a **GetDoorInfoListResponse** message with parameters
    - NextStartReference
    - List of DoorInfo := *doorInfoList3*

    13.3 If *doorInfoList3* contains number of items greater than ItemNumber, FAIL the test and skip other steps.
    13.4 If DUT does not support Door Entity and *doorInfoList3* is not empty, FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send valid **GetDoorInfoListResponse** message.
- The DUT returned more DoorInfo items in **GetDoorInfoListResponse** message than it was specified in Limit parameter.
- The DUT returned more DoorInfo items in **GetDoorInfoListResponse** message than MaxLimit.
- The DUT did not return at least one Door at step 3.

Note: If the DUT does not return any DoorInfo on complete list of doors at step 3, skip steps 4-14, fail the test and go to the next test.

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

4.2.6 GET DOOR INFO LIST – START REFERENCE AND LIMIT

Test Case ID: DOORCONTROL-2-1-7

Specification Coverage: GetDoorInfoList (ONVIF Door Control Service Specification)

Feature Under Test: GetDoorInfoList

WSDL Reference: doorcontrol.wsdl

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

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door 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 door control service capabilities.
4. Verify the **GetServiceCapabilitiesResponse** (MaxLimit) message from the DUT.
5. ONVIF Client will invoke **GetDoorInfoList** request (Limit = MaxLimit, no StartReference) to retrieve the first part of Door Information list from the DUT.
6. Verify the **GetDoorInfoListResponse** message from the DUT. Verify that **GetDoorInfoListResponse** message does not contain number of items greater than Limit. Verify that **GetDoorInfoListResponse** message does not contain number of items greater than MaxLimit.
7. If the number of DoorInfo in **GetDoorInfoListResponse** message contains NextStartReference, repeat steps 5-6 with StartReference = [current NextStartReference]. Otherwise, skip other steps and finalize getting complete door list.
8. Verify that **GetDoorInfoListResponse** messages at step 6 does not contain DoorInfo items with the same tokens values all over the list, i.e. there shall be no **GetDoorInfoListResponse** messages containing the same value for DoorInfo items token.

9. ONVIF Client will invoke **GetDoorInfoList** request (Limit = 1, no StartReference) to retrieve the first part of Door Information list from the DUT.
10. Verify the **GetDoorInfoListResponse** message from the DUT. Verify that **GetDoorInfoListResponse** message does not contain number of items greater than Limit. Verify that **GetDoorInfoListResponse** message does not contain number of items greater than MaxLimit.
11. If the number of DoorInfo in **GetDoorInfoListResponse** message contains NextStartReference, repeat steps 9-10 with StartReference = [current NextStartReference]. Otherwise, skip other steps and finalize getting complete door list.
12. Verify that **GetDoorInfoListResponse** messages at step 10 does not contain DoorInfo items with the same tokens values all over the list, i.e. there shall be no **GetDoorInfoListResponse** messages containing the same value for DoorInfo 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 **GetDoorInfoList** request (Limit = [limit_value], no StartReference, where limit_value is between 1 and MaxLimit) to retrieve the first part of Door Information list from the DUT.
15. Verify the **GetDoorInfoListResponse** message from the DUT. Verify that **GetDoorInfoListResponse** message does not contain number of items greater than Limit. Verify that **GetDoorInfoListResponse** message does not contain number of items greater than MaxLimit.
16. If the number of DoorInfo in **GetDoorInfoListResponse** message contains NextStartReference, repeat steps 14-15 with StartReference = [current NextStartReference]. Otherwise, skip other steps and finalize getting complete door list.
17. Verify that **GetDoorInfoListResponse** messages at step 15 does not contain DoorInfo items with the same tokens values all over the list, i.e. there shall be no **GetDoorInfoListResponse** messages containing the same value for DoorInfo 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 **GetDoorInfoListResponse** message.
- The DUT returned **GetDoorInfoListResponse** message that contains wrong item list than requested according to StartReference and Limit values.
- The DUT returned **GetDoorInfoListResponse** messages for step 6 that contains at least one pair of the same tokens for DoorInfo item all over the messages.
- The DUT returned **GetDoorInfoListResponse** messages for step 10 that contains at least one pair of the same tokens for DoorInfo item all over the messages.
- The DUT returned **GetDoorInfoListResponse** messages for step 15 that contains at least one pair of the same tokens for DoorInfo item all over the messages.
- The DUT returned more DoorInfo items in **GetDoorInfoListResponse** message than it was specified in Limit parameter.
- The DUT returned more DoorInfo items in **GetDoorInfoListResponse** message than MaxLimit.

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

4.2.7 GET DOOR INFO LIST – NO LIMIT

Test Case ID: DOORCONTROL-2-1-8

Specification Coverage: GetDoorInfoList (ONVIF Door Control Service Specification)

Feature Under Test: GetDoorInfoList

WSDL Reference: doorcontrol.wsdl

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

Pre-Requisite: Door Control Service was received from the DUT. At least one Door 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 door control service capabilities.

4. Verify the **GetServiceCapabilitiesResponse** (MaxLimit) message from the DUT.
5. ONVIF Client will invoke **GetDoorInfoList** request (no Limit, no StartReference) to retrieve the first part of Door Information list from the DUT.
6. Verify the **GetDoorInfoListResponse** message from the DUT. Verify that **GetDoorInfoListResponse** message does not contain number of items greater than MaxLimit.
7. If DUT does not support Door Entity and DoorInfoList in **GetDoorInfoListResponse** is not empty, FAIL the test and skip other steps.
8. If **GetDoorInfoListResponse** message contains NextStartReference, repeat steps 5-6 with StartReference = [current NextStartReference]. Otherwise, skip other steps and finalize getting the complete door list.
9. Verify that **GetDoorInfoListResponse** messages at step 6 does not contain DoorInfo items with the same tokens values all over the list, i.e. there shall be no **GetDoorInfoListResponse** messages containing the same value for DoorInfo items token.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

4.2.8 GET DOOR INFO – TOO MANY ITEMS

Test Case ID: DOORCONTROL-2-1-10

Specification Coverage: GetDoorInfo (ONVIF Door Control Service Specification)

Feature Under Test: GetDoorInfo

WSDL Reference: doorcontrol.wsdl

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

Pre-Requisite: Door Control Service was received from the DUT. At least one Door is configured and added to the DUT. Door 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 a complete list of doors 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 doors is less than MaxLimit, skip other steps and go to the text test.
7. ONVIF Client will invoke **GetDoorInfo** request (Token list with subset of DoorInfo.token values from a complete list of doors at step 3 with token number greater than MaxLimit) to retrieve a subset of Door 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 Door at step 3.

Note: If the DUT does not return any DoorInfo on complete list of doors at step 3, skip steps 4-8, fail test and go to the next test.

4.3 Door Control

4.3.1 ACCESS DOOR WITH INVALID TOKEN

Test Case ID: DOORCONTROL-3-1-10

Specification Coverage: AccessDoor (ONVIF Door Control Service Specification)

Feature Under Test: AccessDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify Access Door with invalid Token.

Pre-Requisite: Door Control Service address was received from the DUT. Door 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 **AccessDoor** 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.

4.3.2 BLOCK DOOR WITH INVALID TOKEN

Test Case ID: DOORCONTROL-3-1-11

Specification Coverage: BlockDoor (ONVIF Door Control Service Specification)

Feature Under Test: BlockDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify Block Door with invalid Token.

Pre-Requisite: Door Control Service address was received from the DUT. Door 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 **BlockDoor** 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.

4.3.3 DOUBLE LOCK DOOR WITH INVALID TOKEN

Test Case ID: DOORCONTROL-3-1-12

Specification Coverage: DoubleLockDoor (ONVIF Door Control Service Specification)

Feature Under Test: DoubleLockDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify Double Lock Door with invalid Token.

Pre-Requisite: Door Control Service address was received from the DUT. Door 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 **DoubleLockDoor** 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.

4.3.4 LOCK DOOR WITH INVALID TOKEN

Test Case ID: DOORCONTROL-3-1-13

Specification Coverage: LockDoor (ONVIF Door Control Service Specification)

Feature Under Test: LockDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify Lock Door with invalid Token.

Pre-Requisite: Door Control Service address was received from the DUT. Door 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 **LockDoor** 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.

4.3.5 UNLOCK DOOR WITH INVALID TOKEN

Test Case ID: DOORCONTROL-3-1-14

Specification Coverage: UnlockDoor (ONVIF Door Control Service Specification)

Feature Under Test: UnlockDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify Unlock Door with invalid Token.

Pre-Requisite: Door Control Service address was received from the DUT. Door 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 **UnlockDoor** 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.

4.3.6 LOCK DOWN DOOR WITH INVALID TOKEN

Test Case ID: DOORCONTROL-3-1-15

Specification Coverage: LockDownDoor (ONVIF Door Control Service Specification)

Feature Under Test: LockDownDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify Lock Down Door with invalid Token.

Pre-Requisite: Door Control Service address was received from the DUT. Door 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 **LockDownDoor** 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.

4.3.7 LOCK DOWN RELEASE DOOR WITH INVALID TOKEN

Test Case ID: DOORCONTROL-3-1-16

Specification Coverage: LockDownReleaseDoor (ONVIF Door Control Service Specification)

Feature Under Test: LockDownReleaseDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify Lock Down Release Door with invalid Token.

Pre-Requisite: Door Control Service address was received from the DUT. Door Entity is supported by the DUT. Test Configuration: ONVIF Client and DUT

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **LockDownReleaseDoor** 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.

4.3.8 LOCK OPEN DOOR WITH INVALID TOKEN

Test Case ID: DOORCONTROL-3-1-17

Specification Coverage: LockOpenDoor (ONVIF Door Control Service Specification)

Feature Under Test: LockOpenDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify Lock Open Door with invalid Token.

Pre-Requisite: Door Control Service address was received from the DUT. Door 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 **LockOpenDoor** 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.

4.3.9 LOCK OPEN RELEASE DOOR WITH INVALID TOKEN

Test Case ID: DOORCONTROL-3-1-18

Specification Coverage: LockOpenReleaseDoor (ONVIF Door Control Service Specification)

Feature Under Test: LockOpenReleaseDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify Lock Open Release Door with invalid Token.

Pre-Requisite: Door Control Service address was received from the DUT. Door 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 **LockOpenReleaseDoor** 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.

4.3.10 ACCESS DOOR – COMMAND NOT SUPPORTED

Test Case ID: DOORCONTROL-3-1-19

Specification Coverage: AccessDoor (ONVIF Door Control Service Specification)

Feature Under Test: AccessDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify that fault is returned for Access Door in case Door does not support it.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door 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 doors from the DUT (see Annex A.1).
4. If Door with Token1 (Token1 is the first DoorInfo.token from the complete list of doors at step 3) has DoorInfo.Capabilities.Access equal to true, then skip steps 5-6 and go to the step 7.
5. ONVIF Client will invoke **AccessDoor** request (Token = “Token1”, where Token1 is the first DoorInfo.token from the complete list of doors at step 3) to try changing door state.
6. The DUT will generate SOAP 1.2 fault message.
7. Repeat steps 4-6 for all other tokens from the complete list of doors 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 Door at step 3.

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

4.3.11 BLOCK DOOR – COMMAND NOT SUPPORTED

Test Case ID: DOORCONTROL-3-1-20

Specification Coverage: BlockDoor (ONVIF Door Control Service Specification)

Feature Under Test: BlockDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify that fault is returned for Block Door in case Door does not support it.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door 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 doors from the DUT (see Annex A.1).
4. If Door with Token1 (Token1 is the first DoorInfo.token from the complete list of doors at step 3) has DoorInfo.Capabilities.Block equal to true, then skip steps 5-6 and go to the step 7.
5. ONVIF Client will invoke **BlockDoor** request (Token = “Token1”, where Token1 is the first DoorInfo.token from the complete list of doors at step 3) to try changing door state.
6. The DUT will generate SOAP 1.2 fault message.
7. Repeat steps 5-7 for all other tokens from the complete list of doors 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 Door at step 3.

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

4.3.12 DOUBLE LOCK DOOR – COMMAND NOT SUPPORTED

Test Case ID: DOORCONTROL-3-1-21

Specification Coverage: DoubleLockDoor (ONVIF Door Control Service Specification)

Feature Under Test: DoubleLockDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify that fault is returned for Double Lock Door in case Door does not support it.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door 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 doors from the DUT (see Annex A.1).
4. If Door with Token1 (Token1 is the first DoorInfo.token from the complete list of doors at step 3) has DoorInfo.Capabilities.DoubleLock equal to true, then skip steps 5-6 and go to the step 7.
5. ONVIF Client will invoke **DoubleLockDoor** request (Token = “Token1”, where Token1 is the first DoorInfo.token from the complete list of doors at step 3) to try changing door state.
6. The DUT will generate SOAP 1.2 fault message.
7. Repeat steps 5-7 for all other tokens from complete list of doors 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 Door at step 3.

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

4.3.13 LOCK DOOR – COMMAND NOT SUPPORTED

Test Case ID: DOORCONTROL-3-1-22

Specification Coverage: LockDoor (ONVIF Door Control Service Specification)

Feature Under Test: LockDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify that fault is returned for Lock Door in case Door does not support it.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door 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 doors from the DUT (see Annex A.1).
4. If Door with Token1 (Token1 is the first DoorInfo.token from the complete list of doors at step 3) has DoorInfo.Capabilities.Lock equal to true, then skip steps 5-6 and go to the step 7.
5. ONVIF Client will invoke **LockDoor** request (Token = “Token1”, where Token1 is the first DoorInfo.token from the complete list of doors at step 3) to try changing door state.
6. The DUT will generate SOAP 1.2 fault message.
7. Repeat steps 5-7 for all other tokens from the complete list of doors 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 Door at step 3.

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

4.3.14 UNLOCK DOOR – COMMAND NOT SUPPORTED

Test Case ID: DOORCONTROL-3-1-23

Specification Coverage: UnlockDoor (ONVIF Door Control Service Specification)

Feature Under Test: UnlockDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify that fault is returned for Unlock Door in case Door does not support it.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door 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 doors from the DUT (see Annex A.1).
4. If Door with Token1 (Token1 is the first DoorInfo.token from the complete list of doors at step 3) has DoorInfo.Capabilities.Unlock equal to true, then skip steps 5-6 and go to the step 7.
5. ONVIF Client will invoke **UnlockDoor** request (Token = “Token1”, where Token1 is the first DoorInfo.token from the complete list of doors at step 3) to try changing door state.
6. The DUT will generate SOAP 1.2 fault message.
7. Repeat steps 5-7 for all other tokens from the complete list of doors 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 Door at step 3.

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

4.3.15 LOCK DOWN DOOR – COMMAND NOT SUPPORTED

Test Case ID: DOORCONTROL-3-1-24

Specification Coverage: LockDownDoor (ONVIF Door Control Service Specification)

Feature Under Test: LockDownDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify that fault is returned for Lock Down Door in case Door does not support it.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door 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 doors from the DUT (see Annex A.1).
4. If Door with Token1 (Token1 is the first DoorInfo.token from the complete list of doors at step 3) has DoorInfo.Capabilities.LockDown equal to true, then skip steps 5-6 and go to the step 7.
5. ONVIF Client will invoke **LockDownDoor** request (Token = “Token1”, where Token1 is the first DoorInfo.token from the complete list of doors at step 3) to try changing door state.
6. The DUT will generate SOAP 1.2 fault message.
7. Repeat steps 5-7 for all other tokens from the complete list of doors 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 Door at step 3.

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

4.3.16 LOCK DOWN RELEASE DOOR – COMMAND NOT SUPPORTED

Test Case ID: DOORCONTROL-3-1-25

Specification Coverage: LockDownReleaseDoor (ONVIF Door Control Service Specification)

Feature Under Test: LockDownReleaseDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify that fault is returned for Lock Down Release Door in case Door does not support it.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door 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 doors from the DUT (see Annex A.1).
4. If Door with Token1 (Token1 is the first DoorInfo.token from the complete list of doors at step 3) has DoorInfo.Capabilities.LockDown equal to true, then skip steps 5-6 and go to the step 7.
5. ONVIF Client will invoke **LockDownReleaseDoor** request (Token = “Token1”, where Token1 is the first DoorInfo.token from the complete list of doors at step 3) to try changing door state.
6. The DUT will generate SOAP 1.2 fault message.
7. Repeat steps 5-7 for all other tokens from the complete list of doors 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 Door at step 3.

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

4.3.17 LOCK OPEN DOOR – COMMAND NOT SUPPORTED

Test Case ID: DOORCONTROL-3-1-26

Specification Coverage: LockOpenDoor (ONVIF Door Control Service Specification)

Feature Under Test: LockOpenDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify that fault is returned for Lock Open Door in case Door does not support it.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door 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 doors from the DUT (see Annex A.1).
4. If Door with Token1 (Token1 is the first DoorInfo.token from the complete list of doors at step 3) has DoorInfo.Capabilities.LockOpen equal to true, then skip steps 5-6 and go to the step 7.
5. ONVIF Client will invoke **LockOpenDoor** request (Token = “Token1”, where Token1 is the first DoorInfo.token from the complete list of doors at step 3) to try changing door state.
6. The DUT will generate SOAP 1.2 fault message.
7. Repeat steps 5-7 for all other tokens from the complete list of doors 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 Door at step 3.

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

4.3.18 LOCK OPEN RELEASE DOOR – COMMAND NOT SUPPORTED

Test Case ID: DOORCONTROL-3-1-27

Specification Coverage: LockOpenReleaseDoor (ONVIF Door Control Service Specification)

Feature Under Test: LockOpenReleaseDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify that fault is returned for Lock Open Release Door in case Door does not support it.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door 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 doors from the DUT (see Annex A.1).
4. If Door with Token1 (Token1 is the first DoorInfo.token from the complete list of doors at step 3) has DoorInfo.Capabilities.LockOpen equal to true, then skip steps 5-6 and go to the step 7.
5. ONVIF Client will invoke **LockOpenReleaseDoor** request (Token = “Token1”, where Token1 is the first DoorInfo.token from complete list of doors at step 3) to try changing door state.
6. The DUT will generate SOAP 1.2 fault message.
7. Repeat steps 5-7 for all other tokens from the complete list of doors 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 Door at step 3.

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

4.3.19 ACCESS DOOR

Test Case ID: DOORCONTROL-3-1-28

Specification Coverage: AccessDoor (ONVIF Door Control Service Specification), tns1:Door/State/DoorMode (ONVIF Door Control Service Specification)

Feature Under Test: AccessDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify that Access Door is accepted. To verify Door state change after Access Door command. To verify tns1:Door/State/DoorMode property events.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Access capability is configured and added to the DUT. The doors are closed and there are no schedules. Test Configuration: ONVIF Client and DUT

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get a complete list of doors from the DUT (see Annex A.1).
4. ONVIF Client selects one door from the complete list of doors at step 3 with DoorInfo.Capabilities.Access equal to true and DoorInfo.Capabilities.Lock equal to true. If there is no Door with DoorInfo.Capabilities.Access equal to true, fail the test and skip other steps. If there is no Door with DoorInfo.Capabilities.Access equal to true and DoorInfo.Capabilities.Lock equal to true, then ONVIF Client executes Annex A.4 to start user interaction process, and then goes to step 6.
5. ONVIF Client will execute Annex A.5 to switch the selected door into Locked state.

6. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorMode Topic as Filter and an InitialTerminationTime of timeout1.
7. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
8. Execute Annex A.3 to catch Initialized event for selected door. State of the Door received in Annex A.3 will be assumed as initial State of this Door.
9. ONVIF Client will invoke **AccessDoor** request (Token = [selected Door token]) to change door state.
10. Verify the **AccessDoorResponse** message from the DUT.
11. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
12. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 11 until Notification messages with PropertyOperation = "Changed" contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to selected Door Token with State = “Accessed” is received (Operation Delay option from ONVIF Device Test Tool will be used).
13. Check that all Notification messages with PropertyOperation = "Changed" contain Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token.
14. Check that all Notification messages with PropertyOperation = "Changed" have TopicExpression equal to tns1:Door/State/DoorMode.
15. Check that received Notification message with PropertyOperation = "Changed" contains Data.SimpleItem item with Name = «State» and Value equal to “Accessed”. Check that other Notification messages contain Data.SimpleItem item with Name = «State» and Value with tdc:DoorMode type.
16. Verify received Notification messages (correct value for UTC time, TopicExpression and wsnt:Message).
17. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
18. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 17 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).
19. Check that all Notification messages with PropertyOperation = "Changed" contain Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token.

20. Check that all Notification messages with PropertyOperation = "Changed" have TopicExpression equal to tns1:Door/State/DoorMode.

21. Check that the last Notification message with PropertyOperation = "Changed" contains Data.SimpleItem item with Name = «State» and Value equal to initial Door state from step 8. Check that other Notification messages contain Data.SimpleItem item with Name = «State» and Value with tdc:DoorMode type.

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

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send valid **AccessDoorResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not return at least one Door at step 3.
- The DUT sent an invalid Notification messages with PropertyOperation = "Changed" (correct value for UTC time, TopicExpression and wsnt:Message, Topic = “tns1:Door/State/DoorMode”, contains Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token, contains Data.SimpleItem item with Name = «State» and Value with type equal to tdc:DoorMode) at steps 12 and 18.
- The DUT did not send Notification message with PropertyOperation = "Changed" with State value “Accessed” at step 12.
- The DUT sent the last Notification message with PropertyOperation = "Changed" at step 18 with State value other than initial State at step 8.

Note: If the DUT did not return any door for step 3, skip steps 4-22, fail the test and go to the next test.

Note: If the DUT sends SOAP fault to AccessDoor request then the ONVIF Client deletes Subscription Manager, starts user interaction process (see Annex A.4), and then runs the test from step 6.

Note: If in the Annex A.5 the DUT sends SOAP fault to request to change door mode, then the ONVIF Client starts user interaction process (see Annex A.4), and then runs the test from step 6.

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.

4.3.20 BLOCK DOOR

Test Case ID: DOORCONTROL-3-1-29

Specification Coverage: BlockDoor (ONVIF Door Control Service Specification), tns1:Door/State/DoorMode (ONVIF Door Control Service Specification)

Feature Under Test: BlockDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify that Block Door is accepted. To verify Door state change after Block Door command. To verify tns1:Door/State/DoorMode property events.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Block capability is configured and added to the DUT. The doors are closed and there are no schedules.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get a complete list of doors from the DUT (see Annex A.1).
4. ONVIF Client selects one door from the complete list of doors at step 3 with DoorInfo.Capabilities.Block equal to true and DoorInfo.Capabilities.Locked equal to true. If there is no Door with DoorInfo.Capabilities.Block equal to true, fail the test and skip other steps. If there is no Door with DoorInfo.Capabilities.Block equal to true and

DoorInfo.Capabilities.Lock equal to true, then ONVIF Client executes Annex A.4 to start user interaction process, and then goes to step 6.

5. ONVIF Client will execute Annex A.5 to switch the selected door into Locked state.
6. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorMode Topic as Filter and an InitialTerminationTime of timeout1.
7. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
8. Execute Annex A.3 to catch Initialized event for selected door.
9. ONVIF Client will invoke **BlockDoor** request (Token = [selected Door token]) to change door state.
10. Verify the **BlockDoorResponse** message from the DUT.
11. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
12. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 11 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).
13. Check that all Notification messages with PropertyOperation = "Changed" contain Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token.
14. Check that all Notification messages with PropertyOperation = "Changed" have TopicExpression equal to tns1:Door/State/DoorMode.
15. Verify that the last Notification message with PropertyOperation = "Changed" contains Data.SimpleItem item with Name = «State» and Value equal to “Blocked”. Check that other Notification messages contain Data.SimpleItem item with Name = «State» and Value with tdc:DoorMode type.
16. Verify received Notification messages (correct value for UTC time, TopicExpression and wsnt:Message).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send valid BlockDoorResponse message.

- The DUT did not send CreatePullPointSubscriptionResponse message.
- The DUT did not return at least one Door at step 3.
- The DUT sent an invalid Notification messages with PropertyOperation = "Changed" (correct value for UTC time, TopicExpression and wsnt:Message, Topic = “tns1:Door/State/DoorMode”, contains Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token, contains Data.SimpleItem item with Name = «State» and Value with type equal to tdc:DoorMode) at step 12.
- The DUT sent the last Notification message with PropertyOperation = "Changed" with State value other than “Blocked” at step 12.

Note: If the DUT does not return any door for step 3, skip steps 4-16, fail the test and go to the next test.

Note: If the DUT sends SOAP fault to BlockDoor request then the ONVIF Client deletes Subscription Manager, starts user interaction process (see Annex A.4), and then runs the test from step 6.

Note: If in the Annex A.1 the DUT sends SOAP fault to request to change door mode, then the ONVIF Client starts user interaction process (see Annex A.4), and then runs the test from step 6.

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.

4.3.21 DOUBLE LOCK DOOR

Test Case ID: DOORCONTROL-3-1-30

Specification Coverage: DoubleLockDoor (ONVIF Door Control Service Specification)

Feature Under Test: DoubleLockDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify that Double Lock Door is accepted. To verify Door state change after Double Lock Door command. To verify tns1:Door/State/DoorMode property events.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Double Lock capability is configured and added to the DUT. The doors are closed and there are no schedules.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get complete list of doors from the DUT (see Annex A.1).
4. ONVIF Client selects one door from the complete list of doors at step 3 with DoorInfo.Capabilities.DoubleLock equal to true and DoorInfo.Capabilities.Locked equal to true. If there is no Door with DoorInfo.Capabilities.DoubleLock equal to true, fail the test and skip other steps. If there is no Door with DoorInfo.Capabilities.DoubleLock equal to true and DoorInfo.Capabilities.Lock equal to true, then ONVIF Client executes Annex A.4 to start user interaction process and then goes to step 6.
5. ONVIF Client will execute Annex A.5 to switch the selected door into Locked state.
6. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorMode Topic as Filter and an InitialTerminationTime of timeout1.
7. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
8. Execute Annex A.3 to catch Initialized event for selected door.
9. ONVIF Client will invoke **DoubleLockDoor** request (Token = [selected Door token]) to change door state.
10. Verify the **DoubleLockDoorResponse** message from the DUT.
11. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
12. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 11 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).
13. Check that all Notification messages with PropertyOperation = "Changed" contain Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token.

14. Check that all Notification messages with PropertyOperation = "Changed" have TopicExpression equal to tns1:Door/State/DoorMode.

15. Verify that the last Notification message with PropertyOperation = "Changed" contains Data.SimpleItem item with Name = «State» and Value equal to “DoubleLocked”. Check that other Notification messages contain Data.SimpleItem item with Name = «State» and Value with tdc:DoorMode type.

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

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send valid DoubleLockDoorResponse message.
- The DUT did not send CreatePullPointSubscriptionResponse message.
- The DUT did not return at least one Door at step 3.
- The DUT sent an invalid Notification messages with PropertyOperation = "Changed" (correct value for UTC time, TopicExpression and wsnt:Message, Topic = “tns1:Door/State/DoorMode”, contains Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token, contains Data.SimpleItem item with Name = «State» and Value with type equal to tdc:DoorMode) at step 11.
- The DUT sent the last Notification message with PropertyOperation = "Changed" with State value other than “DoubleLocked” at step 11.

Note: If the DUT does not return any door for step 3, skip steps 4-16, fail the test and go to the next test.

Note: If the DUT sends SOAP fault to DoubleLockDoor request then the ONVIF Client deletes Subscription Manager, starts user interaction process (see Annex A.4), and then runs the test from step 6.

Note: If in the Annex A.5 the DUT sends SOAP fault to request to change door mode, then the ONVIF Client starts user interaction process (see Annex A.4), and then runs the test from step 6.

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.

4.3.22 LOCK DOOR

Test Case ID: DOORCONTROL-3-1-31

Specification Coverage: LockDoor (ONVIF Door Control Service Specification)

Feature Under Test: LockDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify that Lock Door is accepted. To verify Door state change after Lock Door command. To verify tns1:Door/State/DoorMode property events.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Lock capability is configured and added to the DUT. The doors are closed and there are no schedules.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get a complete list of doors from the DUT (see Annex A.1).
4. ONVIF Client selects one door from the complete list of doors at step 3 with DoorInfo.Capabilities.Lock equal to true. If there is no such Door, fail the test and skip other steps.
5. ONVIF Client will invoke **GetDoorState** request (Token = [selected Door token]) to retrieve current state of the door.

6. Verify **GetDoorStateResponse** from the DUT.
7. If DoorState.DoorMode is equal to Unlocked or Blocked or DoubleLocked or Accessed then go to step 23.
8. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorMode Topic as Filter and an InitialTerminationTime of timeout1.
9. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
10. If DoorState.DoorMode is equal to LockedDown then ONVIF Client will invoke LockDownReleaseDoor request (Token = “Token1”). Otherwise, go to step 15.
11. Verify **LockDownReleaseDoorResponse** from the DUT.
12. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
13. Verify that the DUT sends a **PullMessagesResponse**. Repeat step 12 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).
14. Check that the last Notification message with PropertyOperation = "Changed" contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to Token1 and contains Data.SimpleItem item with Name = «State» and Value equals to “Locked”.
15. If DoorState.DoorMode is equal to Locked then ONVIF Client will invoke UnlockDoor request (Token = “Token1”) to move the door into state different from Locked. Otherwise, go to step 17.
16. Verify **UnlockDoorResponse** message from the DUT.
17. If DoorState.DoorMode is equal to **LockedOpen** then ONVIF Client will invoke LockOpenRelease request (Token = “Token1”) to move the door into Unlocked state.
18. Verify **LockOpenReleaseResponse** message from the DUT.
19. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
20. Verify that the DUT sends a **PullMessagesResponse**. Repeat step 19 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).
21. Check that the last Notification message with PropertyOperation = "Changed" contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to Token1 and contains Data.SimpleItem item with Name = «State» and Value differ from “Locked”.

22. ONVIF Client deletes Subscription Manager either by calling **Unsubscribe** or through a timeout and returns to the test.
23. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorMode Topic as Filter and an InitialTerminationTime of timeout1.
24. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
25. Execute Annex A.3 to catch Initialized event for selected door.
26. ONVIF Client will invoke **LockDoor** request (Token = [selected Door token]) to change door state.
27. Verify the **LockDoorResponse** message from the DUT.
28. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
29. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 28 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).
30. Check that all Notification messages with PropertyOperation = "Changed" contain Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token.
31. Check that all Notification messages with PropertyOperation = "Changed" have TopicExpression equal to tns1:Door/State/DoorMode.
32. Verify that the last Notification message with PropertyOperation = "Changed" contains Data.SimpleItem item with Name = «State» and Value equal to “Locked”. Check that other Notification messages contain Data.SimpleItem item with Name = «State» and Value with tdc:DoorMode type.
33. Verify received Notification messages (correct value for UTC time, TopicExpression and wsnt:Message).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send valid **GetDoorStateResponse** message.
- The DUT did not send valid **LockDoorResponse** message.

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not return at least one Door at step 3.
- The DUT sent an invalid Notification messages with PropertyOperation = "Changed" (correct value for UTC time, TopicExpression and wsnt:Message, Topic = “tns1:Door/State/DoorMode”, contains Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token, contains Data.SimpleItem item with Name = «State» and Value with type equal to tdc:DoorMode) at step 29.
- The DUT sent the last Notification message with PropertyOperation = "Changed" with State value other than “Locked” at step 29.

Note: If the DUT does not return any door for step 3, skip steps 4-33, fail the test and go to the next test.

Note: If the DUT sends SOAP fault to command changing the DoorState.DoorMode at step 11 or 16 or 18 or 27 then the ONVIF Client starts user interaction process (see Annex A.4), and runs the test from step 23.

Note: If the DUT sends the last notification message with door state differerent from “Locked” at step 13, then the ONVIF Client starts user interaction process (see Annex A.4), and runs the test from step 23.

Note: If the DUT sends the last notification message with door state not different from “Locked” at step 20, then the ONVIF Client starts user interaction process (see Annex A.4), and runs the test from step 23.

Note: If in the Annex A.5 the DUT sends SOAP fault to request to change door mode, then the ONVIF Client starts user interaction process (see Annex A.4), and then runs the test from step 23.

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.

4.3.23 UNLOCK DOOR

Test Case ID: DOORCONTROL-3-1-32

Specification Coverage: UnlockDoor (ONVIF Door Control Service Specification)

Feature Under Test: UnlockDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify that Unlock Door is accepted or fault is returned. To verify Door state change after Unlock Door command. To verify tns1:Door/State/DoorMode property events.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Unlock capability is configured and added to the DUT. The doors are closed and there are no schedules.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get a complete list of doors from the DUT (see Annex A.1).
4. ONVIF Client selects one door from the complete list of doors at step 3 with DoorInfo.Capabilities.Unlock equal to true and DoorInfo.Capabilities.Locked equal to true. If there is no Door with DoorInfo.Capabilities.Unlock equal to true, fail the test and skip other steps. If there is no Door with DoorInfo.Capabilities.Unlock equal to true and DoorInfo.Capabilities.Lock equal to true then ONVIF Client executes Annex A.4 to start user interaction process and then goes to step 6.
5. ONVIF Client will execute Annex A.5 to switch the selected door into Locked state.
6. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorMode Topic as Filter and an InitialTerminationTime of timeout1.
7. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
8. Execute Annex A.3 to catch Initialized event for selected door.
9. ONVIF Client will invoke **UnlockDoor** request (Token = [selected Door token]) to change door state.

10. Verify the **UnlockDoorResponse** message from the DUT.
11. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
12. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 11 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).
13. Check that all Notification messages with PropertyOperation = "Changed" contain Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token.
14. Check that all Notification messages with PropertyOperation = "Changed" have TopicExpression equal to tns1:Door/State/DoorMode.
15. Verify that the last Notification message with PropertyOperation = "Changed" contains Data.SimpleItem item with Name = «State» and Value equal to “Unlocked”. Check that other Notification messages contain Data.SimpleItem item with Name = «State» and Value with tdc:DoorMode type.
16. Verify received Notification messages (correct value for UTC time, TopicExpression and wsnt:Message).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send valid **UnlockDoorResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not return at least one Door at step 3.
- The DUT sent an invalid Notification messages with PropertyOperation = "Changed" (correct value for UTC time, TopicExpression and wsnt:Message, Topic = “tns1:Door/State/DoorMode”, contains Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token, contains Data.SimpleItem item with Name = «State» and Value with type equal to tdc:DoorMode) at step 12.
- The DUT sent the last Notification message with PropertyOperation = "Changed" with State value other than “Unlocked” at step 12.

Note: If the DUT does not return any door for step 3, skip steps 4-16, fail the test and go to the next test.

Note: If the DUT sends SOAP fault to UnlockDoor request then the ONVIF Client deletes Subscription Manager, starts user interaction process (see Annex A.4), and then runs the test from step 6.

Note: If in the Annex A.5 the DUT sends SOAP fault to request to change door mode, then the ONVIF Client starts user interaction process (see Annex A.4), and then runs the test from step 6.

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 required Notification messages are received for step 9 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.

4.3.24 LOCK OPEN DOOR

Test Case ID: DOORCONTROL-3-1-33

Specification Coverage: LockOpenDoor (ONVIF Door Control Service Specification)

Feature Under Test: LockOpenDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify that Lock Open Door is accepted. To verify Door state change after Lock Open Door command. To verify tns1:Door/State/DoorMode property events.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Lock Open capability is configured and added to the DUT. The doors are closed and there are no schedules.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get a complete list of doors from the DUT (see Annex A.1).
4. ONVIF Client selects one door from the complete list of doors at step 3 with DoorInfo.Capabilities.LockOpen equal to true and DoorInfo.Capabilities.Locked equal to true. If there is no Door with DoorInfo.Capabilities.LockOpen equal to true, fail the test and skip other steps. If there is no Door with DoorInfo.Capabilities.LockOpen equal to true and DoorInfo.Capabilities.Lock equal to true then ONVIF Client executes Annex A.4 to start user interaction process and then goes to step 6.
5. ONVIF Client will execute Annex A.5 to switch the selected door into Locked state.
6. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorMode Topic as Filter and an InitialTerminationTime of timeout1.
7. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
8. Execute Annex A.3 to catch Initialized event for selected door.
9. ONVIF Client will invoke **LockOpenDoor** request (Token = [selected Door token]) to change door state.
10. Verify the **LockOpenDoorResponse** message from the DUT.
11. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
12. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 11 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).
13. Check that all Notification messages with PropertyOperation = "Changed" contain Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token.
14. Check that all Notification messages with PropertyOperation = "Changed" have TopicExpression is equal to tns1:Door/State/DoorMode.
15. Verify that the last Notification message with PropertyOperation = "Changed" contains Data.SimpleItem item with Name = «State» and Value equal to “LockedOpen”. Check that other Notification messages contain Data.SimpleItem item with Name = «State» and Value with tdc:DoorMode type.

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

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send valid **LockOpenDoorResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not return at least one Door at step 3.
- The DUT sent an invalid Notification messages with PropertyOperation = "Changed" (correct value for UTC time, TopicExpression and wsnt:Message, Topic = “tns1:Door/State/DoorMode”, contains Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token, contains Data.SimpleItem item with Name = «State» and Value with type equal to tdc:DoorMode) at step 12.
- The DUT sent the last Notification message with PropertyOperation = "Changed" with State value other than “LockedOpen” at step 12.

Note: If the DUT does not return any door for step 3, skip steps 4-16, fail the test and go to the next test.

Note: If the DUT sends SOAP fault to LockOpenDoor request then the ONVIF Client deletes Subscription Manager, starts user interaction process (see Annex A.4), and then runs the test from step 6.

Note: If in the Annex A.5 the DUT sends SOAP fault to request to change door mode, then the ONVIF Client starts user interaction process (see Annex A.4), and then runs the test from step 6.

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.

