Driving IP-based physical security through global standardization

ONVIF™

ONVIF Device IO Test Specification

Version 16.07

August 8, 2016

Driving IP-based physical security through global standardization

Driving IP-based physical security through global standardization

Revision History

Ver.	Date	Description
16.06	March 16, 2016	Original publication
16.06	March 30, 2016	The tests DEVICEIO-3-1-1 - DEVICEIO-3-1-4 have been added.
16.06	April 6, 2016	The DEVICEIO-1-2-1 - DEVICEIO-1-2-4 and DEVICEIO-3-1-4 have been updated according to feedback from Sano Hiroyuki.
16.06	April 11, 2016	Implemented comments from Sano Hiroyuki about redundant event-testing steps. Implemented comments of Bhetanabottla Sriram from Canon.
16.07	June 22, 2016	Minor spellcheck and version number correction
16.07	July 13, 2016	Tests sequences updated.
16.07	July 27, 2016	Implemented comments from Canon and Sony
16.07	August 5, 2016	Bugfixes based on comments from Hiroyuki Sano

Driving IP-based physical security through global standardization

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

This ONVIF Device IO Test Specification acts as a supplementary document to the [ONVIF DeviceIO Service 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 Device IO Test Specification defines and regulates the conformance testing procedure for the ONVIF conformant devices. Conformance testing is meant to be functional black-box testing. The objective of this specification is to provide test cases to test individual requirements of ONVIF devices according to ONVIF Device IO Service which is defined in [ONVIF DeviceIO Service Specs].

The principal intended purposes are:

1. Provide self-assessment tool for implementations.
2. Provide comprehensive test suite coverage for [ONVIF DeviceIO Service 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 DeviceIO Service Specs]; instead it would cover subset of it. The scope of this specification is to derive all the normative requirements of [ONVIF DeviceIO Service Specs] which are related to ONVIF Device IO Service and some of the optional requirements.

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

1.1.1 Relay Outputs

Relay Outputs section covers the test cases needed for the verification of Relay Outputs service features as mentioned in [ONVIF DeviceIO Service Specs]. The DeviceIO service is used to retrieve and configure the settings of physical outputs of a device.

Briefly it covers the following things:

1. Manage Relay Output Configuration
2. Change Relay Output State

1.1.2 Digital Inputs

Digital Inputs section covers the test cases needed for the verification of Digital Inputs service features as mentioned in [ONVIF DeviceIO Service Specs]. The DeviceIO service is used to retrieve and configure the settings of physical inputs of a device.

Briefly it covers the following thing.

1. Configure Digital Input idle state.

2 Terms and Definitions

2.1 Definitions

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

Relay Output physical outputs of a device.

Digital Input physical inputs of a device

3 Test Overview

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

3.1 Test Setup

3.1.1 Network Configuration for DUT

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

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

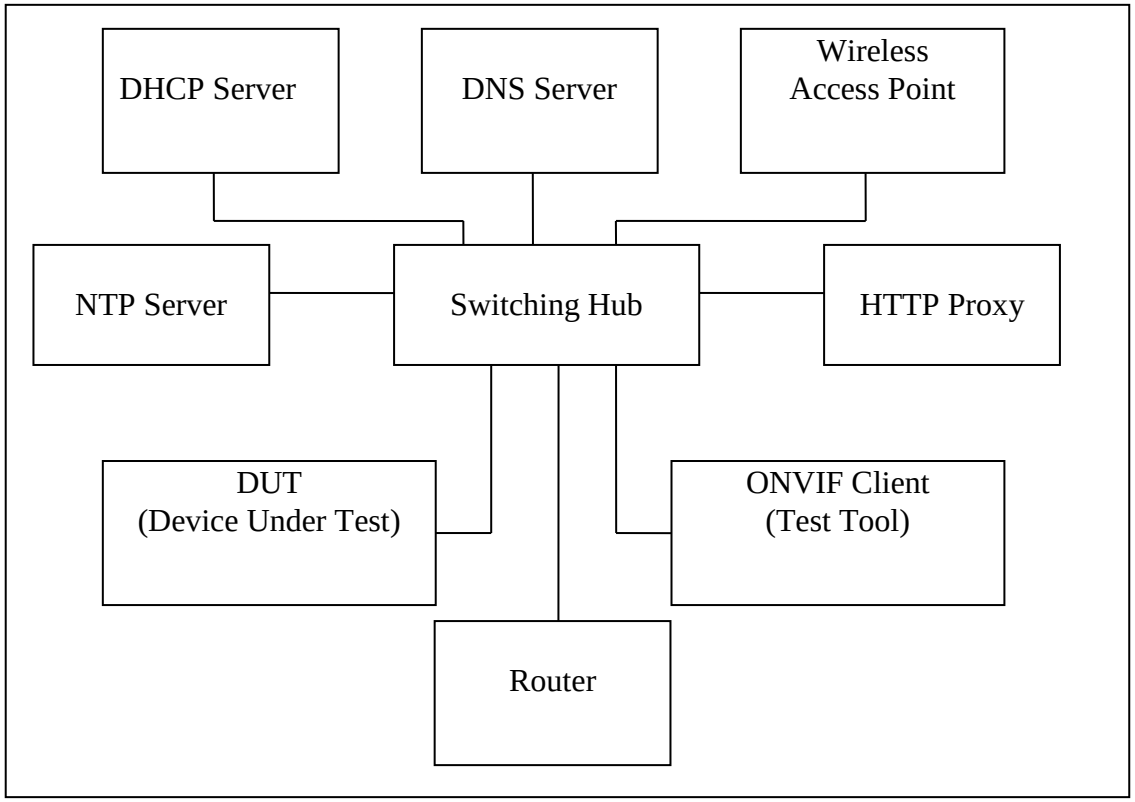


Figure 1: Test Configuration for DUT

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 equipment in the test environment. All devices should be connected to the Switching Hub.

Router: provides router advertisements for IPv6 configuration.

3.2 Prerequisites

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

1. The DUT shall be configured with an IPv4 address.
2. The DUT shall be IP reachable [in the test configuration].
3. The DUT shall be able to be discovered by the Test Tool.
4. The DUT shall be configured with the time i.e. manual configuration of UTC time and if NTP is supported by DUT then NTP time shall be synchronized with NTP Server.
5. The DUT time and Test tool time shall be synchronized with each other either manually or by common NTP server.

3.3 Test Policy

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

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

3.3.1 Relay Output

DUT should respond with proper response message for all SOAP actions. Sending fault messages such as "ter:ConfigurationConflict" will be treated as FAILURE of the test cases.

Please refer to Section 4 for Relay Output Test Cases.

3.3.2 Digital Input

DUT should respond with proper response message for all SOAP actions. Sending fault messages such as “ter:ConfigurationConflict” will be treated as FAILURE of the test cases.

Please refer to Section 5 for Digital Input Test Cases

Driving IP-based physical security through global standardization

4 Relay Output Test Cases

4.1 Manage Relay OutputSettings

4.1.1 IO GETRELAYOUTPUTS

Test Label: Device IO DUT Command GetRelayOutputs Test.

Test Case ID: DEVICEIO-1-1-1

ONVIF Core Specification Coverage: None

Command Under Test: GetRelayOutputs

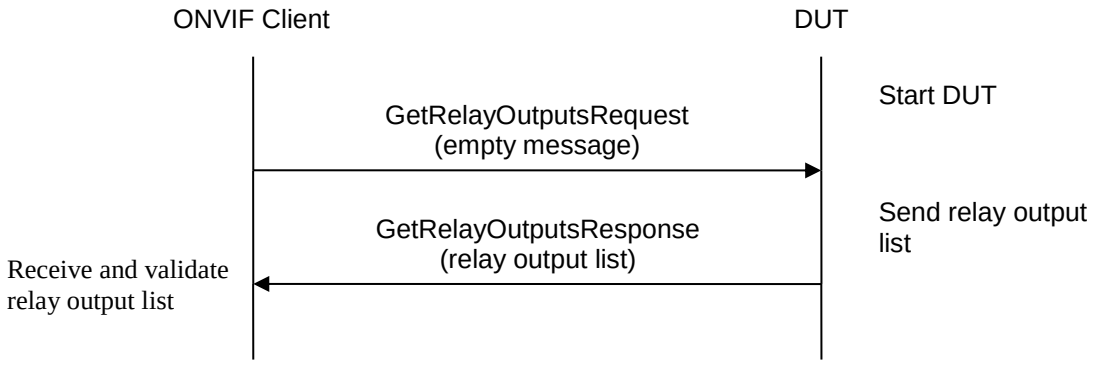
WSDL Reference: deviceio.wsdl

Test Purpose: To retrieve DUT relay outputs using GetRelayOutputs command.

Pre-Requisite: Device IO service is supported by DUT. Relay Outputs supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetRelayOutputsRequest message to retrieve relay outputs supported by the DUT.
4. Verify the GetRelayOutputsResponse message from the DUT.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send GetRelayOutputsResponse message.

The DUT did not send valid GetRelayOutputsResponse message.

The DUT sent at least two RelayOutputs with the same token.

4.1.2 IO GETRELAYOUTPUTS – VERIFY QUANTITY

Test Label: Device IO Get Relay Outputs – Verify Quantity

Test Case ID: DEVICEIO-1-1-2

ONVIF Core Specification Coverage: None

Command Under Test: GetRelayOutputs, GetServiceCapabilities

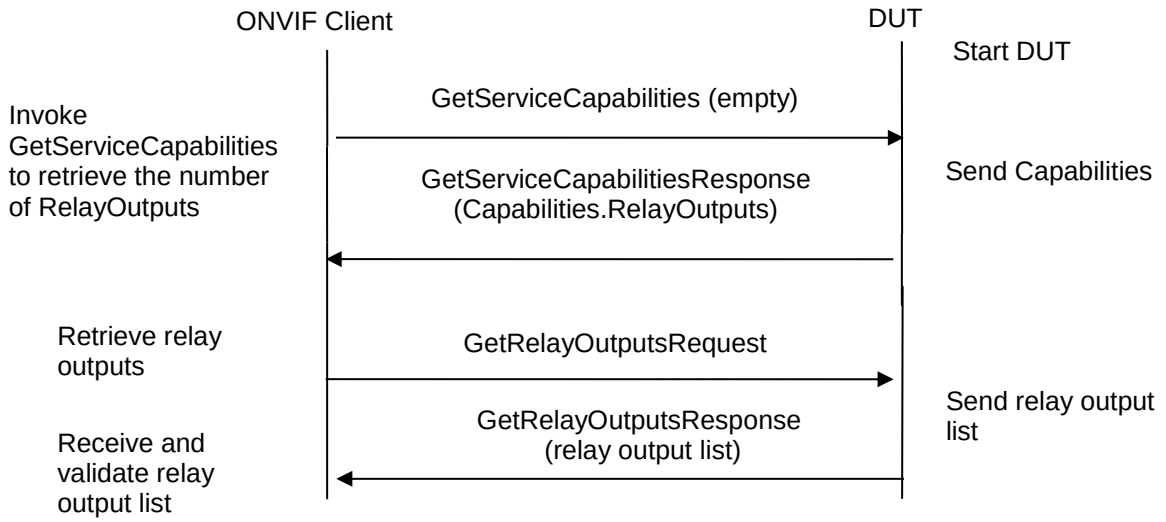
WSDL Reference: deviceio.wsdl

Test Purpose: To verify the number of Relay outputs from GetRelayOutputsResponse message.

Pre-Requisite: Device IO service is supported by DUT. Relay Outputs supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

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

3. ONVIF Client invokes GetServiceCapabilitiesRequest.
4. DUT sends GetServiceCapabilitiesResponse. ONVIF Client verifies the response.
5. ONVIF Client invokes GetRelayOutputsRequest message to retrieve relay outputs supported by the DUT.
6. DUT sends GetRelayOutputsResponse with a list of relay outputs supported.
7. Verify the GetRelayOutputsResponse message from the DUT.
8. Verify the number of Relay Outputs in GetRelayOutputsResponse. This number should be equal to the Capabilities.RelayOutputs number in GetServiceCapabilitiesResponse.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send GetServiceCapabilitiesResponse message.

The DUT did not send valid GetServiceCapabilitiesResponse message.

The DUT did not send GetRelayOutputsResponse message.

The DUT did not send valid GetRelayOutputsResponse message.

The number of Relay Outputs in GetRelayOutputsResponse message is not equal to Device.IO.RelayOutputs number from GetServiceCapabilitiesResponse message.

4.1.3 IO GETRELAYOUTPUTOPTIONS

Test Label: Device IO GetRelayOutputOptions Test.

Test Case ID: DEVICEIO-1-1-3

ONVIF Core Specification Coverage: None

Command Under Test: GetRelayOutputs, GetRelayOutputOptions

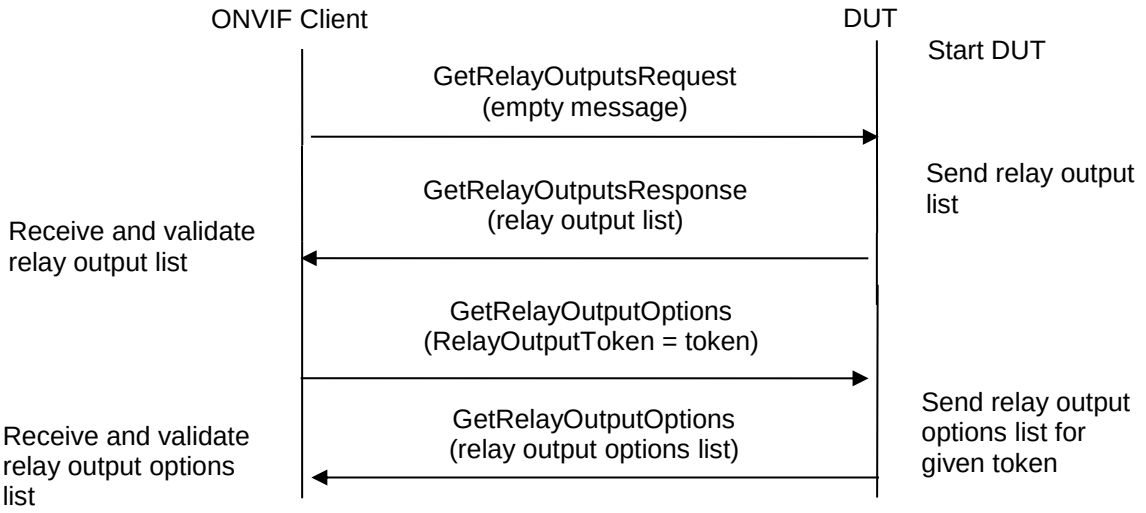
WSDL Reference: deviceio.wsdl

Test Purpose: To verify the behavior of GetRelayOutputOptions command.