4.3.25 LOCK OPEN RELEASE DOOR

Test Case ID: DOORCONTROL-3-1-34

Specification Coverage: LockOpenReleaseDoor (ONVIF Door Control Service Specification)

Feature Under Test: LockOpenReleaseDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify that Lock Open Release Door is accepted. To verify Door state change after Lock Open Release Door command. To verify tns1:Door/State/DoorMode property events.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Lock Open capability is configured and added to the DUT. The doors are closed and there are no schedules. Test Configuration: ONVIF Client and DUT

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get a complete list of doors from the DUT (see Annex A.1).
4. ONVIF Client selects one door from the complete list of doors at step 3 with DoorInfo.Capabilities.LockOpen equal to true and DoorInfo.Capabilities.Locked equal to true. If there is no Door with DoorInfo.Capabilities.LockOpen equal to true, fail the test and skip other steps. If there is no Door with DoorInfo.Capabilities.LockOpen equal to true and DoorInfo.Capabilities.Lock equal to true then ONVIF Client executes Annex A.4 to start user interaction process and then goes to step 14.
5. ONVIF Client will invoke **GetDoorState** request (Token = “Token1”, where Token1 is selected door) to retrieve current state of the door.
6. Verify **GetDoorStateResponse** from the DUT. If DoorState.DoorMode is equal to LockedOpen then skip steps 7-15 and go to step 16.
7. ONVIF Client will execute Annex A.5 to switch the selected door into Locked state.

8. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorMode Topic as Filter and an InitialTerminationTime of timeout1.
9. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
10. ONVIF Client will invoke **LockOpenDoor** request (Token = [selected Door token]) to move the door in LockedOpen state.
11. Verify **LockOpenDoorResponse** message from the DUT.
12. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
13. Verify that the DUT sends a **PullMessagesResponse**. Repeat step 12 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).
14. Check that the last Notification message with PropertyOperation = "Changed" contains Source.SimpleItem item with Name = "DoorToken" and Value equal to selected door tocken and contains Data.SimpleItem item with Name = «State» and Value equal to “LockedOpen”.
15. ONVIF Client deletes Subscription Manager either by calling **Unsubscribe** or through a timeout.
16. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorMode Topic as Filter and an InitialTerminationTime of timeout1.
17. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
18. Execute Annex A.3 to catch Initialized event for selected door. State of the Door received in Annex A.3 will be assumed as initial State of this Door.
19. Check that initial state of the Door equals to “LockedOpen”.
20. ONVIF Client will invoke **LockOpenReleaseDoor** request (Token = [selected Door token]) to change door state.
21. Verify the **LockOpenDoorReleaseResponse** message from the DUT.
22. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
23. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 22 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).
24. Check that all Notification messages with PropertyOperation = "Changed" contain Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token.

25. Check that all Notification messages with PropertyOperation = "Changed" have TopicExpression is equal to tns1:Door/State/DoorMode.

26. Virify that last Notification message with PropertyOperation = "Changed" contains Data.SimpleItem item with Name = «State» and Value other than “LockedOpen”. Check that other Notification messages contain Data.SimpleItem item with Name = «State» and Value with tdc:DoorMode type.

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

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send valid **LockOpenReleaseDoorResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not return at least one Door at step 3.
- The DUT sent the last Notification message with PropertyOperation = "Changed" with State value not equal to “LockedOpen” at step 13.
- The DUT sent Notification message for selected door token with PropertyOperation = "Initialized" with State value not equal to “LockedOpen” at step 18.
- The DUT sent an invalid Notification messages with PropertyOperation = "Changed" (correct value for UTC time, TopicExpression and wsnt:Message, Topic = “tns1:Door/State/DoorMode”, contains Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token, contains Data.SimpleItem item with Name = «State» and Value with type equal to tdc:DoorMode) at step 23.
- The DUT sent the last Notification message with PropertyOperation = "Changed" with State value equal to “LockedOpen” at step 23.

Note: If the DUT does not return any door for step 3, skip steps 4-27, fail the test and go to the next test.

Note: If the DUT sends SOAP fault to LockOpenDoor request or to LockOpenRealiseDoorRequest then the ONVIF Client deletes Subscription Manager, starts user interaction process (see Annex A.4), and then runs the test from step 16.

Note: If in the Annex A.5 the DUT sends SOAP fault to request to change door mode, then the ONVIF Client starts user interaction process (see Annex A.4), and then runs the test from step 16.

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.

4.3.26 LOCK DOWN DOOR

Test Case ID: DOORCONTROL-3-1-35

Specification Coverage: LockDownDoor (ONVIF Door Control Service Specification)

Feature Under Test: LockDownDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify that Lock Down Door is accepted. To verify Door state change after Lock Down Door command. To verify tns1:Door/State/DoorMode property events.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Lock Down capability is configured and added to the DUT. The doors are closed and there are no schedules.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get a complete list of doors from the DUT (see Annex A.1).
4. ONVIF Client selects one door from the complete list of doors at step 3 with DoorInfo.Capabilities.LockDown equal to true and DoorInfo.Capabilities.Locked equal to

true. If there is no Door with DoorInfo.Capabilities.LockDown equal to true, fail the test and skip other steps. If there is no Door with DoorInfo.Capabilities.LockDown equal to true and DoorInfo.Capabilities.Lock equal to true then ONVIF Client executes Annex A.4 to start user interaction process and then goes to step 6.

5. ONVIF Client will execute Annex A.5 to switch the selected door into Locked state.
6. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorMode Topic as Filter and an InitialTerminationTime of timeout1.
7. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
8. Execute Annex A.3 to catch Initialized event for selected door.
9. ONVIF Client will invoke **LockDownDoor** request (Token = [selected Door token]) to change door state.
10. Verify the **LockDownDoorResponse** message from the DUT.
11. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
12. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 11 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).
13. Check that all Notification messages with PropertyOperation = "Changed" contain Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token.
14. Check that all Notification messages with PropertyOperation = "Changed" have TopicExpression is equal to tns1:Door/State/DoorMode.
15. Verify that the last Notification message with PropertyOperation = "Changed" contains Data.SimpleItem item with Name = «State» and Value equal to “LockedDown”. Check that other Notification messages contain Data.SimpleItem item with Name = «State» and Value with tdc:DoorMode type.
16. Verify received Notification messages (correct value for UTC time, TopicExpression and wsnt:Message).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send valid **LockDownDoorResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not return at least one Door at step 3.
- The DUT sent an invalid Notification messages with PropertyOperation = "Changed" (correct value for UTC time, TopicExpression and wsnt:Message, Topic = “tns1:Door/State/DoorMode”, contains Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token, contains Data.SimpleItem item with Name = «State» and Value with type equal to tdc:DoorMode) at step 12.
- The DUT sent the last Notification message with PropertyOperation = "Changed" with State value other than “LockedDown” at step 12.

Note: If the DUT does not return any door for step 3, skip steps 4-16, fail the test and go to the next test.

Note: If the DUT sends SOAP fault to LockDownDoor request, then the ONVIF Client deletes Subscription Manager, starts user interaction process (see Annex A.4), and then runs the test from step 6.

Note: If in the Annex A.5 the DUT sends SOAP fault to request to change door mode, then the ONVIF Client starts user interaction process (see Annex A.4), and then runs the test from step 6.

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.

4.3.27 LOCK DOWN RELEASE DOOR

Test Case ID: DOORCONTROL-3-1-36

Specification Coverage: LockDownReleaseDoor (ONVIF Door Control Service Specification)

Feature Under Test: LockDownReleaseDoor

WSDL Reference: doorcontrol.wsdl

Test Purpose: To verify that Lock Down Release Door is accepted. To verify Door state change after Lock Down Release Door command. To verify tns1:Door/State/DoorMode property events.

Pre-Requisite: Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Lock Down capability is configured and added to the DUT. The doors are closed and there are no schedules.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get a complete list of doors from the DUT (see Annex A.1).
4. ONVIF Client selects one door from the complete list of doors at step 3 with DoorInfo.Capabilities.LockDown equal to true and DoorInfo.Capabilities.Locked equal to true. If there is no Door with DoorInfo.Capabilities.LockDown equal to true, fail the test and skip other steps. If there is no Door with DoorInfo.Capabilities.LockDown equal to true and DoorInfo.Capabilities.Lock equal to true then ONVIF Client executes Annex A.4 to start user interaction process and then goes to step 14.
5. ONVIF Client will invoke **GetDoorState** request (Token = “Token1”, where Token1 is selected door) to retrieve current state of the door.
6. Verify **GetDoorStateResponse** from the DUT. If DoorState.DoorMode is equal to LockedDown then skip steps 7-15 and go to step 16.
7. ONVIF Client will execute Annex A.5 to switch the selected door into Locked state.
8. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorMode Topic as Filter and an InitialTerminationTime of timeout1.
9. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
10. ONVIF Client will invoke **LockDownDoor** request (Token = [selected Door token]) to move the door in LockedDown state.
11. Verify **LockDownDoorResponse** message from the DUT.

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

13. Verify that the DUT sends a **PullMessagesResponse**. Repeat step 12 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).

14. Verify that the last Notification message with PropertyOperation = "Changed" contains Source.SimpleItem item with Name = "DoorToken" and Value equal to selected door tocken and contains Data.SimpleItem item with Name = «State» and Value equal to “LockedDown”.

15. ONVIF Client deletes Subscription Manager either by calling **Unsubscribe** or through a timeout.

16. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorMode Topic as Filter and an InitialTerminationTime of timeout1.

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

18. Execute Annex A.3 to catch Initialized event for selected door. State of the Door received in Annex A.3 will be assumed as initial State of this Door.

19. Check that initial state of the Door equals to “LockedDown”.

20. ONVIF Client will invoke **LockDownReleaseDoor** request (Token = [selected Door token]) to change door state.

21. Verify the **LockDownReleaseDoorResponse** message from the DUT.

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

23. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 22 until timeout has expired (Operation Delay option from ONVIF Device Test Tool will be used).

24. Check that all Notification messages with PropertyOperation = "Changed" contain Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token.

25. Check that all Notification messages with PropertyOperation = "Changed" have TopicExpression is equal to tns1:Door/State/DoorMode.

26. Verify that the last Notification message with PropertyOperation = "Changed" contains Data.SimpleItem item with Name = «State» and Value other than “LockedDown”. Check that other Notification messages contain Data.SimpleItem item with Name = «State» and Value with tdc:DoorMode type.

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

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send valid **LockDownReleaseDoorResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not return at least one Door at step 3.
- The DUT sent the last Notification message with PropertyOperation = "Changed" with State value not equal to “LockedDown” at step 13.
- The DUT sent Notification message for selected door token with PropertyOperation = "Initialized" with State value not equal to “LockedDown” at step 18.
- The DUT sent an invalid Notification messages with PropertyOperation = "Changed" (correct value for UTC time, TopicExpression and wsnt:Message, Topic = “tns1:Door/State/DoorMode”, contains Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token, contains Data.SimpleItem item with Name = «State» and Value with type equal to tdc:DoorMode) at step 23.
- The DUT sent the last Notification message with PropertyOperation = "Changed" with State value equal to “LockedDown” at step 23.

Note: If the DUT does not return any door for step 3, skip steps 4-27, fail the test and go to the next

Note: If the DUT sends SOAP fault to LockDownDoor request or to LockDownReleaseDoorRequest then the ONVIF Client deletes Subscription Manager, starts user interaction process (see Annex A.4), and then runs the test from step 16.

Note: If in the Annex A.5 the DUT sends SOAP fault to request to change door mode, then the ONVIF Client starts user interaction process (see Annex A.4), and then runs the test from step 16.

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.

4.4 Consistency

4.4.1 GET ACCESS POINT INFO LIST AND GET DOOR INFO LIST CONSISTENCY

Test Case ID: DOORCONTROL-5-1-1

Specification Coverage: GetDoorInfoList (ONVIF Door Control Service Specification), GetAccessPointInfoList (ONVIF Access Control Service Specification)

Feature Under Test: GetDoorInfoList, GetAccessPointInfoList

WSDL Reference: accesscontrol.wsdl, doorcontrol.wsdl

Test Purpose: To verify that Door Info List contains all Doors from Access Point Info List.

Pre-Requisite: Access Control Service was received from the DUT. Door Control Service address 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. Profile C scope or Profile A scope.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get a complete list of doors from the DUT (see Annex A.1)
4. Get a complete list of access points from the DUT (see Annex A.2)
5. Verify that the complete list of doors contains all Doors included in the complete list of access points in Entity element in case Type is skipped or equal to tdc:Door.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT returned at least one Door listed in Entity element of AccessPointInfo listed in the complete list of access points in case Type is skipped or equal to tdc:Door that was not included in the complete list of doors.
- The DUT did not return at least one Door at step 3.
- The DUT did not return at least one Access Point at step 4.

4.5 Property Events

4.5.1 DOOR CONTROL – DOOR MODE EVENT

Test Case ID: DOORCONTROL-6-1-1

Specification Coverage: tns1:Door/State/DoorMode (ONVIF Door Control Service Specification), GetDoorState (ONVIF Door Control Service Specification), Properties (ONVIF Core Specification)

Feature Under Test: GetDoorState

WSDL Reference: event.wsdl, doorcontrol.wsdl

Test Purpose: To verify tns1:Door/State/DoorMode event generation after subscription and to verify tns1:Door/State/DoorMode event format.

Pre-Requisite: Event Service was received from the DUT. Device supports Pull-Point Notification feature. Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door 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 doors 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:Door/State/DoorMode. 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 a Property event (MessageDescription.IsProperty = "true").
8. Check that this event contains Source.SimpleItemDescription item with Name = "DoorToken" and Type = "pt:ReferenceToken".
9. Check that this event contains Data.SimpleItemDescription item with Name = «State» and Type = "tdc:DoorMode".
10. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorMode Topic as Filter and an InitialTerminationTime of timeout1.
11. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
12. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
13. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 12 until Notification for all Doors is received.
14. Verify the received Notify messages (correct value for UTC time, TopicExpression and wsnt:Message).
15. Verify that TopicExpression is equal to tns1:Door/State/DoorMode for all received Notify messages.
16. Verify that each notification contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to one of existing Door Tokens (e.g. complete list of doors contains Door with the same token). Verify that there are Notification messages for each Door.
17. Verify that each notification contains Data.SimpleItem item with Name = «State» and Value with type is equal to tdc:DoorMode.
18. Verify that Notify PropertyOperation = "Initialized".
19. ONVIF Client will invoke **GetDoorState** request for each Door with corresponding tokens.
20. Verify the **GetDoorStateResponse** messages from the DUT. Verify that Data.SimpleItem item with Name = «State» from Notification message has the same value with DoorMode elements from corresponding **GetDoorStateResponse** messages for each Door.

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:Door/State/DoorMode at least for one Door.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid DoorToken or State values, PropertyOperation is not equal to "Initialized").
- The DUT did not return at least one Door at step 3.
- The DUT did not return Topic tns1:Door/State/DoorMode 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 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 13 will wait for Notification messages until notification for all Doors is received or Operation Delay has expired. Notification messages for all Doors are assumed as received, if the number of Notification messages is equal to the number of Doors.

Note: If the DUT does not return any door for step 3, skip steps 4-20, 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 DOOR CONTROL – DOOR PHYSICAL STATE EVENT

Test Case ID: DOORCONTROL-6-1-2

Specification Coverage: tns1:Door/State/DoorPhysicalState (ONVIF Door Control Service Specification), GetDoorState (ONVIF Door Control Service Specification), Properties (ONVIF Core Specification)

Feature Under Test: GetDoorState

WSDL Reference: event.wsdl, doorcontrol.wsdl

Test Purpose: To verify tns1:Door/State/DoorPhysicalState event generation after subscription and to verify tns1:Door/State/DoorPhysicalState event format.

Pre-Requisite: Event Service was received from the DUT. Device supports Pull-Point Notification feature. Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Door Monitor capability 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 doors from the DUT (see Annex A.1).
4. Check that there is at least one Door with Capabilities.DoorMonitor = “true”. Otherwise 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:Door/State/DoorPhysicalState. 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 = "DoorToken" and Type = "pt:ReferenceToken".
10. Check that this event contains Data.SimpleItemDescription item with Name = "State" and Type = "tdc:DoorPhysicalState".
11. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorPhysicalState Topic as Filter and an InitialTerminationTime of timeout1.
12. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
13. ONVIF Client will invoke **PullMessages** command 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 Doors with Capabilities.DoorMonitor = “true” is received.
15. Verify received Notify messages (correct value for UTC time, TopicExpression and wsnt:Message).
16. Verify that TopicExpression is equal to tns1:Door/State/DoorPhysicalState for all received Notify messages.
17. Verify that each notification contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to one of existing Door Tokens Capabilities.DoorMonitor = “true” (e.g. complete list of doors contains Door with the same token). Verify that there are Notification messages for each Door with Capabilities.DoorMonitor = “true”.
18. Verify that each notification contains Data.SimpleItem item with Name = "State" and Value with type is equal to tdc:DoorPhysicalState.
19. Verify that Notify PropertyOperation = "Initialized".
20. ONVIF Client will invoke GetDoorState request for each Door with corresponding tokens.
21. Verify the **GetDoorStateResponse** messages from the DUT. Verify that Data.SimpleItem item with Name = "State" from Notification message has the same value with DoorPhysicalState elements from corresponding **GetDoorStateResponse** messages for each Door with Capabilities.DoorMonitor = “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 **PullMessagesResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a Notification message that contains a property event tns1:Door/State/DoorPhysicalState at least for one Door with Capabilities.DoorMonitor = “true”.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid DoorToken or State values, PropertyOperation is not equal to "Initialized").
- The DUT did not return at least one Door at step 3.
- The DUT did not return Topic tns1:Door/State/DoorPhysicalState 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 Doors with Capabilities.DoorMonitor = “true” is received or Operation Delay has expired. Notification messages for all Doors with Capabilities.DoorMonitor = “true” are assumed as received, if the number of Notification messages is equal to the number of Doors with Capabilities.DoorMonitor = “true”.

Note: If the DUT does not return any door 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.

4.5.3 DOOR CONTROL – DOOR PHYSICAL STATE EVENT STATE CHANGE

Test Case ID: DOORCONTROL-6-1-3

Specification Coverage: tns1:Door/State/DoorPhysicalState (ONVIF Door Control Service Specification)

Feature Under Test: None

WSDL Reference: event.wsdl, doorcontrol.wsdl

Test Purpose: To verify tns1:Door/State/DoorPhysicalState event generation after property was changed and to verify tns1:Door/State/DoorPhysicalState event format.

Pre-Requisite: Event Service was received from the DUT. Device supports Pull-Point Notification feature. Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Door Monitor capability 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 doors from the DUT (see Annex A.1).
4. Check that there is at least one Door with Capabilities.DoorMonitor = “true”. Otherwise fail the test and skip other steps.
5. ONVIF Client will select one first Door (token = Token1) with Capabilities.DoorMonitor = “true”.
6. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorPhysicalState Topic as Filter and an InitialTerminationTime of timeout1.
7. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
8. Test Operator will invoke change of DoorPhysicalState property for Door with token = Token1.

9. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
10. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 9 until Notification with PropertyOperation = "Changed" for Door with Capabilities.DoorMonitor = “true” is received.
11. Verify received Notify messages (correct value for UTC time, TopicExpression and wsnt:Message).
12. Verify that TopicExpression is equal to tns1:Door/State/DoorPhysicalState for received Notify message.
13. Verify that notification contains Source.SimpleItem item with Name = "DoorToken" and Value = “Token1”.
14. Verify that notification contains Data.SimpleItem item with Name = "State" and Value with type is equal to tdc:DoorPhysicalState.

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:Door/State/DoorPhysicalState with valid DoorToken.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid DoorToken value).
- The DUT did not return at least one Door at step 3.

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 10 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

Note: If the DUT does not return any door for step 3, skip steps 4-14, 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.