Pre-Requisite: Device IO service is supported by DUT. Relay Outputs supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetRelayOutputsRequest message to retrieve a list of all available relay outputs and their settings
4. The DUT sends the GetRelayOutputsResponse message with list of all available relay outputs and their settings.
5. ONVIF Client verifies the GetRelayOutputsResponse message from the DUT.
6. ONVIF Client selects first relay output from GetRelayOutputsResponse, saves this relay output in **RelayOutput1** variable. Then it runs the following steps:

   6.1. ONVIF Client invokes GetRelayOutputOptionsRequest message RelayOutputToken=**RelayOutput1** token as input parameter.

   6.2. The DUT sends GetRelayOutputOptionsResponse.

   6.3. ONVIF client verifies the GetRelayOutputOptionsResponse message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetRelayOutputsResponse message.

The DUT did not send valid GetRelayOutputsResponse message.

The DUT sent an empty list of RelayOutputs in GetRelayOutputsResponse.

The DUT did not send GetRelayOutputOptionsResponse message.

The DUT did not send valid GetRelayOutputOptionsResponse message.

4.1.4 IO SETRELAYOUTPUTSETTINGS

Test Label: Device IO SetRelayOutputSettings Test.

Test Case ID: DEVICEIO-1-1-4

ONVIF Core Specification Coverage: None

Command Under Test: GetRelayOutputs, SetRelayOutputSettings, GetRelayOutputOptions

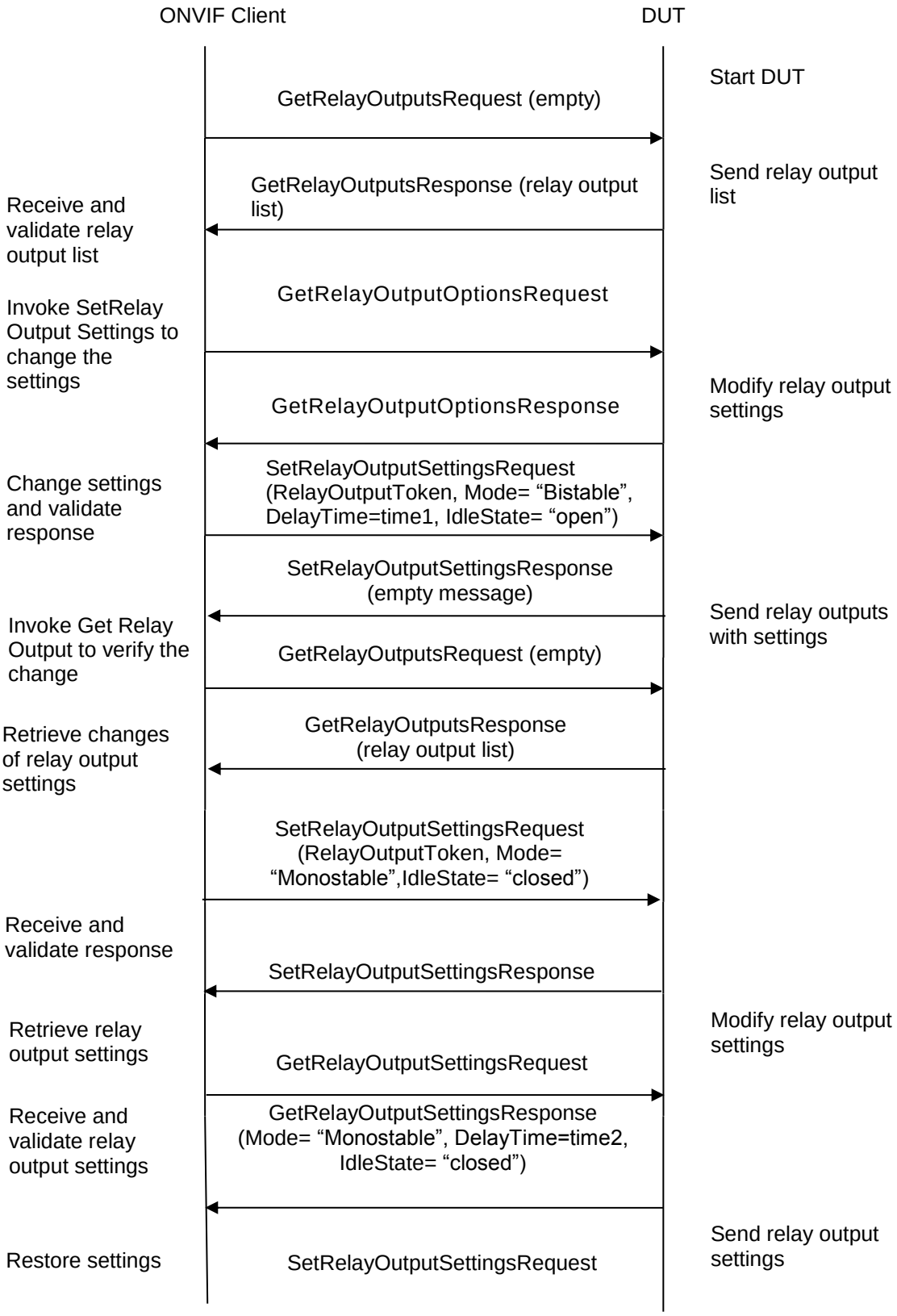
WSDL Reference: deviceio.wsdl

Test Purpose: To verify the behavior of SetRelayOutputSettings command.

Pre-Requisite: Device IO service is supported by DUT. Relay Outputs supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
GetRelayOutputsRequest (empty)
Send relay output list
GetRelayOutputsResponse (relay output list)
Receive and validate relay output list
Invoke SetRelay Output Settings to change the settings
GetRelayOutputOptionsRequest
Modify relay output settings
GetRelayOutputOptionsResponse
Change settings and validate response
SetRelayOutputSettingsRequest (RelayOutputToken, Mode= “Bistable”, DelayTime=time1, IdleState= “open”)
SetRelayOutputSettingsResponse (empty message)
Send relay outputs with settings
Invoke Get Relay Output to verify the change
GetRelayOutputsRequest (empty)
Retrieve changes of relay output settings
GetRelayOutputsResponse (relay output list)
SetRelayOutputSettingsRequest (RelayOutputToken, Mode= “Monostable”,IdleState= “closed”)
Receive and validate response
SetRelayOutputSettingsResponse
Retrieve relay output settings
Modify relay output settings
GetRelayOutputSettingsRequest
Receive and validate relay output settings
GetRelayOutputSettingsResponse (Mode= “Monostable”, DelayTime=time2, IdleState= “closed”)
Restore settings
Send relay output settings
SetRelayOutputSettingsRequest

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetRelayOutputsRequest message to retrieve a list of all available relay outputs and their settings
4. The DUT sends the GetRelayOutputsResponse message with list of all available relay outputs and their settings.
5. ONVIF Client verifies the GetRelayOutputsResponse message from the DUT.
6. ONVIF Client selects first relay output from GetRelayOutputsResponse, saves this relay output in **RelayOutput1** variable. Then it runs the following steps:

    6.1. ONVIF Client saves backup copy of **RelayOutput1** variable in **BackupRelayOutput1** variable.

    6.2. ONVIF Client invokes GetRelayOutputOptionsRequest message RelayOutputToken=**RelayOutput1** token as input parameter.

    6.3. The DUT sends GetRelayOutputOptionsResponse. ONVIF client verifies the GetRelayOutputOptionsResponse message.

    6.4. ONVIF Client finds option with Mode= “Bistable” in GetRelayOutputOptionsResponse and populates **time1** variable. If Discrete= “True”, ONVIF Client populates **time1** with the value from DelayTimes closest to 5 seconds. If Discrete= “False” then **time1** should be populated with“5 seconds”.

    6.5. If there is no option with Mode= “Bistable” then skip the steps 6.6 – 6.11

    6.6. ONVIF Client changes the values of the following properties: Mode= “Bistable”, DelayTime=**time1**, IdleState= “open” of **RelayOutput1** variable**.**

    6.7. ONVIF Client invokes SetRelayOutputSettingsRequest message RelayOutput=**RelayOutput1** as input parameter.

    6.8. The DUT sends SetRelayOutputSettingsResponse. ONVIF client verifies the SetRelayOutputSettingsResponse message.

    6.9. ONVIF Client invokes GetRelayOutputsRequest message to retrieve a list of all available relay outputs and their settings.

    6.10. The DUT sends GetRelayOutputsResponse with list of all relay outputs from device with their settings.

    6.11. ONVIF Client verifies that GetRelayOutputsResponse contains relay output with token = **RelayOutput1** token. For the relay output from GetRelayOutputsResponse, ONVIF client checks the values of Mode and IdleState properties, this values should be equal to the values set in step 6.7

    6.12. ONVIF Client finds option with Mode= “Monostable” in GetRelayOutputOptionsResponse and populates **time1** variable. If Discrete= “True”, ONVIF Client populates **time1** with the value from DelayTimes closest to 5 seconds. If Discrete= “False”then **time1** should be populated with“5 seconds”.

    6.13. If there is no option with Mode= “Monostable” then skip the steps 6.12 – 6.17

6.14. ONVIF Client changes the values of the following properties:Mode= “Monostable”, DelayTime=**time1**, IdleState= “closed” of **RelayOutput1** variable**.**

6.15. The DUT sends SetRelayOutputSettingsResponse. ONVIF client verifies the SetRelayOutputSettingsResponse message.

6.16. ONVIF Client invokes GetRelayOutputsRequest message to retrieve a list of all available relay outputs and their settings

6.17. The DUT sends GetRelayOutputsResponse with list of all relay outputs from device with their settings.

6.18. ONVIF Client verifies that GetRelayOutputsResponse contains relay output with token = **RelayOutput1** token. For the relay output from GetRelayOutputsResponse, ONVIF client checks the values of Mode and IdleState properties, this values should be equal to the values set in step 6.14

6.19. ONVIF Client invokes SetRelayOutputSettingsRequest message RelayOutput=**BackupRelayOutput1**as input parameter to restore the configuration changes.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetRelayOutputsResponse message.

The DUT did not send valid GetRelayOutputsResponse message.

The DUT sent empty list of RelayOutputs in GetRelayOutputsResponse.

The DUT did not send SetRelayOutputSettingsResponse message.

The DUT did not send valid SetRelayOutputSettingsResponse message.

The DUT did not send correct changed settings in GetRelayOutputsResponse message.

The DUT did not send GetRelayOutputOptionsResponse message.

The DUT did not send valid GetRelayOutputOptionsResponse message.

4.1.5 IO SETRELAYOUTPUTSETTINGS – INVALID TOKEN

Test Label: Device IO SetRelayOutputSettings Test – Invalid Token

Test Case ID: DEVICEIO-1-1-5

ONVIF Core Specification Coverage: None

Command Under Test: SetRelayOutputSettings

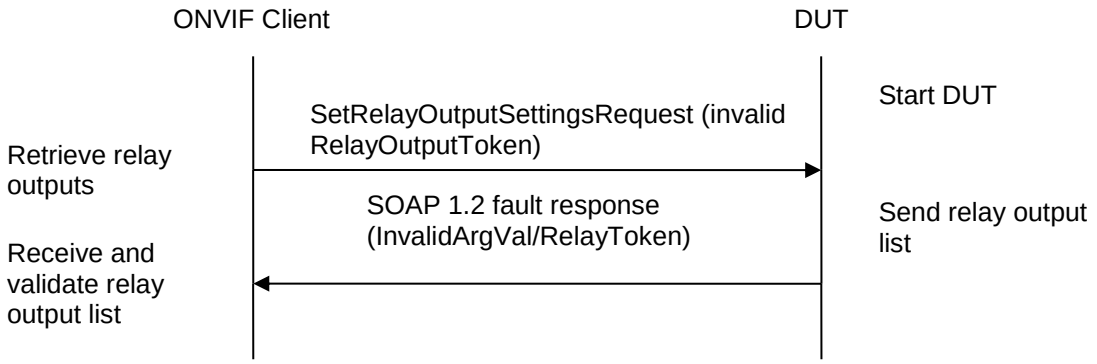
WSDL Reference: deviceio.wsdl

Test Purpose: To verify the behavior ofSetRelayOutputSettings command in case of invalid token.

Pre-Requisite: Device IO service is supported by DUT. Relay Outputs supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes SetRelayOutputSettingsRequest message with RelayOutput token = “OnvifTest123”.
4. The DUT will generate SOAP 1.2 fault message (InvalidArgVal/RelayToken)
5. ONVIF Client verifies fault message.

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.).

4.2 Relay Output State

4.2.1 IO SETRELAYOUTPUTSTATE – BISTABLE MODE (OPENED IDLE STATE)

Test Label: Device IO Set Relay Output State – Bistable Mode (Opened Idle State) Test