4.5.4 DOOR CONTROL – DOUBLE LOCK PHYSICAL STATE EVENT

Test Case ID: DOORCONTROL-6-1-4

Specification Coverage: tns1:Door/State/DoubleLockPhysicalState (ONVIF Door Control Service Specification), GetDoorState (ONVIF Door Control Service Specification), Properties (ONVIF Core Specification)

Feature Under Test: GetDoorState

WSDL Reference: event.wsdl, doorcontrol.wsdl

Test Purpose: To verify tns1:Door/State/DoubleLockPhysicalState event generation after subscription and to verify tns1:Door/State/DoubleLockPhysicalState event format.

Pre-Requisite: Event Service was received from the DUT. Device supports Pull-Point Notification feature. Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Double Lock Monitor capability 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 doors from the DUT (see Annex A.1).
4. Check that there is at least one Door with Capabilities.DoubleLockMonitor = “true”. Otherwise 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:Door/State/DoubleLockPhysicalState. 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 = "DoorToken" and Type = "pt:ReferenceToken".
10. Check that this event contains Data.SimpleItemDescription item with Name = "State" and Type = "tdc:LockPhysicalState".
11. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoubleLockPhysicalState Topic as Filter and an InitialTerminationTime of timeout1.
12. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
13. ONVIF Client will invoke PullMessages command 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 Doors with Capabilities.DoorMonitor = “true” is received.
15. Verify received Notify messages (correct value for UTC time, TopicExpression and wsnt:Message).
16. Verify that TopicExpression is equal to tns1:Door/State/DoubleLockPhysicalState for all received Notify messages.
17. Verify that each notification contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to one of existing Door Tokens Capabilities.DoubleLockMonitor = “true” (e.g. complete list of doors contains Door with the same token). Verify that there are Notification messages for each Door with Capabilities.DoubleLockMonitor = “true”.

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

19. Verify that Notify PropertyOperation = "Initialized".

20. ONVIF Client will invoke GetDoorState request for each Door with corresponding tokens.

21. Verify the **GetDoorStateResponse** messages from the DUT. Verify that Data.SimpleItem item with Name = "State" from Notification message has the same value with DoubleLockPhysicalState elements from corresponding **GetDoorStateResponse** messages for each Door with Capabilities.DoubleLockMonitor = “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 **PullMessagesResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a Notification message that contains a property event tns1:Door/State/DoubleLockPhysicalState at least for one Door with Capabilities.DoubleLockMonitor = “true”.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid DoorToken or State values, PropertyOperation is not equal to "Initialized").
- The DUT did not return at least one Door at step 3.
- The DUT did not return Topic tns1:Door/State/DoubleLockPhysicalState 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 Doors with Capabilities.DoubleLockMonitor = “true” is received or Operation Delay has expired. Notification messages for all Doors with Capabilities.DoubleLockMonitor = “true” are assumed

as received, if the number of Notification messages is equal to the number of Doors with Capabilities.DoubleLockMonitor = “true”.

Note: If the DUT does not return any door 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.

4.5.5 DOOR CONTROL – DOUBLE LOCK PHYSICAL STATE EVENT STATE CHANGE

Test Case ID: DOORCONTROL-6-1-5

Specification Coverage: tns1:Door/State/DoubleLockPhysicalState (ONVIF Door Control Service Specification)

Feature Under Test: None

WSDL Reference: event.wsdl, doorcontrol.wsdl

Test Purpose: To verify tns1:Door/State/DoubleLockPhysicalState event generation after property was changed and to verify tns1:Door/State/DoubleLockPhysicalState event format.

Pre-Requisite: Event Service was received from the DUT. Device supports Pull-Point Notification feature. Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Double Lock Monitor capability 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 doors from the DUT (see Annex A.1).
4. Check that there is at least one Door with Capabilities.DoubleLockMonitor = “true”. Otherwise fail the test and skip other steps.
5. ONVIF Client will select one first Door (token = Token1) with Capabilities.DoubleLockMonitor = “true”.
6. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoubleLockPhysicalState Topic as Filter and an InitialTerminationTime of timeout1.
7. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
8. Test Operator will invoke change of DoubleLockPhysicalState property for Door with token = Token1.
9. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
10. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 9 until Notification with PropertyOperation = "Changed" for Door with Capabilities.DoubleLockMonitor = “true” is received.
11. Verify received Notify messages (correct value for UTC time, TopicExpression and wsnt:Message).
12. Verify that TopicExpression is equal to tns1:Door/State/DoubleLockPhysicalState for received Notify message.
13. Verify that notification contains Source.SimpleItem item with Name = "DoorToken" and Value = “Token1”.
14. Verify that notification contains Data.SimpleItem item with Name = "State" and Value with type is equal to tdc:DoubleLockPhysicalState.

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:Door/State/DoubleLockPhysicalState with valid DoorToken.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid DoorToken value).
- The DUT did not return at least one Door at step 3.

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 10 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

Note: If the DUT does not return any door for step 3, skip steps 4-14, 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.

4.5.6 DOOR CONTROL – LOCK PHYSICAL STATE EVENT

Test Case ID: DOORCONTROL-6-1-6

Specification Coverage: tns1:Door/State/LockPhysicalState (ONVIF Door Control Service Specification), GetDoorState (ONVIF Door Control Service Specification), Properties (ONVIF Core Specification)

Feature Under Test: GetDoorState

WSDL Reference: event.wsdl, doorcontrol.wsdl

Test Purpose: To verify tns1:Door/State/LockPhysicalState event generation after subscription and to verify tns1:Door/State/LockPhysicalState event format.

Pre-Requisite: Event Service was received from the DUT. Device supports Pull-Point Notification feature. Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Lock Monitor capability 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 doors from the DUT (see Annex A.1).
4. Check that there is at least one Door with Capabilities.LockMonitor = “true”. Otherwise 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:Door/State/LockPhysicalState. 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 = "DoorToken" and Type = "pt:ReferenceToken".
10. Check that this event contains Data.SimpleItemDescription item with Name = "State" and Type = "tdc:LockPhysicalState".
11. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/LockPhysicalState Topic as Filter and an InitialTerminationTime of timeout1.
12. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
13. ONVIF Client will invoke **PullMessages** command 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 Doors with Capabilities.DoorMonitor = “true” is received.

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

16. Verify that TopicExpression is equal to tns1:Door/State/LockPhysicalState for all received Notify messages.

17. Verify that each notification contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to one of existing Door Tokens Capabilities.LockMonitor = “true” (e.g. complete list of doors contains Door with the same token). Verify that there are Notification messages for each Door with Capabilities.LockMonitor = “true”.

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

19. Verify that Notify PropertyOperation = "Initialized".

20. ONVIF Client will invoke **GetDoorState** request for each Door with corresponding tokens.

21. Verify the **GetDoorStateResponse** messages from the DUT. Verify that Data.SimpleItem item with Name = "State" from Notification message has the same value with LockPhysicalState elements from corresponding **GetDoorStateResponse** messages for each Door with Capabilities.LockMonitor = “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 **PullMessagesResponse** message.
- The DUT did not send valid **SubscriptionReference**.
- The DUT did not send a Notification message that contains a property event tns1:Door/State/LockPhysicalState at least for one Door with Capabilities.LockMonitor = “true”.

- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid DoorToken or State values, PropertyOperation is not equal to "Initialized").
- The DUT did not return at least one Door at step 3.
- The DUT did not return Topic tns1:Door/State/LockPhysicalState 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 Doors with Capabilities.LockMonitor = “true” is received or Operation Delay has expired. Notification messages for all Doors with Capabilities.LockMonitor = “true” are assumed as received, if the number of Notification messages is equal to the number of Doors with Capabilities.LockMonitor = “true”.

Note: If the DUT does not return any door 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.

4.5.7 DOOR CONTROL – LOCK PHYSICAL STATE EVENT STATE CHANGE

Test Case ID: DOORCONTROL-6-1-7

Specification Coverage: tns1:Door/State/LockPhysicalState (ONVIF Door Control Service Specification)

Feature Under Test: None

WSDL Reference: event.wsdl, doorcontrol.wsdl

Test Purpose: To verify tns1:Door/State/LockPhysicalState event generation after property was changed and to verify tns1:Door/State/LockPhysicalState event format.

Pre-Requisite: Event Service was received from the DUT. Device supports Pull-Point Notification feature. Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Lock Monitor capability 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 doors from the DUT (see Annex A.1).
4. Check that there is at least one Door with Capabilities.LockMonitor = “true”. Otherwise, fail the test and skip other steps.
5. ONVIF Client will select one first Door (token = Token1) with Capabilities.LockMonitor = “true”.
6. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/LockPhysicalState Topic as Filter and an InitialTerminationTime of timeout1.
7. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
8. Test Operator will invoke change of LockPhysicalState property for Door with token = Token1.
9. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
10. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 9 until Notification with PropertyOperation = "Changed" for Door with Capabilities.LockMonitor = “true” is received.
11. Verify received Notify messages (correct value for UTC time, TopicExpression and wsnt:Message).
12. Verify that TopicExpression is equal to tns1:Door/State/LockPhysicalState for received Notify message.
13. Verify that notification contains Source.SimpleItem item with Name = "DoorToken" and Value = “Token1”.

14. Verify that notification contains Data.SimpleItem item with Name = "State" and Value with type is equal to tdc:LockPhysicalState.

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:Door/State/LockPhysicalState with valid DoorToken.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid DoorToken value).
- The DUT did not return at least one Door at step 3.

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 10 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

Note: If the DUT does not return any door for step 3, skip steps 4-14, 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.

4.5.8 DOOR CONTROL – DOOR TAMPER EVENT

Test Case ID: DOORCONTROL-6-1-8

Specification Coverage: tns1:Door/State/DoorTamper (ONVIF Door Control Service Specification), GetDoorState (ONVIF Door Control Service Specification), Properties (ONVIF Core Specification)

Feature Under Test: GetDoorState

WSDL Reference: event.wsdl, doorcontrol.wsdl

Test Purpose: To verify tns1:Door/State/DoorTamper event generation after subscription and to verify tns1:Door/State/DoorTamper event format.

Pre-Requisite: Event Service was received from the DUT. Device supports Pull-Point Notification feature. Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Tamper capability 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 doors from the DUT (see Annex A.1).
4. Check that there is at least one Door with Capabilities.Tamper = “true”. Otherwise 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:Door/State/DoorTamper. 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 = "DoorToken" and Type = "pt:ReferenceToken".
10. Check that this event contains Data.SimpleItemDescription item with Name = "State" and Type = "tdc:DoorTamperState".
11. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorTamper Topic as Filter and an InitialTerminationTime of timeout1.
12. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
13. ONVIF Client will invoke **PullMessages** command 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 Doors with Capabilities.DoorMonitor = “true” is received.
15. Verify received Notify messages (correct value for UTC time, TopicExpression and wsnt:Message).
16. Verify that TopicExpression is equal to tns1:Door/State/DoorTamper for all received Notify messages.
17. Verify that each notification contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to one of existing Door Tokens Capabilities.Tamper = “true” (e.g. complete list of doors contains Door with the same token). Verify that there are Notification messages for each Door with Capabilities.Tamper = “true”.
18. Verify that each notification contains Data.SimpleItem item with Name = "State" and Value with type is equal to tdc:DoorTamperState.
19. Verify that Notify PropertyOperation = "Initialized".
20. ONVIF Client will invoke **GetDoorState** request for each Door with corresponding tokens.
21. Verify the **GetDoorStateResponse** messages from the DUT. Verify that Data.SimpleItem item with Name = "State" from Notification message has the same value with Tamper elements from corresponding **GetDoorStateResponse** messages for each Door with Capabilities.Tamper = “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 **PullMessagesResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a Notification message that contains a property event tns1:Door/State/DoorTamper at least for one Door with Capabilities.Tamper = “true”.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid DoorToken or State values, PropertyOperation is not equal to "Initialized").
- The DUT did not return at least one Door at step 3.
- The DUT did not return Topic tns1:Door/State/DoorTamper 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 Doors with Capabilities.Tamper = “true” is received or Operation Delay has expired. Notification messages for all Doors with Capabilities.Tamper = “true” are assumed as received, if the number of Notification messages is equal to the number of Doors with Capabilities.Tamper = “true”.

Note: If the DUT does not return any door 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.

4.5.9 DOOR CONTROL – DOOR TAMPER EVENT STATE CHANGE

Test Case ID: DOORCONTROL-6-1-9

Specification Coverage: tns1:Door/State/DoorTamper (ONVIF Door Control Service Specification)

Feature Under Test: None

WSDL Reference: event.wsdl, doorcontrol.wsdl

Test Purpose: To verify tns1:Door/State/DoorTamper event generation after property was changed and to verify tns1:Door/State/DoorTamper event format.

Pre-Requisite: Event Service was received from the DUT. Device supports Pull-Point Notification feature. Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Tamper capability 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 doors from the DUT (see Annex A.1).
4. Check that there is at least one Door with Capabilities.Tamper = “true”. Otherwise fail the test and skip other steps.
5. ONVIF Client will select one first Door (token = Token1) with Capabilities.Tamper = “true”.
6. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorTamper Topic as Filter and an InitialTerminationTime of timeout1.
7. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
8. Test Operator will invoke change of DoorTamper property for Door with token = Token1.
9. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
10. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 9 until Notification with PropertyOperation = "Changed" for Door with Capabilities.Tamper = “true” is received.

11. Verify received Notify messages (correct value for UTC time, TopicExpression and wsnt:Message).
12. Verify that TopicExpression is equal to tns1:Door/State/DoorTamper for received Notify message.
13. Verify that notification contains Source.SimpleItem item with Name = "DoorToken" and Value = “Token1”.
14. Verify that notification contains Data.SimpleItem item with Name = "State" and Value with type is equal to tdc:DoorTamperState.

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:Door/State/DoorTamper with valid DoorToken.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid DoorToken value).
- The DUT did not return at least one Door at step 3.

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 10 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

Note: If the DUT does not return any door for step 3, skip steps 4-14, 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.

4.5.10 DOOR CONTROL – DOOR ALARM EVENT

Test Case ID: DOORCONTROL-6-1-10

Specification Coverage: tns1:Door/State/DoorAlarm (ONVIF Door Control Service Specification), GetDoorState (ONVIF Door Control Service Specification), Properties (ONVIF Core Specification)

Feature Under Test: GetDoorState

WSDL Reference: event.wsdl, doorcontrol.wsdl

Test Purpose: To verify tns1:Door/State/DoorAlarm event generation after subscription and to verify tns1:Door/State/DoorAlarm event format.

Pre-Requisite: Event Service was received from the DUT. Device supports Pull-Point Notification feature. Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Alarm capability 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 doors from the DUT (see Annex A.1).
4. Check that there is at least one Door with Capabilities.Alarm = “true”. Otherwise 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:Door/State/DoorAlarm. 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 = "DoorToken" and Type = "pt:ReferenceToken".
10. Check that this event contains Data.SimpleItemDescription item with Name = "State" and Type = "tdc:DoorAlarmState".
11. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorAlarm Topic as Filter and an InitialTerminationTime of timeout1.
12. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
13. ONVIF Client will invoke **PullMessages** command 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 Doors with Capabilities.DoorMonitor = “true” is received.
15. Verify received Notify messages (correct value for UTC time, TopicExpression and wsnt:Message).
16. Verify that TopicExpression is equal to tns1:Door/State/DoorAlarm for all received Notify messages.
17. Verify that each notification contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to one of existing Door Tokens Capabilities.Alarm = “true” (e.g. complete list of doors contains Door with the same token). Verify that there are Notification messages for each Door with Capabilities.Alarm = “true”.
18. Verify that each notification contains Data.SimpleItem item with Name = "State" and Value with type is equal to tdc:DoorAlarmState.
19. Verify that Notify PropertyOperation = "Initialized".
20. ONVIF Client will invoke **GetDoorState** request for each Door with corresponding tokens.
21. Verify the **GetDoorStateResponse** messages from the DUT. Verify that Data.SimpleItem item with Name = "State" from Notification message has the same value with Alarm

elements from corresponding **GetDoorStateResponse** messages for each Door with Capabilities.Alarm = “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 **PullMessagesResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a Notification message that contains a property event tns1:Door/State/DoorAlarm at least for one Door with Capabilities.Alarm = “true”.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid DoorToken or State values, PropertyOperation is not equal to "Initialized").
- The DUT did not return at least one Door at step 3.
- The DUT did not return Topic tns1:Door/State/DoorAlarm 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 Doors with Capabilities.Alarm = “true” is received or Operation Delay has expired. Notification messages for all Doors with Capabilities.Alarm = “true” are assumed as received, if the number of Notification messages is equal to the number of Doors with Capabilities.Alarm = “true”.

Note: If the DUT does not return any door 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.

4.5.11 DOOR CONTROL – DOOR ALARM EVENT STATE CHANGE

Test Case ID: DOORCONTROL-6-1-11

Specification Coverage: tns1:Door/State/DoorAlarm (ONVIF Door Control Service Specification)

Feature Under Test: None

WSDL Reference: event.wsdl, doorcontrol.wsdl

Test Purpose: To verify tns1:Door/State/DoorAlarm event generation after property was changed and to verify tns1:Door/State/DoorAlarm event format.

Pre-Requisite: Event Service was received from the DUT. Device supports Pull-Point Notification feature. Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Alarm capability 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 doors from the DUT (see Annex A.1).
4. Check that there is at least one Door with Capabilities.Alarm = “true”. Otherwise fail the test and skip other steps.
5. ONVIF Client will select one first Door (token = Token1) with Capabilities.Alarm = “true”.
6. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorAlarm Topic as Filter and an InitialTerminationTime of timeout1.
7. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.

8. Test Operator will invoke change of DoorAlarm property for Door with token = Token1.
9. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
10. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 9 until Notification with PropertyOperation = "Changed" for Door with Capabilities.Alarm = “true” is received.
11. Verify received Notify messages (correct value for UTC time, TopicExpression and wsnt:Message).
12. Verify that TopicExpression is equal to tns1:Door/State/DoorAlarm for received Notify message.
13. Verify that notification contains Source.SimpleItem item with Name = "DoorToken" and Value = “Token1”.
14. Verify that notification contains Data.SimpleItem item with Name = "State" and Value with type is equal to tdc:DoorAlarmState.

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:Door/State/DoorAlarm with valid DoorToken.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid DoorToken value).
- The DUT did not return at least one Door at step 3.

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 10 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

Note: If the DUT does not return any door for step 3, skip steps 4-14, 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.

4.5.12 DOOR CONTROL – DOOR FAULT EVENT

Test Case ID: DOORCONTROL-6-1-12

Specification Coverage: tns1:Door/State/DoorFault (ONVIF Door Control Service Specification), GetDoorState (ONVIF Door Control Service Specification), Properties (ONVIF Core Specification)

Feature Under Test: GetDoorState

WSDL Reference: event.wsdl, doorcontrol.wsdl

Test Purpose: To verify tns1:Door/State/DoorFault event generation after subscription and to verify tns1:Door/State/DoorFault event format.