Test Case ID: DEVICEIO-1-2-1

ONVIF Core Specification Coverage: None

Command Under Test: GetRelayOutputs, SetRelayOutputSettings, SetRelayOutputState

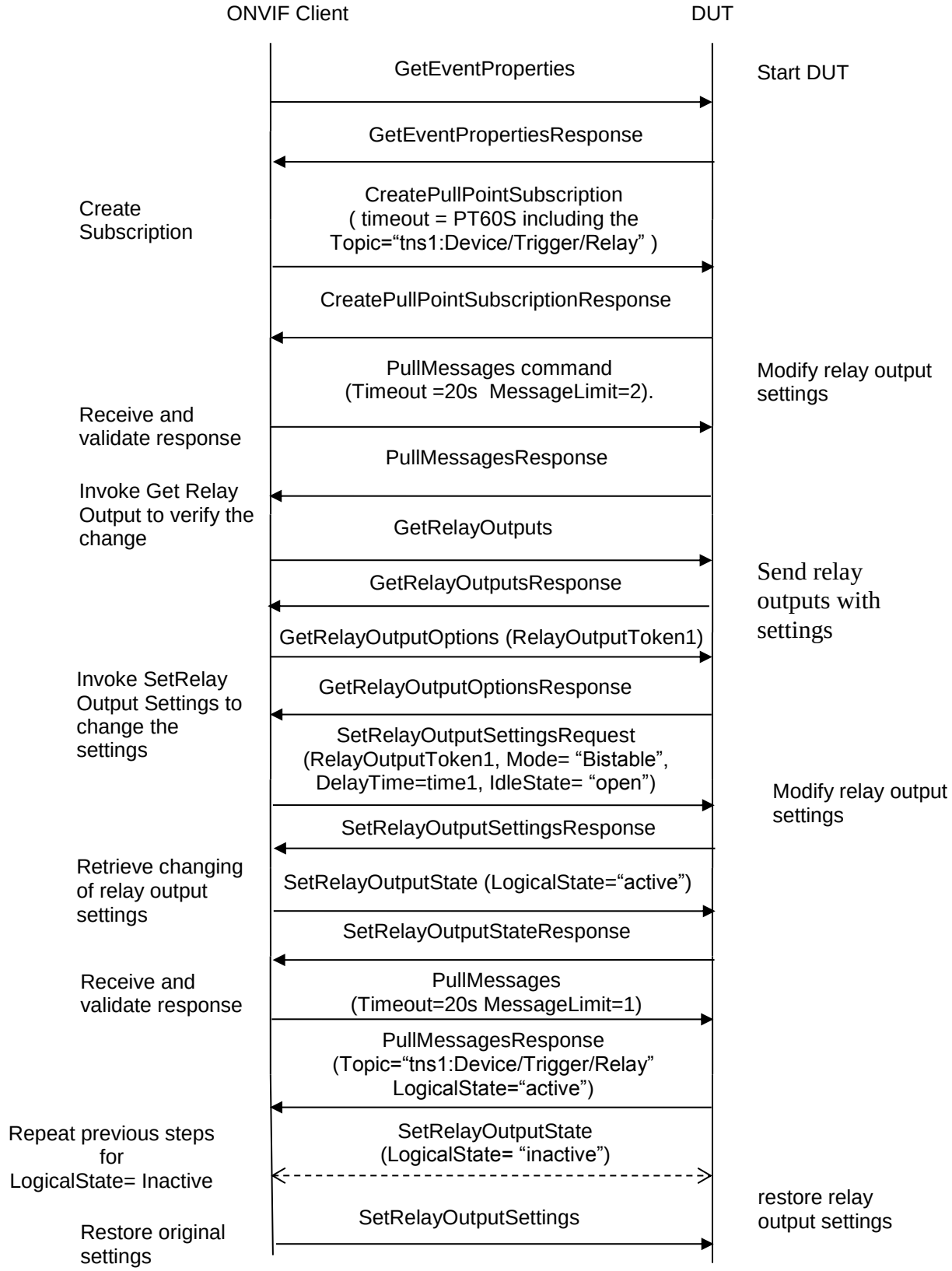
WSDL Reference: deviceio.wsdl, event.wsdl

Test Purpose: To verify the behavior of SetRelayOutputState command in the case of bistable mode and opened idle state as well as appropriate event messaging.

Pre-Requisite: Device IO service is supported by DUT. Relay Outputs supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
GetEventProperties
Start DUT
GetEventPropertiesResponse
Create Subscription
CreatePullPointSubscription (timeout = PT60S including the Topic="tns1:Device/Trigger/Relay")
CreatePullPointSubscriptionResponse
PullMessages command (Timeout =20s MessageLimit=2).
Modify relay output settings
Receive and validate response
PullMessagesResponse
Invoke Get Relay Output to verify the change
GetRelayOutputs
GetRelayOutputsResponse
Send relay outputs with settings
GetRelayOutputOptions (RelayOutputToken1)
Invoke SetRelay Output Settings to change the settings
GetRelayOutputOptionsResponse
SetRelayOutputSettingsRequest (RelayOutputToken1, Mode= "Bistable", DelayTime=time1, IdleState= "open")
Modify relay output settings
SetRelayOutputSettingsResponse
Retrieve changing of relay output settings
SetRelayOutputState (LogicalState="active")
SetRelayOutputStateResponse
Receive and validate response
PullMessages (Timeout=20s MessageLimit=1)
PullMessagesResponse (Topic="tns1:Device/Trigger/Relay" LogicalState="active")
Repeat previous steps for LogicalState= Inactive
SetRelayOutputState (LogicalState= "inactive")
restore relay output settings
Restore original settings
SetRelayOutputSettings

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetEventProperties command.
4. Verify that the DUT sends a GetEventPropertiesResponse, and response contains Topic="tns1:Device/Trigger/Relay" and this topic contains MessageDescription item. This MessageDescription is defined in "Relay Output Trigger" section of ONVIF-DeviceIo-Service- Spec document.
5. ONVIF Client will invoke CreatePullPointSubscription message with a suggested timeout of PT60S and a Filter including the Topic="tns1:Device/Trigger/Relay"
6. Verify that the DUT sends a CreatePullPointSubscriptionResponse.
7. ONVIF Client will invoke PullMessages command with a PullMessagesTimeout of 20s and a MessageLimit of 2.
8. Verify that the DUT sends a PullMessagesResponse that contains at least one NotificationMessage with Topic="tns1:Device/Trigger/Relay" and this message contains a property event.
9. ONVIF Client invokes GetRelayOutputsRequest message to retrieve a list of all available relay outputs and their settings
10. The DUT sends the GetRelayOutputsResponse message with list of all available relay outputs and their settings.
11. ONVIF Client selects first relay output from GetRelayOutputsResponse, saves this relay output in **RelayOutput1** variable. Then it runs the following steps:

    11.1. ONVIF Client saves backup copy of **RelayOutput1** variable in **BackupRelayOutput1** variable.

    11.2. ONVIF Client invokes GetRelayOutputOptionsRequest message RelayOutputToken=**RelayOutput1** token as input parameter.

    11.3. The DUT sends GetRelayOutputOptionsResponse. ONVIF client verifies the GetRelayOutputOptionsResponse message.

    11.4. If there is no option with Mode= "Bistable" in GetRelayOutputOptionsResponse then ONVIF Client skips other steps and pass the test.

    11.5. ONVIF Client changes the values of the following properties: Mode= "Bistable",IdleState= "open" of **RelayOutput1**variable**.**

    11.6. ONVIF Client invokes SetRelayOutputSettingsRequest message RelayOutput=**RelayOutput1**as input parameter.

    11.7. The DUT sends SetRelayOutputSettingsResponse. ONVIF client verifies the SetRelayOutputSettingsResponse message.

    11.8. ONVIF Clientinvokes SetRelayOutputStateRequest message with RelayOutputToken = **RelayOutput1** token and LogicalState= "active" as input parameters.