Pre-Requisite: Event Service was received from the DUT. Device supports Pull-Point Notification feature. Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Fault capability 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 doors from the DUT (see Annex A.1).
4. Check that there is at least one Door with Capabilities.Fault = “true”. Otherwise 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:Door/State/DoorFault. 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 = "DoorToken" and Type = "pt:ReferenceToken".
10. Check that this event contains Data.SimpleItemDescription item with Name = "State" and Type = "tdc:DoorFaultState".
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:Door/State/DoorFault Topic as Filter and an InitialTerminationTime of timeout1.
13. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
14. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
15. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 14 until Notification for all Doors with Capabilities.DoorMonitor = “true” is received.
16. Verify received Notify messages (correct value for UTC time, TopicExpression and wsnt:Message).
17. Verify that TopicExpression is equal to tns1:Door/State/DoorFault for all received Notify messages.
18. Verify that each notification contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to one of existing Door Tokens Capabilities.Fault = “true” (e.g. complete list of doors contains Door with the same token). Verify that there are Notification messages for each Door with Capabilities.Fault = “true”.

19. Verify that each notification contains Data.SimpleItem item with Name = "State" and Value with type is equal to tdc:DoorFaultState.
20. Verify that each notification which contains Data.SimpleItem item with Name = "Reason" has Value with type is equal to xs:string.
21. Verify that Notify PropertyOperation = "Initialized".
22. ONVIF Client will invoke **GetDoorState** request for each Door with corresponding tokens.
23. Verify the **GetDoorStateResponse** messages from the DUT. Verify that Data.SimpleItem item with Name = "State" from Notification message has the same value with Fault elements from corresponding **GetDoorStateResponse** messages for each Door with Capabilities.Fault = “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 **PullMessagesResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a Notification message that contains a property event tns1:Door/State/DoorFault at least for one Door with Capabilities.Fault = “true”.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid DoorToken, Reason or State values, PropertyOperation is not equal to "Initialized").
- The DUT did not return at least one Door at step 3.
- The DUT did not return Topic tns1:Door/State/DoorFault 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 15 will wait for Notification messages until notification for all Doors with Capabilities.Fault = “true” is received or Operation Delay has expired. Notification messages for

all Doors with Capabilities.Fault = “true” are assumed as received, if the number of Notification messages is equal to the number of Doors with Capabilities.Fault = “true”.

Note: If the DUT does not return any door for step 3, skip steps 4-23, 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.

4.5.13 DOOR CONTROL – DOOR FAULT EVENT STATE CHANGE

Test Case ID: DOORCONTROL-6-1-13

Specification Coverage: tns1:Door/State/DoorFault (ONVIF Door Control Service Specification)

Feature Under Test: None

WSDL Reference: event.wsdl, doorcontrol.wsdl

Test Purpose: To verify tns1:Door/State/DoorFault event generation after property was changed and to verify tns1:Door/State/DoorFault event format.

Pre-Requisite: Event Service was received from the DUT. Device supports Pull-Point Notification feature. Door Control Service was received from the DUT. Door Entity is supported by the DUT. At least one Door with Fault capability 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 doors from the DUT (see Annex A.1).
4. Check that there is at least one Door with Capabilities.Fault = “true”. Otherwise, fail the test and skip other steps.
5. ONVIF Client will select one first Door (token = Token1) with Capabilities.Fault = “true”.
6. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorFault Topic as Filter and an InitialTerminationTime of timeout1.
7. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
8. Test Operator will invoke change of DoorFault property for Door with token = Token1.
9. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
10. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 9 until Notification with PropertyOperation = "Changed" for Door with Capabilities.Fault = “true” is received.
11. Verify received Notify messages (correct value for UTC time, TopicExpression and wsnt:Message).
12. Verify that TopicExpression is equal to tns1:Door/State/DoorFault for received Notify message.
13. Verify that notification contains Source.SimpleItem item with Name = "DoorToken" and Value = “Token1”.
14. Verify that notification contains Data.SimpleItem item with Name = "State" and Value with type is equal to tdc:DoorFaultState.
15. Verify that notification contains Data.SimpleItem item with Name = "Reason" and Value with type is equal to xs:string.

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:Door/State/DoorFault with valid DoorToken.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem or Data.SimpleItem, wrong type of Value fields, invalid DoorToken value).
- The DUT did not return at least one Door at step 3.

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 10 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

Note: If the DUT does not return any door for step 3, skip steps 4-15, 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.

4.6 Door Configuration

4.6.1 DOOR CONTROL – ADD OR CHANGE DOOR EVENT

Test Case ID: DOORCONTROL-7-1-1

Specification Coverage: tns1:Configuration/Door/Changed (ONVIF Door Control Service Specification), GetDoorInfo (ONVIF Door Control Service Specification)

Feature Under Test: GetDoorInfo

WSDL Reference: event.wsdl, doorcontrol.wsdl

Test Purpose: To verify tns1:Configuration/Door/Changed event generation after adding new door or changing door configuration to the DUT and to verify tns1:Configuration/Door/Changed event format.

Pre-Requisite: Event Service was received from the DUT. Device supports Pull-Point Notification feature. Door Control Service was received from the DUT. At least one Door could be added to the DUT or changed.

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/Door/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 = "DoorToken" and Type = "pt:ReferenceToken".
8. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Configuration/Door/Changed Topic as Filter and an InitialTerminationTime of timeout1.
9. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
10. Test Operator will add Door or change Door configuration.
11. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
12. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessages. Repeat step 11 until Notification received.

13. Verify received Notify messages (correct value for UTC time, TopicExpression and wsnt:Message).
14. Verify that TopicExpression is equal to tns1:Configuration/Door/Changed for received message.
15. Verify that notification contains Source.SimpleItem item with Name = "DoorToken" and Value with type is equal to tdc:ReferenceToken.
16. ONVIF Client will invoke **GetDoorInfo** request (Token from Notify message) to retrieve a subset of Door Information from the DUT.
17. Verify the **GetDoorInfoResponse** message from the DUT. Verify that requested Door 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 **GetDoorInfoResponse** with specified Door.
- The DUT did not send a Notification message that contains an event tns1:Configuration/Door/Changed with valid DoorToken.
- The DUT sent an invalid Notification message (invalid DoorToken value).
- The DUT did not return at least one Door at step 3.

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 12 will wait for 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.

4.6.2 DOOR CONTROL – REMOVE DOOR EVENT

Test Case ID: DOORCONTROL-7-1-2

Specification Coverage: tns1:Configuration/Door/Removed (ONVIF Door Control Service Specification), GetDoorInfo (ONVIF Door Control Service Specification)

Feature Under Test: GetDoorInfo

WSDL Reference: event.wsdl, doorcontrol.wsdl

Test Purpose: To verify tns1: Configuration/Door/Removed event generation after removing door from the DUT and to verify tns1: Configuration/Door/Removed event format.

Pre-Requisite: Event Service was received from the DUT. Device supports Pull-Point Notification feature. Door Control Service was received from the DUT. At least one Door could be removed 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 **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/Door/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 = "DoorToken" and Type = "pt:ReferenceToken".
8. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Configuration/Door/Removed Topic as Filter and an InitialTerminationTime of timeout1.
9. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
10. Test Operator will remove Door.
11. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
12. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessages. Repeat step 11 until Notification received.
13. Verify received Notify messages (correct value for UTC time, TopicExpression and wsnt:Message).
14. Verify that TopicExpression is equal to tns1:Configuration/Door/Removed for received message.
15. Verify that notification contains Source.SimpleItem item with Name = "DoorToken" and Value with type is equal to tdc:ReferenceToken.
16. ONVIF Client will invoke **GetDoorInfo** request (Token from Notify message) to retrieve subset of Door Information from the DUT.
17. Verify the **GetDoorInfoResponse** 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 **PullMessagesResponse** message.

- The DUT did not send valid SubscriptionReference.
- The DUT did not send a **GetDoorInfoResponse** with empty list.
- The DUT did not send a Notification message that contains an event tns1:Configuration/Door/Removed with valid DoorToken.
- The DUT sent an invalid Notification message (invalid DoorToken value).
- The DUT did not return at least one Door at step 3.

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 12 will wait for 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.

4.7 Door Management

4.7.1 GET DOORS

Test Case ID: DOORCONTROL-8-1-1

Specification Coverage: GetDoors command (ONVIF Door Control Service Specification), GetDoorList command (ONVIF Door Control Service Specification),

Feature Under Test: GetDoors

WSDL Reference: door.wsdl

Test Purpose: To verify Get Doors command.

Pre-Requisite: Door Control Service is received from the DUT. Door Management is supported by the DUT as indicated by DoorManagementSupported 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 doors (out *doorCompleteList*) by following the procedure mentioned in Annex A.6.
4. If the DUT does not supports Door Entity and *doorCompleteList* is not empty, FAIL the test and skip other steps.
5. If the DUT does not supports Door Entity, skip other steps.
6. If *doorCompleteList* is empty, FAIL the test and skip other steps.
7. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.7.
8. Set the following:
   - *tokenList* := [subset of *doorCompleteList*.token values with items number equal to *cap*.MaxLimit]
9. ONVIF client invokes **GetDoors** with parameters
   - Token list := *tokenList*
10. The DUT responds with **GetDoorsResponse** message with parameters
    - Door list =: *doorList1*
11. If *doorList1* does not contain Door item for each token from *tokenList*, FAIL the test and skip other steps.
12. If *doorList1* contains at least two Door items with equal token, FAIL the test and skip other steps.
13. If *doorList1* contains other Door items than listed in *tokenList*, FAIL the test and skip other steps.

14. For each Door.token *token* from *doorCompleteList* repeat the following steps:

    14.1. ONVIF client invokes **GetDoors** with parameters

    - Token[0] := *token*

    14.2. The DUT responds with **GetDoorsResponse** message with parameters

    - Door list =: *doorList2*

    14.3. If *doorList2* does not contain only one Door item with token equal to *token*, FAIL the test and skip other steps.

    14.4. If *doorList2*[0] item does not have equal field values to *doorCompleteList*[token = *token*] item, FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Note: If number of items in *doorCompleteList* is less than *cap*.MaxLimit, then all *doorCompleteList*.Token items shall be used for the step 8.

Note: The following fields are compared at step 14.4:

- Door:
  - token
  - Name
  - Description
  - Capabilities
    - Access
    - AccessTimingOverride
    - Lock
    - Unlock

- Block
- DoubleLock
- LockDown
- LockOpen
- DoorMonitor
- LockMonitor
- DoubleLockMonitor
- Alarm
- Tamper
- Fault

- DoorType
- Timings
  - ReleaseTime
  - OpenTime
  - ExtendedReleaseTime
  - DelayTimeBeforeRelock
  - ExtendedOpenTime
  - PreAlarmTime

4.7.2 GET DOOR LIST - LIMIT

Test Case ID: DOORCONTROL-8-1-2

Specification Coverage: GetDoorList command (ONVIF Door Control Service Specification)

Feature Under Test: GetDoorList

WSDL Reference: door.wsdl

Test Purpose: To verify Get Door List using Limit.

Pre-Requisite: Door Control Service is received from the DUT. Door Management is supported by the DUT as indicated by DoorManagementSupported capability. Door Entity is supported by the DUT as indicated by MaxDoors > 0 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.7.
4. ONVIF client invokes **GetDoorList** with parameters
   - Limit := 1
   - StartReference skipped
5. The DUT responds with **GetDoorListResponse** message with parameters
   - NextStartReference
   - Door list =: *doorList1*
6. If *doorList1* contains more Door items than 1, FAIL the test and skip other steps.
7. If *cap*.MaxLimit = 1, skip other steps.
8. ONVIF client invokes **GetDoorList** with parameters
   - Limit := *cap*.MaxLimit
   - StartReference skipped
9. The DUT responds with **GetDoorListResponse** message with parameters
   - NextStartReference
   - Door list =: *doorList2*
10. If *doorList2* contains more Door items than *cap*.MaxLimit, FAIL the test and skip other steps.
11. If *cap*.MaxLimit = 2, skip other steps.
12. Set the following:
    - *limit* := [number between 1 and *cap*.MaxLimit]

13. ONVIF client invokes **GetDoorList** with parameters
    - Limit := *limit*
    - StartReference skipped
14. The DUT responds with **GetDoorListResponse** message with parameters
    - NextStartReference
    - Door list =: *doorList3*
15. If *doorList3* contains more Door items than *limit*, FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

4.7.3 GET DOOR LIST - START REFERENCE AND LIMIT

Test Case ID: DOORCONTROL-8-1-3

Specification Coverage: GetDoorList command (ONVIF Door Control Service Specification)

Feature Under Test: GetDoorList

WSDL Reference: door.wsdl

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

Pre-Requisite: Door Control Service is received from the DUT. Door Management is supported by the DUT as indicated by DoorManagementSupported capability. Door Entity is supported by the DUT as indicated by MaxDoors > 0 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 by following the procedure mentioned in Annex A.7 with the following output parameters
   - out *cap* - Door Control capabilities
4. ONVIF client invokes **GetDoorList** with parameters
   - Limit := *cap*.MaxLimit
   - StartReference skipped
5. The DUT responds with **GetDoorListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - Door list =: *doorCompleteList1*
6. If *doorCompleteList1* contains more Door 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 **GetDoorList** with parameters
      - Limit := *cap*.MaxLimit
      - StartReference := *nextStartReference*
   7.2. The DUT responds with **GetDoorListResponse** message with parameters
      - NextStartReference =: *nextStartReference*
      - Door list =: *doorListPart*
   7.3. If *doorListPart* contains more Door items than *cap*.MaxLimit, FAIL the test and skip other steps.
   7.4. Set the following:
      - *doorCompleteList1* := *doorCompleteList1* + *doorListPart*
8. If *doorCompleteList1* contains at least two Door item with equal token, FAIL the test and skip other steps.
9. If *cap*.MaxLimit = 1:
   9.1. ONVIF Client retrieves a complete list of door info by following the procedure mentioned in Annex A.1 with the following output parameters

- out *doorInfoCompleteList* - complete door info list

9.2. If *doorCompleteList1* does not contain all doors from *doorInfoCompleteList*, FAIL the test and skip other steps.

9.3. If *doorCompleteList1* contains doors other than doors from *doorInfoCompleteList*, FAIL the test and skip other steps.

9.4. For each DoorInfo.token *token* from *doorInfoCompleteList* repeat the following steps:

9.4.1. If *doorCompleteList1*[token = *token*] item does not have equal field values to *doorInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.

9.5. Skip other steps.

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

- Limit := 1
- StartReference skipped

11. The DUT responds with **GetDoorListResponse** message with parameters

- NextStartReference =: *nextStartReference*
- Door list =: *doorCompleteList2*

12. If *doorCompleteList2* contains more Door 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 **GetDoorList** with parameters

- Limit := 1
- StartReference := *nextStartReference*

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

- NextStartReference =: *nextStartReference*
- Door list =: *doorListPart*

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

13.4. Set the following:

- *doorCompleteList2* := *doorCompleteList2* + *doorListPart*

14. If *doorCompleteList2* contains at least two Door item with equal token, FAIL the test and skip other steps.
15. If *doorCompleteList2* does not contain all doors from *doorCompleteList1*, FAIL the test and skip other steps.
16. If *doorCompleteList2* contains doors other than doors from *doorCompleteList1*, FAIL the test and skip other steps.
17. If *cap*.MaxLimit = 2:
    17.1. ONVIF Client retrieves a complete list of doors by following the procedure mentioned in Annex A.1 with the following output parameters
    - out *doorInfoCompleteList* - complete door info list

    17.2. If *doorCompleteList2* does not contain all doors from *doorInfoCompleteList*, FAIL the test and skip other steps.
    17.3. If *doorCompleteList2* contains doors other than doors from *doorInfoCompleteList*, FAIL the test and skip other steps.
    17.4. For each DoorInfo.token *token* from *doorInfoCompleteList* repeat the following steps:
        17.4.1. If *doorCompleteList2*[token = *token*] item does not have equal field values to *doorInfoCompleteList*[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 **GetDoorList** with parameters
    - Limit := *limit*
    - StartReference skipped
20. The DUT responds with **GetDoorListResponse** message with parameters
    - NextStartReference =: *nextStartReference*
    - Door list =: *doorCompleteList3*
21. If *doorCompleteList3* contains more Door 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 **GetDoorList** with parameters

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

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

- NextStartReference =: *nextStartReference*
- Door list =: *doorListPart*

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

22.4. Set the following:

- *doorCompleteList3* := *doorCompleteList3* + *doorListPart*

23. If *doorCompleteList3* contains at least two Door items with equal token, FAIL the test and skip other steps.
24. If *doorCompleteList3* does not contain all doors from *doorCompleteList1*, FAIL the test and skip other steps.
25. If *doorCompleteList3* contains doors other than doors from *doorCompleteList1*, FAIL the test and skip other steps.
26. ONVIF Client retrieves a complete list of doors by following the procedure mentioned in Annex A.1 with the following output parameters
    - out *doorInfoCompleteList* - complete door info list
27. If *doorCompleteList3* does not contain all doors from *doorInfoCompleteList*, FAIL the test and skip other steps.
28. If *doorCompleteList3* contains doors other than doors from *doorInfoCompleteList*, FAIL the test and skip other steps.
29. For each DoorInfo.token *token* from *doorInfoCompleteList* repeat the following steps:

    29.1. If *doorCompleteList3*[token = *token*] item does not have equal field values to *doorInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Note: The following fields are compared at step 29.1:

- Door Info:
  - token
  - Name
  - Description
  - Capabilities
    - Access
    - AccessTimingOverride
    - Lock
    - Unlock
    - Block
    - DoubleLock
    - LockDown
    - LockOpen
    - DoorMonitor
    - LockMonitor
    - DoubleLockMonitor
    - Alarm
    - Tamper
    - Fault

4.7.4 GET DOOR LIST - NO LIMIT

Test Case ID: DOORCONTROL-8-1-4

Specification Coverage: GetDoorList command (ONVIF Door Control Service Specification)

Feature Under Test: GetDoorList

WSDL Reference: door.wsdl

Test Purpose: To verify Get Door List without using Limit.

Pre-Requisite: Door Control Service is received from the DUT. Door Management is supported by the DUT as indicated by DoorManagementSupported capability. Door Entity is supported by the DUT as indicated by MaxDoors > 0 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.7.
4. ONVIF client invokes **GetDoorList** with parameters
   - Limit skipped
   - StartReference skipped
5. The DUT responds with **GetDoorListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - Door list =: *doorCompleteList*
6. If *doorCompleteList* contains more Door 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 **GetDoorList** with parameters
      - Limit skipped
      - StartReference := *nextStartReference*
   7.2. The DUT responds with **GetDoorListResponse** message with parameters
      - NextStartReference =: *nextStartReference*
      - Door list =: *doorListPart*

7.3. If *doorListPart* contains more Door items than *cap*.MaxLimit, FAIL the test and skip other steps.

7.4. Set the following:

- *doorCompleteList* := *doorCompleteList* + *doorListPart*

8. If *doorCompleteList* contains at least two Door item with equal token, FAIL the test.
9. ONVIF Client retrieves a complete list of doors by following the procedure mentioned in Annex A.1 with the following output parameters
   - out *doorInfoCompleteList* - complete door info list
10. If *doorCompleteList* does not contain all doors from *doorInfoCompleteList*, FAIL the test and skip other steps.
11. If *doorCompleteList* contains doors other than doors from *doorInfoCompleteList*, FAIL the test and skip other steps.
12. For each DoorInfo.token *token* from *doorInfoCompleteList* repeat the following steps:

    12.1. If *doorCompleteList*[token = *token*] item does not have equal field values to *doorInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Note: The following fields are compared at step 12.1:

- DoorInfo:
  - token
  - Name
  - Description
  - Capabilities
    - Access

- AccessTimingOverride
- Lock
- Unlock
- Block
- DoubleLock
- LockDown
- LockOpen
- DoorMonitor
- LockMonitor
- DoubleLockMonitor
- Alarm
- Tamper
- Fault

4.7.5 CREATE DOOR (DOOR CAPABILITIES TRUE)

Test Case ID: DOORCONTROL-8-1-5

Specification Coverage: CreateDoor command (ONVIF Door Control Service Specification), Door (ONVIF Door Control Service Specification)

Feature Under Test: CreateDoor

WSDL Reference: door.wsdl, event.wsdl

Test Purpose: To verify creation of door and generating of door changed notifications.