    11.9. The DUT sends SetRelayOutputStateResponse message. ONVIF Client verifies the response.

11.10. ONVIF Client will invoke PullMessages command with PullMessagesTimeout=20s and MessageLimit=1 as input argument.

11.11. Verify that the DUT sends a PullMessagesResponse that contains one NotificationMessage with Topic="tns1:Device/Trigger/Relay" and this message contains property event and LogicalState = "active" in this property event.

11.12. ONVIF Client invokes SetRelayOutputStateRequest message with RelayOutputToken = **RelayOutput1** token and LogicalState= "inactive" as input parameters.

11.13. The DUT sends SetRelayOutputStateResponse message. ONVIF Client verifies the response.

11.14. ONVIF Client will invoke PullMessages command withPullMessagesTimeout=20s and MessageLimit=1 as input argument.

11.15. Verify that the DUT sends a PullMessagesResponse that contains one NotificationMessage with Topic="tns1:Device/Trigger/Relay" and this message contains property event and LogicalState = "inactive" in this property event.

11.16. ONVIF Client invokes SetRelayOutputSettingsRequest message RelayOutput=**BackupRelayOutput1**as input parameter to restore the configuration changes.

Test Result:

PASS –

DUT passes all assertions.

The GetRelayOutputOptionsResponse did not contain option with Mode= "Bistable"

FAIL –

The DUT did not send GetEventPropertiesResponse message

GetEventPropertiesResponse did not contain Topic="tns1:Device/Trigger/Relay"

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not send PullMessagesResponse message.

PullMessagesResponse did not contain NotificationMessage with contain Topic="tns1:Device/Trigger/Relay"

The DUT sent PullMessagesResponse with incorrect value in LogicalState in NotificationMessage.

The DUT did not send GetRelayOutputsResponse message.

The DUT sent empty list of RelayOutputs in GetRelayOutputsResponse.

The DUT did not send GetRelayOutputOptionsResponse message.

The DUT did not send SetRelayOutputSettingsResponse message.

The DUT did not send SetRelayOutputStateResponse message.

The DUT did not send valid SetRelayOutputStateResponse message.

The DUT did not change relay state as expected.

4.2.2 IO SETRELAYOUTPUTSTATE – BISTABLE MODE (CLOSED IDLE STATE)

Test Label: Device IO Set Relay Output State – Bistable Mode (Closed Idle State) Test.

Test Case ID: DEVICEIO-1-2-2

ONVIF Core Specification Coverage: None

Command Under Test: GetRelayOutputs, SetRelayOutputSettings, SetRelayOutputState

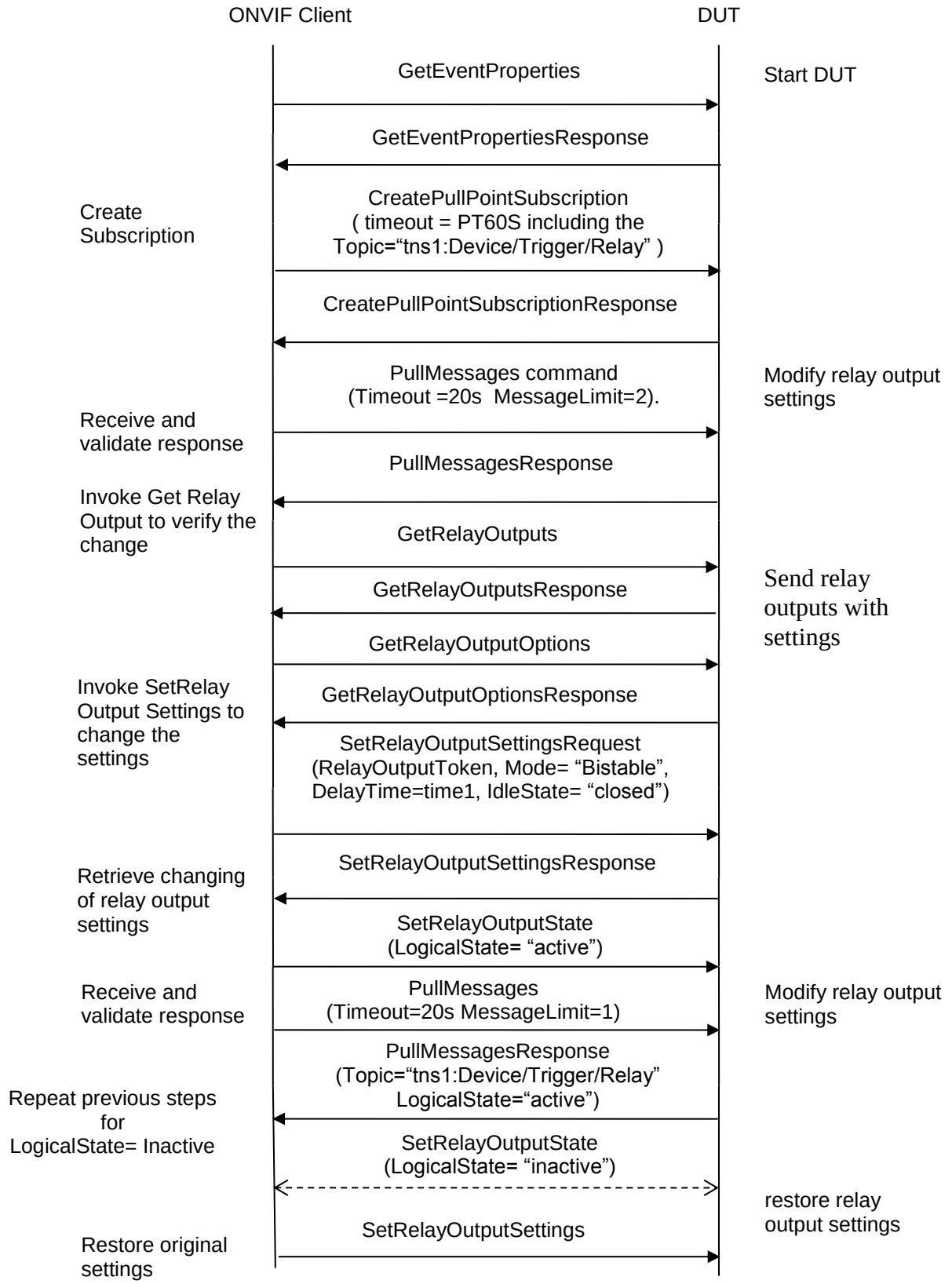
WSDL Reference: deviceio.wsdl

Test Purpose: To verify the behavior of SetRelayOutputState command in the case of bistable mode and closed idle state.