Pre-Requisite: Door Control Service is received from the DUT. Event Service was received from the DUT. Device supports Pull-Point Notification feature. The DUT shall have enough free storage capacity for one additional Door. Door Management is supported by the DUT as indicated by DoorManagementSupported capability. Door Entity is supported by the DUT as indicated by MaxDoors > 0 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 doors by following the procedure mentioned in Annex A.6 with the following output parameters
   - out *doorCompleteList1* - complete door list
4. ONVIF Client checks free storage for additional Door by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - in *doorCompleteList1* - complete door list
   - out *doorToRestore* - deleted door
5. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.13 with the following input and output parameters
   - in **tns1:Configuration/Door/Changed** - Notification Topic
   - out *s* - Subscription reference
   - out *currentTime* - current time for the DUT
   - out *terminationTime* - Subscription termination time
6. ONVIF client invokes **CreateDoor** with parameters
   - Door.token := ""
   - Door.Name := "Test Name"
   - Door.Description := "Test Description"
   - Door.Capabilities.Access := true
   - Door.Capabilities.AccessTimingOverride := true
   - Door.Capabilities.Lock := true
   - Door.Capabilities.Unlock := true
   - Door.Capabilities.Block := true
   - Door.Capabilities.DoubleLock := true

- Door.Capabilities.LockDown := true
- Door.Capabilities.LockOpen := true
- Door.Capabilities.DoorMonitor := true
- Door.Capabilities.LockMonitor := true
- Door.Capabilities.DoubleLockMonitor := true
- Door.Capabilities.Alarm := true
- Door.Capabilities.Tamper := true
- Door.Capabilities.Fault := true
- Door.DoorType := "pt:Door"
- Door.Timings.ReleaseTime := PT60S
- Door.Timings.OpenTime := PT120S
- Door.Timings.ExtendedReleaseTime := PT30S
- Door.Timings.DelayTimeBeforeRelock := PT30S
- Door.Timings.ExtendedOpenTime := PT30S
- Door.Timings.PreAlarmTime := PT10S

7. The DUT responds with **CreateDoorResponse** message with parameters
   - Token =: *doorToken*
8. ONVIF Client retrieves and checks **tns1:Configuration/Door/Changed** event for the specified Door token by following the procedure mentioned in Annex A.15 with the following input and output parameters
   - in *s* - Subscription reference
   - in *currentTime* - current time for the DUT
   - in *terminationTime* - subscription termination time
   - in *doorToken* - Door token
9. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.14 with the following input and output parameters

- in *s* - Subscription reference

10. ONVIF Client retrieves a door by following the procedure mentioned in Annex A.10 with the following input and output parameters
    - in *doorToken* - door token
    - out *doorList* - door list
11. If *doorList*[0] item does not have equal field values to values from step 6, FAIL the test and go step 18.
12. ONVIF Client retrieves a door info by following the procedure mentioned in Annex A.11 with the following input and output parameters
    - in *doorToken* - door token
    - out *doorInfoList* - door info list
13. If *doorInfoList*[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 door information list by following the procedure mentioned in Annex A.1 with the following input and output parameters
    - out *doorInfoCompleteList* - door info list
15. If *doorInfoCompleteList* does not have *doorInfo*.[token = *doorToken*] 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 doors by following the procedure mentioned in Annex A.6 with the following input and output parameters
    - out *doorCompleteList2* - door list
17. If *doorCompleteList2* does not have *door*.[token = *doorToken*] item with equal field values to values from step 6, FAIL the test and go step 18.
18. ONVIF Client deletes the Door by following the procedure mentioned in Annex A.9 to restore DUT configuration with the following input and output parameters
    - in *doorToken* - door token
19. ONVIF Client restores door deleted at step 4 if any.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Note: The following fields are compared at steps 11, 17:

- Door:
  - token
  - Name
  - Description
  - Capabilities
    - Access
    - AccessTimingOverride
    - Lock
    - Unlock
    - Block
    - DoubleLock
    - LockDown
    - LockOpen
    - DoorMonitor
    - LockMonitor
    - DoubleLockMonitor
    - Alarm
    - Tamper
    - Fault
  - DoorType
  - Timings

- ReleaseTime
- OpenTime
- ExtendedReleaseTime
- DelayTimeBeforeRelock
- ExtendedOpenTime
- PreAlarmTime

Note: The following fields are compared at steps 13, 15:

- DoorInfo:
  - token
  - Name
  - Description
  - Capabilities
    - Access
    - AccessTimingOverride
    - Lock
    - Unlock
    - Block
    - DoubleLock
    - LockDown
    - LockOpen
    - DoorMonitor
    - LockMonitor
    - DoubleLockMonitor
    - Alarm

- Tamper
- Fault

4.7.6 CREATE DOOR (DOOR CAPABILITIES FALSE)

Test Case ID: DOORCONTROL-8-1-6

Specification Coverage: CreateDoor command (ONVIF Door Control Service Specification), Door (ONVIF Door Control Service Specification)

Feature Under Test: CreateDoor

WSDL Reference: door.wsdl, event.wsdl

Test Purpose: To verify creation of door and generating of door changed notifications.

Pre-Requisite: Door Control Service is received from the DUT. Event Service was received from the DUT. Device supports Pull-Point Notification feature. The DUT shall have enough free storage capacity for one additional Door. Door Management is supported by the DUT as indicated by DoorManagementSupported capability. Door Entity is supported by the DUT as indicated by MaxDoors > 0 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 doors by following the procedure mentioned in Annex A.6 with the following output parameters
   - out *doorCompleteList1* - complete door list
4. ONVIF Client checks free storage for additional Door by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - in *doorCompleteList1* - complete door list
   - out *doorToRestore* - deleted door
5. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.13 with the following input and output parameters

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

6. ONVIF client invokes **CreateDoor** with parameters
   - Door.token := ""
   - Door.Name := "Test Name"
   - Door.Description := "Test Description"
   - Door.Capabilities.Access := false
   - Door.Capabilities.AccessTimingOverride := false
   - Door.Capabilities.Lock := false
   - Door.Capabilities.Unlock := false
   - Door.Capabilities.Block := false
   - Door.Capabilities.DoubleLock := false
   - Door.Capabilities.LockDown := false
   - Door.Capabilities.LockOpen := false
   - Door.Capabilities.DoorMonitor := false
   - Door.Capabilities.LockMonitor := false
   - Door.Capabilities.DoubleLockMonitor := false
   - Door.Capabilities.Alarm := false
   - Door.Capabilities.Tamper := false
   - Door.Capabilities.Fault := false
   - Door.DoorType := "pt:ManTrap"
   - Door.Timings.ReleaseTime := PT60S

- Door.Timings.OpenTime := PT120S
- Door.Timings.ExtendedReleaseTime skipped
- Door.Timings.DelayTimeBeforeRelock skipped
- Door.Timings.ExtendedOpenTime skipped
- Door.Timings.PreAlarmTime skipped

7. The DUT responds with **CreateDoorResponse** message with parameters
   - Token =: *doorToken*
8. ONVIF Client retrieves and checks **tns1:Configuration/Door/Changed** event for the specified Door token by following the procedure mentioned in Annex A.15 with the following input and output parameters
   - in *s* - Subscription reference
   - in *currentTime* - current time for the DUT
   - in *terminationTime* - subscription termination time
   - in *doorToken* - Door token
9. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.14 with the following input and output parameters
   - in *s* - Subscription reference
10. ONVIF Client retrieves a door by following the procedure mentioned in Annex A.10 with the following input and output parameters
    - in *doorToken* - door token
    - out *doorList* - door list
11. If *doorList*[0] item does not have equal field values to values from step 6, FAIL the test and go step 18.
12. ONVIF Client retrieves a door info by following the procedure mentioned in Annex A.11 with the following input and output parameters
    - in *doorToken* - door token
    - out *doorInfoList* - door info list

13. If *doorInfoList*[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 door information list by following the procedure mentioned in Annex A.1 with the following input and output parameters
    - out *doorInfoCompleteList* - door info list
15. If *doorInfoCompleteList* does not have *doorInfo*.[token = *doorToken*] 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 doors by following the procedure mentioned in Annex A.6 with the following input and output parameters
    - out *doorCompleteList2* - door list
17. If *doorCompleteList2* does not have *door*.[token = *doorToken*] item with equal field values to values from step 6, FAIL the test and go step 18.
18. ONVIF Client deletes the Door by following the procedure mentioned in Annex A.9 to restore DUT configuration with the following input and output parameters
    - in *doorToken* - door token
19. ONVIF Client restores door deleted at step 4 if any.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Note: The following fields are compared at steps 11, 17:

- Door:
  - token
  - Name
  - Description
  - Capabilities
    - Access

- AccessTimingOverride
- Lock
- Unlock
- Block
- DoubleLock
- LockDown
- LockOpen
- DoorMonitor
- LockMonitor
- DoubleLockMonitor
- Alarm
- Tamper
- Fault

- DoorType
- Timings
  - ReleaseTime
  - OpenTime
  - ExtendedReleaseTime
  - DelayTimeBeforeRelock
  - ExtendedOpenTime
  - PreAlarmTime

Note: The following fields are compared at steps 13, 15:

- DoorInfo:
  - token

- Name
- Description
- Capabilities
  - Access
  - AccessTimingOverride
  - Lock
  - Unlock
  - Block
  - DoubleLock
  - LockDown
  - LockOpen
  - DoorMonitor
  - LockMonitor
  - DoubleLockMonitor
  - Alarm
  - Tamper
  - Fault

4.7.7 MODIFY DOOR

Test Case ID: DOORCONTROL-8-1-7

Specification Coverage: DoorInfo (ONVIF Door Control Service Specification), Door (ONVIF Door Control Service Specification), ModifyDoor command (ONVIF Door Control Service Specification)

Feature Under Test: ModifyDoor

WSDL Reference: door.wsdl, event.wsdl

Test Purpose: To verify modifying of door and generating of door changed notifications.

Pre-Requisite: Door Control Service is received from the DUT. Event Service was received from the DUT. Device supports Pull-Point Notification feature. The DUT shall have enough free storage capacity for one additional Door. Door Management is supported by the DUT as indicated by DoorManagementSupported capability. Door Entity is supported by the DUT as indicated by MaxDoors > 0 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 by following the procedure mentioned in Annex A.7 with the following output parameters
   - out *cap* - door control capabilities
4. ONVIF Client retrieves a complete list of doors by following the procedure mentioned in Annex A.6 with the following output parameters
   - out *doorCompleteList1* - complete door list
5. ONVIF Client checks free storage for additional Door by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - in *doorCompleteList1* - complete door list
   - out *doorToRestore* - deleted door
6. ONVIF Client creates door by following the procedure mentioned in Annex A.12 with the following input and output parameters
   - in false - for all door capabilities
   - in "pt:Door" - Door Type
   - in PT60S - ReleaseTime
   - in PT120S - OpenTime
7. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.13 with the following input and output parameters
   - in **tns1:Configuration/Door/Changed** - Notification Topic
   - out *s* - Subscription reference

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

8. ONVIF client invokes **ModifyDoor** with parameters
   - Door.Token := *doorToken*
   - Door.Name := "Test Name 2"
   - Door.Description := "Test Description 2"
   - Door.Capabilities.Access := true
   - Door.Capabilities.AccessTimingOverride := true
   - Door.Capabilities.Lock := true
   - Door.Capabilities.Unlock := true
   - Door.Capabilities.Block := true
   - Door.Capabilities.DoubleLock := true
   - Door.Capabilities.LockDown := true
   - Door.Capabilities.LockOpen := true
   - Door.Capabilities.DoorMonitor := true
   - Door.Capabilities.LockMonitor := true
   - Door.Capabilities.DoubleLockMonitor := true
   - Door.Capabilities.Alarm := true
   - Door.Capabilities.Tamper := true
   - Door.Capabilities.Fault := true
   - Door.DoorType := "pt:Door"
   - Door.Timings.ReleaseTime := PT30S
   - Door.Timings.OpenTime := PT60S
   - Door.Timings.ExtendedReleaseTime := PT10S

- Door.Timings.DelayTimeBeforeRelock := PT10S
- Door.Timings.ExtendedOpenTime := PT10S
- Door.Timings.PreAlarmTime := PT10S

9. The DUT responds with empty **ModifyDoorResponse** message.
10. ONVIF Client retrieves and checks **tns1:Configuration/Door/Changed** event for the specified Door token by following the procedure mentioned in Annex A.15 with the following input and output parameters
    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *doorToken* - Door token
11. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.14 with the following input and output parameters
    - in *s* - Subscription reference
12. ONVIF Client retrieves a door by following the procedure mentioned in Annex A.10 with the following input and output parameters
    - in *doorToken* - door token
    - out *doorList* - door list
13. If *doorList*[0] item does not have equal field values to values from step 8, FAIL the test and go step 20.
14. ONVIF Client retrieves a door info by following the procedure mentioned in Annex A.11 with the following input and output parameters
    - in *doorToken* - door token
    - out *doorInfoList* - door info list
15. If *doorInfoList*[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 door information list by following the procedure mentioned in Annex A.1 with the following input and output parameters

- out *doorInfoCompleteList* - door info list

17. If *doorInfoCompleteList* does not have *doorInfo*.[token:= *doorToken*] 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 doors by following the procedure mentioned in Annex A.6 with the following input and output parameters
    - out *doorCompleteList2* - door list
19. If *doorCompleteList2* does not have *door*.[token:= *doorToken*] item with equal field values to values from step 8, FAIL the test and go step 20.
20. ONVIF Client deletes the Door by following the procedure mentioned in Annex A.9 to restore DUT configuration with the following input and output parameters
    - in *doorToken* - door token
21. ONVIF Client restores door deleted at step 5 if any.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

- Door:
  - token
  - Name
  - Description
  - Capabilities
    - Access
    - AccessTimingOverride
    - Lock

- Unlock
- Block
- DoubleLock
- LockDown
- LockOpen
- DoorMonitor
- LockMonitor
- DoubleLockMonitor
- Alarm
- Tamper
- Fault

- DoorType
- Timings
  - ReleaseTime
  - OpenTime
  - ExtendedReleaseTime
  - DelayTimeBeforeRelock
  - ExtendedOpenTime
  - PreAlarmTime

Note: The following fields are compared at steps 13, 15:

- DoorInfo:
  - token
  - Name
  - Description

- Capabilities
  - Access
  - AccessTimingOverride
  - Lock
  - Unlock
  - Block
  - DoubleLock
  - LockDown
  - LockOpen
  - DoorMonitor
  - LockMonitor
  - DoubleLockMonitor
  - Alarm
  - Tamper
  - Fault

4.7.8 DELETE DOOR

Test Case ID: DOORCONTROL-8-1-8

Specification Coverage: DeleteDoor command (ONVIF Door Control Service Specification)

Feature Under Test: DeleteDoor

WSDL Reference: door.wsdl and event.wsdl

Test Purpose: To verify deleting of door and generating of door removed notifications.

Pre-Requisite: Door Control Service is received from the DUT. Event Service was received from the DUT. Device supports Pull-Point Notification feature. The DUT shall have enough free storage capacity for one additional Door. Door Management is supported by the DUT as indicated by DoorManagementSupported capability. Door Entity is supported by the DUT as indicated by MaxDoors > 0 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 by following the procedure mentioned in Annex A.7 with the following output parameters
   - out *cap* - door control capabilities
4. ONVIF Client retrieves a complete list of doors by following the procedure mentioned in Annex A.6 with the following output parameters
   - out *doorCompleteList1* - complete door list
5. ONVIF Client checks free storage for additional Door by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - in *doorCompleteList1* - complete door list
   - out *doorToRestore* - deleted door
6. ONVIF Client creates door by following the procedure mentioned in Annex A.12 with the following input and output parameters
   - in false - for all door capabilities
   - in "pt:Door" - Door Type
   - in PT60S - ReleaseTime
   - in PT120S - OpenTime
7. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.13 with the following input and output parameters
   - in **tns1:Configuration/Door/Removed** - Notification Topic
   - out *s* - Subscription reference
   - out *currentTime* - current time for the DUT
   - out *terminationTime* - Subscription termination time
8. ONVIF Client invokes **DeleteDoor** with parameters

- Token := *doorToken*

9. The DUT responds with empty **DeleteDoorResponse** message.
10. ONVIF Client retrieves and checks **tns1:Configuration/Door/Removed** event for the specified Door token by following the procedure mentioned in Annex A.15 with the following input and output parameters
    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *doorToken* - Door token
11. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.14 with the following input and output parameters
    - in *s* - Subscription reference
12. ONVIF Client retrieves a door by following the procedure mentioned in Annex A.10 with the following input and output parameters
    - in *doorToken* - door token
    - out *doorList* - door list
13. If *doorList* is not empty, FAIL the test and go step 20.
14. ONVIF Client retrieves a door info by following the procedure mentioned in Annex A.11 with the following input and output parameters
    - in *doorToken* - door token
    - out *doorInfoList* - door info list
15. If *doorInfoList* is not empty, FAIL the test and go step 20.
16. ONVIF Client retrieves a complete door information list by following the procedure mentioned in Annex A.1 with the following input and output parameters
    - out *doorInfoCompleteList* - door info list
17. If *doorInfoCompleteList* contains *doorInfo*.[token:= *doorToken*] item, FAIL the test and go step 20.

18. ONVIF Client retrieves a complete list of doors by following the procedure mentioned in Annex A.6 with the following input and output parameters
    - out *doorCompleteList2* - door list
19. If *doorCompleteList2* contains *door*.[token:= *doorToken*] item, FAIL the test and go step 20.
20. ONVIF Client restores door deleted at step 5 if any.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

4.7.9 GET DOORS WITH INVALID TOKEN

Test Case ID: DOORCONTROL-8-1-9

Specification Coverage: GetDoors command (ONVIF Door Control Service Specification)

Feature Under Test: GetDoors

WSDL Reference: door.wsdl

Test Purpose: To verify Get Door with invalid token.

Pre-Requisite: Door Control Service is received from the DUT. Door Management is supported by the DUT as indicated by DoorManagementSupported 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 door info by following the procedure mentioned in Annex A.1 with the following output parameters
    - out *doorInfoCompleteList* - complete door info list
4. Set the following:

- *invalidToken* := value not equal to any *doorInfoCompleteList*.token

5. ONVIF client invokes **GetDoors** with parameters
   - Token[0] := *invalidToken*
6. The DUT responds with **GetDoorsResponse** message with parameters
   - Door list =: *doorsList*
7. If *doorsList* is not empty, FAIL the test.
8. If *doorInfoCompleteList* is empty, skip other steps.
9. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.7 with the following output parameters
   - out *cap* - door control capabilities
10. If *cap*.MaxLimit is less than 2, skip other steps.
11. ONVIF client invokes **GetDoors** with parameters
    - Token[0] := *invalidToken*
    - Token[1] := *doorInfoCompleteList*[0].token
12. The DUT responds with **GetDoorsResponse** message with parameters
    - DoorInfo list =: *doorsList*
13. If *doorsList* is empty, FAIL the test.
14. If *doorsList* contains more than one item, FAIL the test.
15. If *doorsList*[0].token does not equal to *doorInfoCompleteList*[0].token, FAIL the test.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

4.7.10 GET DOORS - TOO MANY ITEMS

Test Case ID: DOORCONTROL-8-1-10

Specification Coverage: GetDoors command (ONVIF Door Control Service Specification)

Feature Under Test: GetDoors

WSDL Reference: door.wsdl

Test Purpose: To verify Get Door if there are more items than MaxLimit in request.

Pre-Requisite: Door Control Service is received from the DUT. Door Management is supported by the DUT as indicated by DoorManagementSupported capability. Door Entity is supported by the DUT as indicated by MaxDoors > 0 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 doors by following the procedure mentioned in Annex A.6 with the following output parameters
   - out *doorCompleteList* - complete door list
4. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.7 with the following output parameters
   - out *cap* - door control capabilities
5. If *doorCompleteList*.token items number is less than *cap*.MaxLimit or equal to *cap*.MaxLimit, skip other steps.
6. Set the following:
   - *tokenList* := [subset of *doorCompleteList*.token values with items number equal to *cap*.MaxLimit + 1]
7. ONVIF client invokes **GetDoors** 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

4.7.11 CREATE DOOR - NOT EMPTY DOOR TOKEN

Test Case ID: DOORCONTROL-8-1-11

Specification Coverage: CreateDoor command (ONVIF Door Control Service Specification)

Feature Under Test: CreateDoor

WSDL Reference: door.wsdl

Test Purpose: To verify Create Door with not Empty Token Verification.

Pre-Requisite: Door Control Service is received from the DUT. The DUT shall have enough free storage capacity for one additional Door. Door Management is supported by the DUT as indicated by DoorManagementSupported 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 doors by following the procedure mentioned in Annex A.6 with the following output parameters
   - out *doorCompleteList1* - complete door list
4. ONVIF Client checks free storage for additional Door by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - in *doorCompleteList1* - complete door list
   - out *doorToRestore* - deleted door
5. ONVIF client invokes **CreateDoor** with parameters
   - Door.token := "DoorToken"
   - Door.Name := "Test Name"
   - Door.Description := "Test Description"

- Door.Capabilities.Access := false
- Door.Capabilities.AccessTimingOverride := false
- Door.Capabilities.Lock := false
- Door.Capabilities.Unlock := false
- Door.Capabilities.Block := false
- Door.Capabilities.DoubleLock := false
- Door.Capabilities.LockDown := false
- Door.Capabilities.LockOpen := false
- Door.Capabilities.DoorMonitor := false
- Door.Capabilities.LockMonitor := false
- Door.Capabilities.DoubleLockMonitor := false
- Door.Capabilities.Alarm := false
- Door.Capabilities.Tamper := false
- Door.Capabilities.Fault := false
- Door.DoorType := "pt:Door"
- Door.Timings.ReleaseTime := PT60S
- Door.Timings.OpenTime := PT120S
- Door.Timings.ExtendedReleaseTime skipped
- Door.Timings.DelayTimeBeforeRelock skipped
- Door.Timings.ExtendedOpenTime skipped
- Door.Timings.PreAlarmTime skipped

6. The DUT responds with **env:Sender\ter:InvalidArgVal** SOAP 1.2 fault.
7. ONVIF Client restores door deleted at step 4 if any.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

4.7.12 MODIFY DOOR WITH INVALID TOKEN

Test Case ID: DOORCONTROL-8-1-12

Specification Coverage: ModifyDoor command (ONVIF Door Control Service Specification)

Feature Under Test: ModifyDoor

WSDL Reference: door.wsdl

Test Purpose: To verify modifying of door with invalid token.

Pre-Requisite: Door Control Service is received from the DUT. Door Management is supported by the DUT as indicated by DoorManagementSupported 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 door info by following the procedure mentioned in Annex A.1 with the following output parameters
   - out *doorInfoCompleteList* - complete door info list
4. Set the following:
   - *invalidToken* := value not equal to any *doorInfoCompleteList*.token
5. ONVIF client invokes **ModifyDoor** with parameters
   - Door.Token := *invalidToken*
   - Door.Name := "Test Name"
   - Door.Description := "Test Description"
   - Door.Capabilities.Access := false

- Door.Capabilities.AccessTimingOverride := false
- Door.Capabilities.Lock := false
- Door.Capabilities.Unlock := false
- Door.Capabilities.Block := false
- Door.Capabilities.DoubleLock := false
- Door.Capabilities.LockDown := false
- Door.Capabilities.LockOpen := false
- Door.Capabilities.DoorMonitor := false
- Door.Capabilities.LockMonitor := false
- Door.Capabilities.DoubleLockMonitor := false
- Door.Capabilities.Alarm := false
- Door.Capabilities.Tamper := false
- Door.Capabilities.Fault := false
- Door.DoorType := "pt:ManTrap"
- Door.Timings.ReleaseTime := PT60S
- Door.Timings.OpenTime := PT120S
- Door.Timings.ExtendedReleaseTime skipped
- Door.Timings.DelayTimeBeforeRelock skipped
- Door.Timings.ExtendedOpenTime skipped
- Door.Timings.PreAlarmTime 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

4.7.13 DELETE DOOR WITH INVALID TOKEN

Test Case ID: DOORCONTROL-8-1-13

Specification Coverage: DeleteDoor command (ONVIF Door Control Service Specification)

Feature Under Test: DeleteDoor

WSDL Reference: door.wsdl

Test Purpose: To verify deleting of door with invalid token.

Pre-Requisite: Door Control Service is received from the DUT. Door Management is supported by the DUT as indicated by DoorManagementSupported 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 door info by following the procedure mentioned in Annex A.1 with the following output parameters
   - out *doorInfoCompleteList* - complete door info list
4. Set the following:
   - *invalidToken* := value not equal to any *doorInfoCompleteList*.token
5. ONVIF Client invokes **DeleteDoor** 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

4.7.14 CREATE NEW DOOR WITH SET DOOR (DOOR CAPABILITIES TRUE)

Test Case ID: DOORCONTROL-8-1-14

Specification Coverage: DoorInfo (ONVIF Door Control Service Specification), Door (ONVIF Door Control Service Specification), SetDoor command (ONVIF Door Control Service Specification)

Feature Under Test: SetDoor

WSDL Reference: door.wsdl, event.wsdl

Test Purpose: To verify creation of door using SetDoor command and generating of appropriate notifications.

Pre-Requisite: Door Control Service is received from the DUT. Event Service was received from the DUT. Device supports Pull-Point Notification feature. Client Supplied Token is supported by the DUT as indicated by ClientSuppliedTokenSupported capability. The DUT shall have enough free storage capacity for one additional Door. Door Entity is supported by the DUT as indicated by MaxDoors > 0 capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves Door Control Service Capabilities by following the procedure mentioned in Annex A.7 with the following input and output parameters
   - out *cap* - Door Control Service capabilities
4. ONVIF Client retrieves complete list of doors by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - out *initialDoorCompleteList* - door complete list
5. ONVIF Client checks free storage for additional door by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - in *initialDoorCompleteList* - door complete list

- out *doorToRestore* - removed door

6. Set *doorToken* := token that differs from tokens listed in *initialDoorCompleteList*.
7. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.13 with the following input and output parameters
   - in **tns1:Configuration/Door/Changed** - Notification Topic
   - out *s* - Subscription reference
   - out *currentTime* - current time for the DUT
   - out *terminationTime* - Subscription termination time
8. ONVIF client invokes **SetDoor** request with parameters
   - Door.token := *doorToken*
   - Door.Name := "Test Name"
   - Door.Description := "Test Description"
   - Door.Capabilities.Access := true
   - Door.Capabilities.AccessTimingOverride := true
   - Door.Capabilities.Lock := true
   - Door.Capabilities.Unlock := true
   - Door.Capabilities.Block := true
   - Door.Capabilities.DoubleLock := true
   - Door.Capabilities.LockDown := true
   - Door.Capabilities.LockOpen := true
   - Door.Capabilities.DoorMonitor := true
   - Door.Capabilities.LockMonitor := true
   - Door.Capabilities.DoubleLockMonitor := true
   - Door.Capabilities.Alarm := true
   - Door.Capabilities.Tamper := true

- Door.Capabilities.Fault := true
- Door.DoorType := "pt:Door"
- Door.Timings.ReleaseTime := PT60S
- Door.Timings.OpenTime := PT120S
- Door.Timings.ExtendedReleaseTime := PT30S
- Door.Timings.DelayTimeBeforeRelock := PT30S
- Door.Timings.ExtendedOpenTime := PT30S
- Door.Timings.PreAlarmTime := PT10S

9. The DUT responds with **SetDoorResponse** message.
10. ONVIF Client retrieves and checks **tns1:Configuration/Door/Changed** event for the specified Door token by following the procedure mentioned in Annex A.15 with the following input and output parameters
    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *doorToken* - Door token
11. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.14 with the following input and output parameters
    - in *s* - Subscription reference
12. ONVIF Client retrieves a door by following the procedure mentioned in Annex A.10 with the following input and output parameters
    - in *doorToken* - door token
    - out *doorList* - the list of doors
13. If *doorList*[0] item does not have equal field values to values from step 8, FAIL the test, restore the DUT state, and skip other steps.
14. ONVIF Client retrieves a door info by following the procedure mentioned in Annex A.11 with the following input and output parameters

- in *doorToken* - door token
- out *doorInfoList* - the list of doors info

15. If *doorInfoList*[0] item does not have equal field values to values from step 8, FAIL the test, restore the DUT state, and skip other steps.
16. ONVIF Client retrieves complete list of doors info by following the procedure mentioned in Annex A.1 with the following input and output parameters
    - out *doorInfoCompleteList* - door info complete list
17. If *doorInfoCompleteList* does not have DoorInfo[token = *doorToken*] item with equal field values to values from step 8, FAIL the test, restore the DUT state, and skip other steps.
18. ONVIF Client retrieves complete list of doors by following the procedure mentioned in Annex A.6 with the following input and output parameters
    - out *doorCompleteList* - door complete list
19. If *doorCompleteList* does not have Door[token = *doorToken*] item with equal field values to values from step 8, FAIL the test, restore the DUT state, and skip other steps.
20. ONVIF Client deletes the Door by following the procedure mentioned in Annex A.9 with the following input and output parameters
    - in *doorToken* - door token
21. ONVIF Client restores door deleted at step 5 if any.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Note: The following fields are compared at steps 13 and 19:

- Door:
  - token
  - Name
  - Description

- Capabilities
  - Access
  - AccessTimingOverride
  - Lock
  - Unlock
  - Block
  - DoubleLock
  - LockDown
  - LockOpen
  - DoorMonitor
  - LockMonitor
  - DoubleLockMonitor
  - Alarm
  - Tamper
  - Fault
- DoorType
- Timings
  - ReleaseTime
  - OpenTime
  - ExtendedReleaseTime
  - DelayTimeBeforeRelock
  - ExtendedOpenTime
  - PreAlarmTime

Note: The following fields are compared at steps 15 and 17:

- DoorInfo:
  - token
  - Name
  - Description
  - Capabilities
    - Access
    - AccessTimingOverride
    - Lock
    - Unlock
    - Block
    - DoubleLock
    - LockDown
    - LockOpen
    - DoorMonitor
    - LockMonitor
    - DoubleLockMonitor
    - Alarm
    - Tamper
    - Fault

4.7.15 CREATE NEW DOOR WITH SET DOOR (DOOR CAPABILITIES FALSE)

Test Case ID: DOORCONTROL-8-1-15

Specification Coverage: DoorInfo (ONVIF Door Control Service Specification), Door (ONVIF Door Control Service Specification), SetDoor command (ONVIF Door Control Service Specification)

Feature Under Test: SetDoor

WSDL Reference: door.wsdl, event.wsdl

Test Purpose: To verify creation of door using SetDoor command and generating of appropriate notifications.

Pre-Requisite: Door Control Service is received from the DUT. Event Service was received from the DUT. Device supports Pull-Point Notification feature. Client Supplied Token is supported by the DUT as indicated by ClientSuppliedTokenSupported capability. The DUT shall have enough free storage capacity for one additional Door. Door Entity is supported by the DUT as indicated by MaxDoors > 0 capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves Door Control Service Capabilities by following the procedure mentioned in Annex A.7 with the following input and output parameters
   - out *cap* - Door Control Service capabilities
4. ONVIF Client retrieves complete list of doors by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - out *initialDoorCompleteList* - door complete list
5. ONVIF Client checks free storage for additional door by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - in *initialDoorCompleteList* - door complete list
   - out *doorToRestore* - removed door
6. Set *doorToken* := token that differs from tokens listed in *initialDoorCompleteList*.
7. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.13 with the following input and output parameters
   - in **tns1:Configuration/Door/Changed** - Notification Topic
   - out *s* - Subscription reference
   - out *currentTime* - current time for the DUT

- out *terminationTime* - Subscription termination time

8. ONVIF client invokes **SetDoor** request with parameters
   - Door.token := *doorToken*
   - Door.Name := "Test Name"
   - Door.Description := "Test Description"
   - Door.Capabilities.Access := false
   - Door.Capabilities.AccessTimingOverride := false
   - Door.Capabilities.Lock := false
   - Door.Capabilities.Unlock := false
   - Door.Capabilities.Block := false
   - Door.Capabilities.DoubleLock := false
   - Door.Capabilities.LockDown := false
   - Door.Capabilities.LockOpen := false
   - Door.Capabilities.DoorMonitor := false
   - Door.Capabilities.LockMonitor := false
   - Door.Capabilities.DoubleLockMonitor := false
   - Door.Capabilities.Alarm := false
   - Door.Capabilities.Tamper := false
   - Door.Capabilities.Fault := false
   - Door.DoorType := "pt:ManTrap"
   - Door.Timings.ReleaseTime := PT60S
   - Door.Timings.OpenTime := PT120S
   - Door.Timings.ExtendedReleaseTime skipped
   - Door.Timings.DelayTimeBeforeRelock skipped

- Door.Timings.ExtendedOpenTime skipped
- Door.Timings.PreAlarmTime skipped

9. The DUT responds with **SetDoorResponse** message.
10. ONVIF Client retrieves and checks **tns1:Configuration/Door/Changed** event for the specified Door token by following the procedure mentioned in Annex A.15 with the following input and output parameters
    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *doorToken* - Door token
11. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.14 with the following input and output parameters
    - in *s* - Subscription reference
12. ONVIF Client retrieves a door by following the procedure mentioned in Annex A.10 with the following input and output parameters
    - in *doorToken* - door token
    - out *doorList* - the list of doors
13. If *doorList*[0] item does not have equal field values to values from step 8, FAIL the test, restore the DUT state, and skip other steps.
14. ONVIF Client retrieves a door info by following the procedure mentioned in Annex A.11 with the following input and output parameters
    - in *doorToken* - door token
    - out *doorInfoList* - the list of doors info
15. If *doorInfoList*[0] item does not have equal field values to values from step 8, FAIL the test, restore the DUT state, and skip other steps.
16. ONVIF Client retrieves complete list of doors info by following the procedure mentioned in Annex A.1 with the following input and output parameters
    - out *doorInfoCompleteList* - door info complete list

17. If *doorInfoCompleteList* does not have DoorInfo[token = *doorToken*] item with equal field values to values from step 8, FAIL the test, restore the DUT state, and skip other steps.
18. ONVIF Client retrieves complete list of doors by following the procedure mentioned in Annex A.6 with the following input and output parameters
    - out *doorCompleteList* - door complete list
19. If *doorCompleteList* does not have Door[token = *doorToken*] item with equal field values to values from step 8, FAIL the test, restore the DUT state, and skip other steps.
20. ONVIF Client deletes the Door by following the procedure mentioned in Annex A.9 with the following input and output parameters
    - in *doorToken* - door token
21. ONVIF Client restores door deleted at step 5 if any.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Note: The following fields are compared at steps 13 and 19:

- Door:
  - token
  - Name
  - Description
  - Capabilities
    - Access
    - AccessTimingOverride
    - Lock
    - Unlock
    - Block

- DoubleLock
- LockDown
- LockOpen
- DoorMonitor
- LockMonitor
- DoubleLockMonitor
- Alarm
- Tamper
- Fault

- DoorType
- Timings
  - ReleaseTime
  - OpenTime
  - ExtendedReleaseTime
  - DelayTimeBeforeRelock
  - ExtendedOpenTime
  - PreAlarmTime

Note: The following fields are compared at steps 15 and 17:

- DoorInfo:
  - token
  - Name
  - Description
  - Capabilities
    - Access

- AccessTimingOverride
- Lock
- Unlock
- Block
- DoubleLock
- LockDown
- LockOpen
- DoorMonitor
- LockMonitor
- DoubleLockMonitor
- Alarm
- Tamper
- Fault

4.7.16 MODIFY DOOR WITH SET DOOR

Test Case ID: DOORCONTROL-8-1-16

Specification Coverage: DoorInfo (ONVIF Door Control Service Specification), Door (ONVIF Door Control Service Specification), SetDoor command (ONVIF Door Control Service Specification)

Feature Under Test: SetDoor

WSDL Reference: door.wsdl, event.wsdl

Test Purpose: To verify modifying of door using SetDoor command and generating of door changed notifications.