Pre-Requisite: Device IO service is supported by DUT. Relay Outputs supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
GetEventProperties
Start DUT
GetEventPropertiesResponse
Create Subscription
CreatePullPointSubscription (timeout = PT60S including the Topic="tns1:Device/Trigger/Relay")
CreatePullPointSubscriptionResponse
PullMessages command (Timeout =20s MessageLimit=2).
Modify relay output settings
Receive and validate response
PullMessagesResponse
Invoke Get Relay Output to verify the change
GetRelayOutputs
GetRelayOutputsResponse
Send relay outputs with settings
GetRelayOutputOptions
GetRelayOutputOptionsResponse
Invoke SetRelay Output Settings to change the settings
SetRelayOutputSettingsRequest (RelayOutputToken, Mode= "Bistable", DelayTime=time1, IdleState= "closed")
SetRelayOutputSettingsResponse
Retrieve changing of relay output settings
SetRelayOutputState (LogicalState= "active")
Receive and validate response
PullMessages (Timeout=20s MessageLimit=1)
Modify relay output settings
PullMessagesResponse (Topic="tns1:Device/Trigger/Relay" LogicalState="active")
Repeat previous steps for LogicalState= Inactive
SetRelayOutputState (LogicalState= "inactive")
restore relay output settings
SetRelayOutputSettings
Restore original settings

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetEventProperties command.
4. Verify that the DUT sends a GetEventPropertiesResponse, and response contains Topic="tns1:Device/Trigger/Relay" and this topic contains MessageDescription item. This MessageDescription is defined in "Relay Output Trigger" section of ONVIF-DeviceIo-Service- Spec document.
5. ONVIF Client will invoke CreatePullPointSubscription message with a suggested timeout of PT60S and a Filter including the Topic="tns1:Device/Trigger/Relay"
6. Verify that the DUT sends a CreatePullPointSubscriptionResponse.
7. ONVIF Client will invoke PullMessages command with a PullMessagesTimeout of 20s and a MessageLimit of 2.
8. Verify that the DUT sends a PullMessagesResponse that contains at least one NotificationMessage with Topic="tns1:Device/Trigger/Relay" and this message contains a property event.
9. ONVIF Client invokes GetRelayOutputsRequest message to retrieve a list of all available relay outputs and their settings
10. The DUT sends the GetRelayOutputsResponse message with list of all available relay outputs and their settings.
11. ONVIF Client selects first relay output from GetRelayOutputsResponse, saves this relay output in **RelayOutput1** variable. Then it runs the following steps:
    - 11.1. ONVIF Client saves backup copy of **RelayOutput1** variable in **BackupRelayOutput1** variable.
    - 11.2. ONVIF Client invokes GetRelayOutputOptionsRequest message RelayOutputToken=**RelayOutput1** token as input parameter.
    - 11.3. The DUT sends GetRelayOutputOptionsResponse. ONVIF client verifies the GetRelayOutputOptionsResponse message.
    - 11.4. If there is no option with Mode= "Bistable" in GetRelayOutputOptionsResponse then ONVIF Client skips other steps and pass the test.
    - 11.5. ONVIF Client changes the values of the following properties: Mode= "Bistable",IdleState= "closed" of **RelayOutput1**variable**.**
    - 11.6. ONVIF Client invokes SetRelayOutputSettingsRequest message RelayOutput=**RelayOutput1**as input parameter.
    - 11.7. The DUT sends SetRelayOutputSettingsResponse. ONVIF client verifies the SetRelayOutputSettingsResponse message.

11.8. ONVIF Client invokes SetRelayOutputStateRequest message with RelayOutputToken = **RelayOutput1** token and LogicalState= “active” as input parameters.

11.9. The DUT sends SetRelayOutputStateResponse message. ONVIF Client verifies the response.

11.10. ONVIF Client will invoke PullMessages command with PullMessagesTimeout=20s and MessageLimit=1 as input argument.

11.11. Verify that the DUT sends a PullMessagesResponse that contains one NotificationMessage with Topic=“tns1:Device/Trigger/Relay” and this message contains property event and LogicalState = “active” in this property event.

11.12. ONVIF Client invokes SetRelayOutputStateRequest message with RelayOutputToken = **RelayOutput1** token and LogicalState= “inactive” as input parameters.

11.13. The DUT sends SetRelayOutputStateResponse message. ONVIF Client verifies the response.

11.14. ONVIF Client will invoke PullMessages command with PullMessagesTimeout=20s and MessageLimit=1 as input argument.

11.15. Verify that the DUT sends a PullMessagesResponse that contains one NotificationMessage with Topic=“tns1:Device/Trigger/Relay” and this message contains property event and LogicalState = “inactive” in this property event.

11.16. ONVIF Client invokes SetRelayOutputSettingsRequest message RelayOutput=**BackupRelayOutput1**as input parameter to restore the configuration changes.

Test Result:

PASS –

DUT passes all assertions.

The GetRelayOutputOptionsResponse did not contain option with Mode= “Bistable”

FAIL –

The DUT did not send GetEventPropertiesResponse message

GetEventPropertiesResponse did not contain Topic=”tns1:Device/Trigger/Relay”

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not send PullMessagesResponse message.

PullMessagesResponse did not contain NotificationMessage with contain Topic=”tns1:Device/Trigger/Relay”

The DUT sent PullMessagesResponse with incorrect value in LogicalState in NotificationMessage.

The DUT did not send GetRelayOutputsResponse message.

The DUT sent empty list of RelayOutputs in GetRelayOutputsResponse.

The DUT did not send SetRelayOutputSettingsResponse message.

The DUT did not send SetRelayOutputStateResponse message.

The DUT did not send valid SetRelayOutputStateResponse message.

The DUT did not change relay state as expected.

4.2.3 IO SETRELAYOUTPUTSTATE – MONOSTABLE MODE (OPENED IDLE STATE)

Test Label: Device IO Set Relay Output State – Monostable Mode (Opened Idle State)Test.