Pre-Requisite: Door Control Service is received from the DUT. Event Service was received from the DUT. Device supports Pull-Point Notification feature. The DUT shall have enough free storage capacity for one additional Door. Client Supplied Token is supported by the DUT as indicated by ClientSuppliedTokenSupported capability. Door Entity is supported by the DUT as indicated by MaxDoors > 0 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 by following the procedure mentioned in Annex A.7 with the following output parameters
   - out *cap* - door control capabilities
4. ONVIF Client retrieves a complete list of doors by following the procedure mentioned in Annex A.6 with the following output parameters
   - out *doorCompleteList1* - complete door list
5. ONVIF Client checks free storage for additional Door by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - in *doorCompleteList1* - complete door list
   - out *doorToRestore* - deleted door
6. ONVIF Client creates door by following the procedure mentioned in Annex A.12 with the following input and output parameters
   - in false - for all door capabilities
   - in "pt:Door" - Door Type
   - in PT60S - ReleaseTime
   - in PT120S - OpenTime
7. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.13 with the following input and output parameters
   - in **tns1:Configuration/Door/Changed** - Notification Topic
   - out *s* - Subscription reference
   - out *currentTime* - current time for the DUT
   - out *terminationTime* - Subscription termination time
8. ONVIF client invokes **SetDoor** with parameters
   - Door.Token := *doorToken*

- Door.Name := "Test Name 2"
- Door.Description := "Test Description 2"
- Door.Capabilities.Access := true
- Door.Capabilities.AccessTimingOverride := true
- Door.Capabilities.Lock := true
- Door.Capabilities.Unlock := true
- Door.Capabilities.Block := true
- Door.Capabilities.DoubleLock := true
- Door.Capabilities.LockDown := true
- Door.Capabilities.LockOpen := true
- Door.Capabilities.DoorMonitor := true
- Door.Capabilities.LockMonitor := true
- Door.Capabilities.DoubleLockMonitor := true
- Door.Capabilities.Alarm := true
- Door.Capabilities.Tamper := true
- Door.Capabilities.Fault := true
- Door.DoorType := "pt:Door"
- Door.Timings.ReleaseTime := PT30S
- Door.Timings.OpenTime := PT60S
- Door.Timings.ExtendedReleaseTime := PT10S
- Door.Timings.DelayTimeBeforeRelock := PT10S
- Door.Timings.ExtendedOpenTime := PT10S
- Door.Timings.PreAlarmTime := PT10S

9. The DUT responds with empty **SetDoorResponse** message.

10. ONVIF Client retrieves and checks **tns1:Configuration/Door/Changed** event for the specified Door token by following the procedure mentioned in Annex A.15 with the following input and output parameters
    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *doorToken* - Door token
11. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.14 with the following input and output parameters
    - in *s* - Subscription reference
12. ONVIF Client retrieves a door by following the procedure mentioned in Annex A.10 with the following input and output parameters
    - in *doorToken* - door token
    - out *doorList* - door list
13. If *doorList*[0] item does not have equal field values to values from step 8, FAIL the test and go step 20.
14. ONVIF Client retrieves a door info by following the procedure mentioned in Annex A.11 with the following input and output parameters
    - in *doorToken* - door token
    - out *doorInfoList* - door info list
15. If *doorInfoList*[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 door information list by following the procedure mentioned in Annex A.1 with the following input and output parameters
    - out *doorInfoCompleteList* - door info list
17. If *doorInfoCompleteList* does not have *doorInfo*.[token:= *doorToken*] 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 doors by following the procedure mentioned in Annex A.6 with the following input and output parameters

- out *doorCompleteList2* - door list

19. If *doorCompleteList2* does not have *door*.[token:= *doorToken*] item with equal field values to values from step 8, FAIL the test and go step 20.
20. ONVIF Client deletes the Door by following the procedure mentioned in Annex A.9 to restore DUT configuration with the following input and output parameters
    - in *doorToken* - door token
21. ONVIF Client restores door deleted at step 5 if any.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

- Door:
  - token
  - Name
  - Description
  - Capabilities
    - Access
    - AccessTimingOverride
    - Lock
    - Unlock
    - Block
    - DoubleLock
    - LockDown
    - LockOpen

- DoorMonitor
- LockMonitor
- DoubleLockMonitor
- Alarm
- Tamper
- Fault

- DoorType
- Timings
  - ReleaseTime
  - OpenTime
  - ExtendedReleaseTime
  - DelayTimeBeforeRelock
  - ExtendedOpenTime
  - PreAlarmTime

Note: The following fields are compared at steps 13, 15:

- DoorInfo:
  - token
  - Name
  - Description
  - Capabilities
    - Access
    - AccessTimingOverride
    - Lock
    - Unlock

- Block
- DoubleLock
- LockDown
- LockOpen
- DoorMonitor
- LockMonitor
- DoubleLockMonitor
- Alarm
- Tamper
- Fault

4.7.17 SET DOOR - EMPTY DOOR TOKEN

Test Case ID: DOORCONTROL-8-1-17

Specification Coverage: SetDoor command (ONVIF Door Control Service Specification)

Feature Under Test: SetDoor

WSDL Reference: door.wsdl, event.wsdl

Test Purpose: To verify SetDoor command with empty door token.

Pre-Requisite: Door Control Service is received from the DUT. Client Supplied Token is supported by the DUT as indicated by ClientSuppliedTokenSupported capability. The DUT shall have enough free storage capacity for one additional Door.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves Door Control Service Capabilities by following the procedure mentioned in Annex A.7 with the following input and output parameters
   - out *cap* - Door Control Service capabilities

4. ONVIF Client retrieves complete list of doors by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - out *initialDoorCompleteList* - door complete list
5. ONVIF Client checks free storage for additional door by following the procedure mentioned in Annex A.8 with the following input and output parameters
   - in *initialDoorCompleteList* - door complete list
   - out *doorToRestore* - removed door
6. ONVIF client invokes **SetDoor** request with parameters
   - Door.token := ""
   - Door.Name := "Test Name"
   - Door.Description := "Test Description"
   - Door.Capabilities.Access := true
   - Door.Capabilities.AccessTimingOverride := true
   - Door.Capabilities.Lock := true
   - Door.Capabilities.Unlock := true
   - Door.Capabilities.Block := true
   - Door.Capabilities.DoubleLock := true
   - Door.Capabilities.LockDown := true
   - Door.Capabilities.LockOpen := true
   - Door.Capabilities.DoorMonitor := true
   - Door.Capabilities.LockMonitor := true
   - Door.Capabilities.DoubleLockMonitor := true
   - Door.Capabilities.Alarm := true
   - Door.Capabilities.Tamper := true
   - Door.Capabilities.Fault := true
   - Door.DoorType := "pt:Door"

- Door.Timings.ReleaseTime := PT60S
- Door.Timings.OpenTime := PT120S
- Door.Timings.ExtendedReleaseTime := PT30S
- Door.Timings.DelayTimeBeforeRelock := PT30S
- Door.Timings.ExtendedOpenTime := PT30S
- Door.Timings.PreAlarmTime := PT10S

7. The DUT responds with **env:Sender\ter:InvalidArgVal** SOAP 1.2 fault.
8. ONVIF Client restores door deleted at step 5 if any.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Annex A Helper Procedures and Additional Notes

A.1 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.2 Get Complete Access Point Info List

The following algorithm will be used to get the complete list of Access Points:

1. ONVIF Client will invoke **GetAccessPointInfoList** request (no Limit, no StartReference) to retrieve the first part of Access Point Information list from the DUT.
2. Verify the **GetAccessPointInfoListResponse** message from the DUT.
3. If **GetAccessPointInfoListResponse** message contains NextStartReference, repeat steps 1-2 with StartReference = [current NextStartReference]. Otherwise, skip other steps and finalize getting complete access point list.

The complete ordered list of access points with information will be made by the means of uniting all **GetAccessPointInfoListResponse** messages. Also total number of access points will be calculated.

A.3 Catching of Initialized event

According to ONVIF-Core-Specification (9.4 Properties Item) device shall provide notifications informing the client of all objects with the requested property, which are alive at the time of the subscription. Therefore, after **CreatePullPointSubscription** request in DOORCONTROL-3-1-28 – DOORCONTROL-3-1-36 tests device shall generate Initialized events with current Door State for all existing doors.

ONVIF Client will follow the following logic to catch Initialized event for required Door token:

1. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
2. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. If no NotificationMessage for selected Door is received and Operation Delay time does not expire, ONVIF Client repeats step 1.
3. Verify that the DUT sends Notification message for selected Door with current state (Notifications for other Doors will be ignored for this test case).
4. Verify received Notification message (correct value for UTC time, TopicExpression and wsnt:Message).
5. Verify that PropertyOperation = "Initialized".
6. Verify that TopicExpression is equal to tns1:Door/State/DoorMode.
7. Verify that notification contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to selected Door Token.
8. Verify that notification contains Data.SimpleItem item with Name = «State» and Value with type is equal to tdc:DoorMode.

FAIL –

- The DUT did not send a valid Notification message that contains a property event tns1:Door/State/DoorMode for selected Door during Operation delay time (correct value for UTC time, TopicExpression and wsnt:Message, PropertyOperation = "Initialized", Topic = “tns1:Door/State/DoorMode”, contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to selected Door Token, contains Data.SimpleItem item with Name = «State» and Value with type equal to tdc:DoorMode).

Note: Notifications for other Doors than selected will be ignored.

Note: Test will be failed, if no required Notification messages are received at step 3 during a certain period of time (Operation Delay should be used in ONVIF Device Test Tool).

A.4 User Interaction Process

1. ONVIF Client sends **Unsubscribe** request to end current subscription if subscription manager has been created during test.
2. Verify **UnsubscribeResponse** message.

3. User Interaction window appears.
4. Test Operator puts Door into the appropriate state. At that state the command (used in the test), which changes the door state, shall be accepted by the DUT.
5. Returns to the test and use DoorToken from user Interaction window as token of selected Door.

A.5 Door's transition to Locked state

ONVIF Client will try to move the selected for the test door into 'Locked' state according to the following logics:

1. ONVIF Client will invoke **GetDoorState** request (Token = "Token1", where Token1 is selected door) to retrieve current state of the door.
2. Verify **GetDoorStateResponse** from the DUT.
3. If DoorState.DoorMode is equal to Locked then skip other steps and returns to the test.
4. ONVIF Client will invoke **CreatePullPointSubscription** request with tns1:Door/State/DoorMode Topic as Filter and an InitialTerminationTime of timeout1.
5. Verify **CreatePullPointSubscriptionResponse** message from the DUT.
6. If DoorState.DoorMode is equal to Unlocked then ONVIF Client will invoke **LockDoor** request (Token = "Token1") to switch the door into Locked state. Otherwise, go to step 12.
7. Verify **LockDoorResponse** message from the DUT.
8. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
9. Verify that the DUT sends a **PullMessagesResponse** request. Repeat step 8 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).
10. Check that the last Notification message with PropertyOperation = "Changed" contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to Token1 and contains Data.SimpleItem item with Name = «State» and Value equal to "Locked".
11. ONVIF Client deletes Subscription Manager either by calling **Unsubscribe** or through a timeout and returns to the test.
12. If DoorState.DoorMode is equal to Accessed then ONVIF Client will invoke LockDoor request (Token = "Token1") to switch the door into Locked state. Otherwise, go to step 18.

13. Verify **LockDoorResponse** message from the DUT.
14. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
15. Verify that the DUT sends a **PullMessagesResponse**. Repeat step 14 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).
16. Check that the last Notification message with PropertyOperation = "Changed" contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to Token1 and contains Data.SimpleItem item with Name = «State» and Value equal to “Locked”.
17. ONVIF Client deletes Subscription Manager either by calling **unsubscribe** or through a timeout and returns to the test.
18. If DoorState.DoorMode is equal to LockedDown then ONVIF Client will invoke **LockDownReleaseDoor** request (Token = “Token1”) to switch the door into Locked state. Otherwise, go to step 24.
19. Verify **LockDownReleaseDoorResponse** message from the DUT.
20. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
21. Verify that the DUT sends a **PullMessagesResponse**. Repeat step 20 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).
22. Check that the last Notification message with PropertyOperation = "Changed" contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to Token1 and contains Data.SimpleItem item with Name = «State» and Value equal to “Locked”.
23. ONVIF Client deletes Subscription Manager either by calling **unsubscribe** or through a timeout and returns to the test.
24. If DoorState.DoorMode is equal to LockedOpen then ONVIF Client will invoke **LockOpenReleaseDoor** request (Token = “Token1”) to move the door into Unlocked state. Otherwise go to step 35.
25. Verify **LockOpenReleaseDoorResponse** message from the DUT.
26. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
27. Verify that the DUT sends a **PullMessagesResponse**. Repeat step 26 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).

28. Check that the last Notification message with PropertyOperation = "Changed" contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to Token1 and contains Data.SimpleItem item with Name = «State» and Value equal to “Unlocked”.

29. ONVIF Client will invoke **LockDoor** request (Token = “Token1”) to switch the door into Locked state.

30. Verify **LockDoorResponse** message from the DUT.

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

32. Verify that the DUT sends a **PullMessagesResponse**. Repeat step 31 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).

33. Check that the last Notification message with PropertyOperation = "Changed" contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to Token1 and contains Data.SimpleItem item with Name = «State» and Value equal to “Locked”.

34. ONVIF Client deletes Subscription Manager either by calling **unsubscribe** or through a timeout and returns to the test.

35. If DoorState.DoorMode is equal to DoubleLocked then ONVIF Client will invoke **LockDoor** request (Token = “Token1”) to move the door into Locked state. Otherwise, go to step 41.

36. Verify **LockDoorResponse** message from the DUT.

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

38. Verify that the DUT sends a **PullMessagesResponse**. Repeat step 37 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).

39. Check that the last Notification message with PropertyOperation = "Changed" contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to Token1 and contains Data.SimpleItem item with Name = «State» and Value equal to “Locked”.

40. ONVIF Client deletes Subscription Manager either by calling **unsubscribe** or through a timeout and returns to the test.

41. If DoorState.DoorMode is equal to Blocked then ONVIF Client will invoke LockDoor request (Token = “Token1”) to switch the door into Locked state. Otherwise, skip other steps and run user interaction process.

42. Verify **LockDoorResponse** message from the DUT.

43. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
44. Verify that the DUT sends a **PullMessagesResponse**. Repeat step 14 until timeout expires (Operation Delay option from ONVIF Device Test Tool will be used).
45. Check that the last Notification message with PropertyOperation = "Changed" contains Source.SimpleItem item with Name = "DoorToken" and Value is equal to Token1 and contains Data.SimpleItem item with Name = «State» and Value equal to “Locked”.
46. ONVIF Client deletes Subscription Manager either by calling **unsubscribe** or through a timeout and returns to the test.

FAIL –

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT sent the last Notification message with PropertyOperation = "Changed" with State value not equal to expected state at step 9 or 15 or 32 or 38.
- The DUT sent an invalid Notification messages with PropertyOperation = "Changed" (correct value for UTC time, TopicExpression and wsnt:Message, Topic = “tns1:Door/State/DoorMode”, contains Source.SimpleItem item with Name = "DoorToken" and Value equal to selected Door Token, contains Data.SimpleItem item with Name = «State» and Value with type equal to tdc:DoorMode).

Note: If the DUT sends SOAP fault to request to change door mode, then the ONVIF Client starts user interaction process (see Annex A.4).

Note: If DUT sends the last notification message with PropertyOperation = "Changed" with door state different from expected at step 21 or at step 27, then the ONVIF Client returns to the test and starts user interaction process (see Annex A.4)

A.6 Get Door List

Name: HelperGetDoorList

Procedure Purpose: Helper procedure to get complete doors list.

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

Input: None.

Returns: The complete list of doors (*doorCompleteList*).

Procedure:

1. ONVIF client invokes **GetDoorList** with parameters
   - Limit skipped
   - StartReference skipped
2. The DUT responds with **GetDoorListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - Door list =: *doorCompleteList*
3. Until *nextStartReference* is not null, repeat the following steps:
   3.1. ONVIF client invokes **GetDoorList** with parameters
      - Limit skipped
      - StartReference := *nextStartReference*
   3.2. The DUT responds with **GetDoorListResponse** message with parameters
      - NextStartReference =: *nextStartReference*
      - Door list =: *doorListPart*
   3.3. Set the following:
      - *doorCompleteList* := *doorCompleteList* + *doorListPart*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **GetDoorListResponse** message

A.7 Get Service Capabilities

Name: HelperGetServiceCapabilities

Procedure Purpose: Helper procedure to get 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.8 Free Storage for Additional Door

Name: HelperCheckFreeStorageForDoor

Procedure Purpose: Helper procedure to provide possibility to add a door.

Pre-requisite: Door Control Service is received from the DUT. Door Management is supported by the DUT as indicated by DoorManagementSupported capability.

Input: The complete list of doors (*doorCompleteList*).

Returns: Removed door (*doorToRestore*).

Procedure:

1. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.7.
2. If *cap*.MaxDoors is skipped set *cap*.MaxDoors := 10.
3. If number of items of *doorCompleteList* less than cap.MaxDoor, skip other steps.
4. ONVIF Client find a door to delete by following the procedure mentioned in Annex A.17 with the following input parameters
   - in *doorCompleteList* - door list.
   - out *doorToRestore* - deleted door to restore.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- None.

A.9 Delete Door

Name: HelperDeleteDoor

Procedure Purpose: Helper procedure to delete 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 Token (*doorToken*).

Returns: None.

Procedure:

1. ONVIF Client invokes **DeleteDoor** request with parameters
   - Token =: *doorToken*
2. The DUT responds with **DeleteDoorResponse** message.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **DeleteDoorResponse** message

A.10 Get Door

Name: HelperGetDoor

Procedure Purpose: Helper procedure to get 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 Token (*doorToken*).

Returns: Door List (*doorList*).

Procedure:

1. ONVIF Client invokes **GetDoors** with parameters
   - Token[0] := *doorToken*
2. The DUT responds with **GetDoorsResponse** message with parameters:
   - Door list =: *doorList*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.11 Get Door Info

Name: HelperGetDoorInfo

Procedure Purpose: Helper procedure to get door info.

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

Input: Door Token (*doorToken*).

Returns: Door Info List (*doorInfoList*).

Procedure:

1. ONVIF Client invokes **GetDoorInfo** with parameters
   - Token[0] := *doorToken*
2. The DUT sends the **GetDoorInfoResponse** message with parameters
   - DoorInfo =: *doorInfoList*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **GetDoorInfoResponse** message

A.12 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*), *doorType*, *doorTimings*.

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 := *doorTimings*.ReleaseTime
- Door.Timings.OpenTime := *doorTimings*.OpenTime
- Door.Timings.ExtendedReleaseTime := *doorTimings*.ExtendedReleaseTime
- Door.Timings.DelayTimeBeforeRelock := *doorTimings*.DelayTimeBeforeRelock
- Door.Timings.ExtendedOpenTime := *doorTimings*.ExtendedOpenTime
- Door.Timings.PreAlarmTime := *doorTimings*.PreAlarmTime

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.13 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.14 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.15 Retrieve Door Changed Event by PullPoint

Name: HelperPullDoorChanged

Procedure Purpose: Helper procedure to retrieve and check tns1:Configuration/Door/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 Door token (*doorToken*).

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/Door/Changed", FAIL the test, restore the DUT state, and skip other steps.

1.5. If *notificationMessageList* is not empty and the DoorToken source simple item in *notificationMessageList* is equal to *doorToken*, 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 *doorToken*, 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.16 Retrieve Door Removed Event by PullPoint

Name: HelperPullDoorRemoved

Procedure Purpose: Helper procedure to retrieve and check tns1:Configuration/Door/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 Door token (*doorToken*).

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/Door/Removed", FAIL the test, restore the DUT state, and skip other steps.

1.5. If *notificationMessageList* is not empty and the DoorToken source simple item in *notificationMessageList* is equal to *doorToken*, 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 *doorToken*, 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.17 Find Door To Delete

Name: HelperFindDoorToDelete

Procedure Purpose: Helper procedure to find a door in the door list that can be deleted.

Pre-requisite: Door Control Service is received from the DUT. Door Management is supported by the DUT as indicated by DoorManagementSupported capability.

Input: Door list (*doorList*).

Returns: Deleted Door (*deletedDoor*).

Procedure:

1. For each Door *door* from *doorList* repeat the following steps:

   1.1. ONVIF Client invokes **DeleteDoor** request with parameters

   - Token =: *door*.@token

   1.2. The DUT responds with SOAP fault message or with **DeleteDoorResponse** message.

   1.3. If DUT returns **DeleteDoorResponse** message, set *deletedDoor* := *door* and skip other steps.

2. Fail the test.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **DeleteDoorResponse** message or SOAP fault message.