Test Case ID: DEVICEIO-1-2-3

ONVIF Core Specification Coverage: None

Command Under Test: GetRelayOutputs, SetRelayOutputSettings, SetRelayOutputState

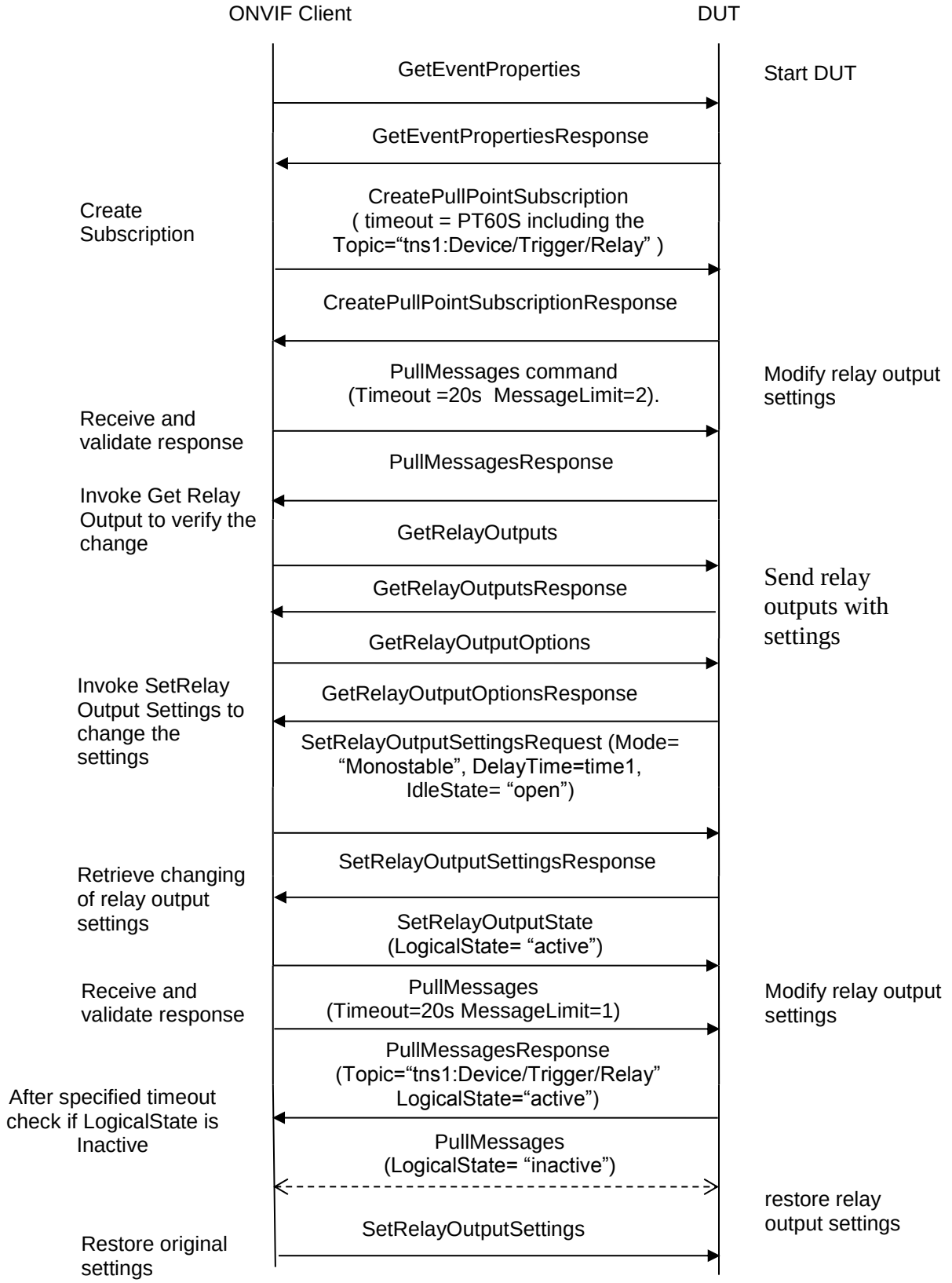
WSDL Reference:deviceio.wsdl

Test Purpose: To verify the behavior of SetRelayOutputState command in the case of bistable mode and opened idle state as well as appropriate event messaging.

Pre-Requisite: Device IO service is supported by DUT. Relay Outputs supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetEventProperties command.
4. Verify that the DUT sends a GetEventPropertiesResponse, and response contains Topic="tns1:Device/Trigger/Relay" and this topic contains MessageDescription item. This MessageDescription is defined in "Relay Output Trigger" section of ONVIF-DeviceIo-Service- Spec document.
5. ONVIF Client will invoke CreatePullPointSubscription message with a suggested timeout of PT60S and a Filter including the Topic="tns1:Device/Trigger/Relay"
6. Verify that the DUT sends a CreatePullPointSubscriptionResponse.
7. ONVIF Clientwill invoke PullMessages command with a PullMessagesTimeout of 20s and a MessageLimit of 2.
8. Verify that the DUT sends a PullMessagesResponse that contains at least one NotificationMessage with Topic="tns1:Device/Trigger/Relay" and this message contains a property event.
9. ONVIF Clientinvokes GetRelayOutputsRequest message to retrieve a list of all available relay outputs and their settings
10. The DUT sends the GetRelayOutputsResponse message with list of all available relay outputs and their settings.
11. ONVIF Client selects first relay output from GetRelayOutputsResponse, saves this relay output in **RelayOutput1** variable. Then it runs the following steps:

    11.1. ONVIF Client saves backup copy of **RelayOutput1** variable in **BackupRelayOutput1** variable.

    11.2. ONVIF Client invokes GetRelayOutputOptionsRequest message RelayOutputToken=**RelayOutput1** token as input parameter.

    11.3. The DUT sends GetRelayOutputOptionsResponse. ONVIF client verifies the GetRelayOutputOptionsResponse message.

    11.4. ONVIF Client finds option with Mode= "Monostable" in GetRelayOutputOptionsResponse and populates **time1** variable. If Discrete= "True", ONVIF Client populates **time1** with the value from DelayTimes closest to 5 seconds. If Discrete= "False" then **time1** should be populated with "5 seconds".

    11.5. If there is no option with Mode= "Monostable" in GetRelayOutputOptionsResponse then ONVIF Client skips other steps and pass the test.

    11.6. ONVIF Client changes the values of the following properties: Mode= "Monostable", DelayTime=**time1**, IdleState= "open" of **RelayOutput1**variable**.**

    11.7. ONVIF Client invokes SetRelayOutputSettingsRequest message RelayOutput=**RelayOutput1**as input parameter.

    11.8. The DUT sends SetRelayOutputSettingsResponse. ONVIF client verifies the SetRelayOutputSettingsResponse message.

11.9. ONVIF Client invokes SetRelayOutputStateRequest message with RelayOutputToken = **RelayOutput1** token and LogicalState= “active” as input parameters.

11.10. The DUT sends SetRelayOutputStateResponse message. ONVIF Client verifies the response.

11.11. ONVIF Client will invoke PullMessages command with PullMessagesTimeout=20s and MessageLimit=1 as input argument.

11.12. Verify that the DUT sends a PullMessagesResponse that contains one NotificationMessage with Topic=“tns1:Device/Trigger/Relay” and this message contains property event and LogicalState = “active” in this property event.

11.13. ONVIF Client waits until timeout (**time1)** is expired.

11.14. ONVIF Client will invoke PullMessages command with PullMessagesTimeout=20s and MessageLimit=1 as input argument.

11.15. Verify that the DUT sends a PullMessagesResponse that contains one NotificationMessage with Topic=“tns1:Device/Trigger/Relay” and this message contains property event and LogicalState = “inactive” in this property event.

11.16. ONVIF Client invokes SetRelayOutputSettingsRequest message RelayOutput=**BackupRelayOutput1**as input parameter to restore the configuration changes.

Test Result:

PASS –

DUT passes all assertions.

The GetRelayOutputOptionsResponse did not contain option with Mode= “Monostable”

FAIL –

The DUT did not send GetEventPropertiesResponse message

GetEventPropertiesResponse did not contain Topic=”tns1:Device/Trigger/Relay”

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not send PullMessagesResponse message.

PullMessagesResponse did not contain NotificationMessage with contain Topic=”tns1:Device/Trigger/Relay”

The DUT sent PullMessagesResponse with incorrect value in LogicalState in NotificationMessage.

The DUT did not send GetRelayOutputsResponse message.

The DUT sent empty list of RelayOutputs in GetRelayOutputsResponse.

The DUT did not send SetRelayOutputSettingsResponse message.

The DUT did not send SetRelayOutputStateResponse message.

The DUT did not send valid SetRelayOutputStateResponse message.

The DUT did not change relay state as expected.

4.2.4 IO SETRELAYOUTPUTSTATE – MONOSTABLE MODE (CLOSED IDLE STATE)

Test Label: Device IO Set Relay Output State – Monostable Mode (Opened Idle State) Test.

Test Case ID: DEVICEIO-1-2-4

ONVIF Core Specification Coverage: None

Command Under Test: GetRelayOutputs, SetRelayOutputSettings, SetRelayOutputState

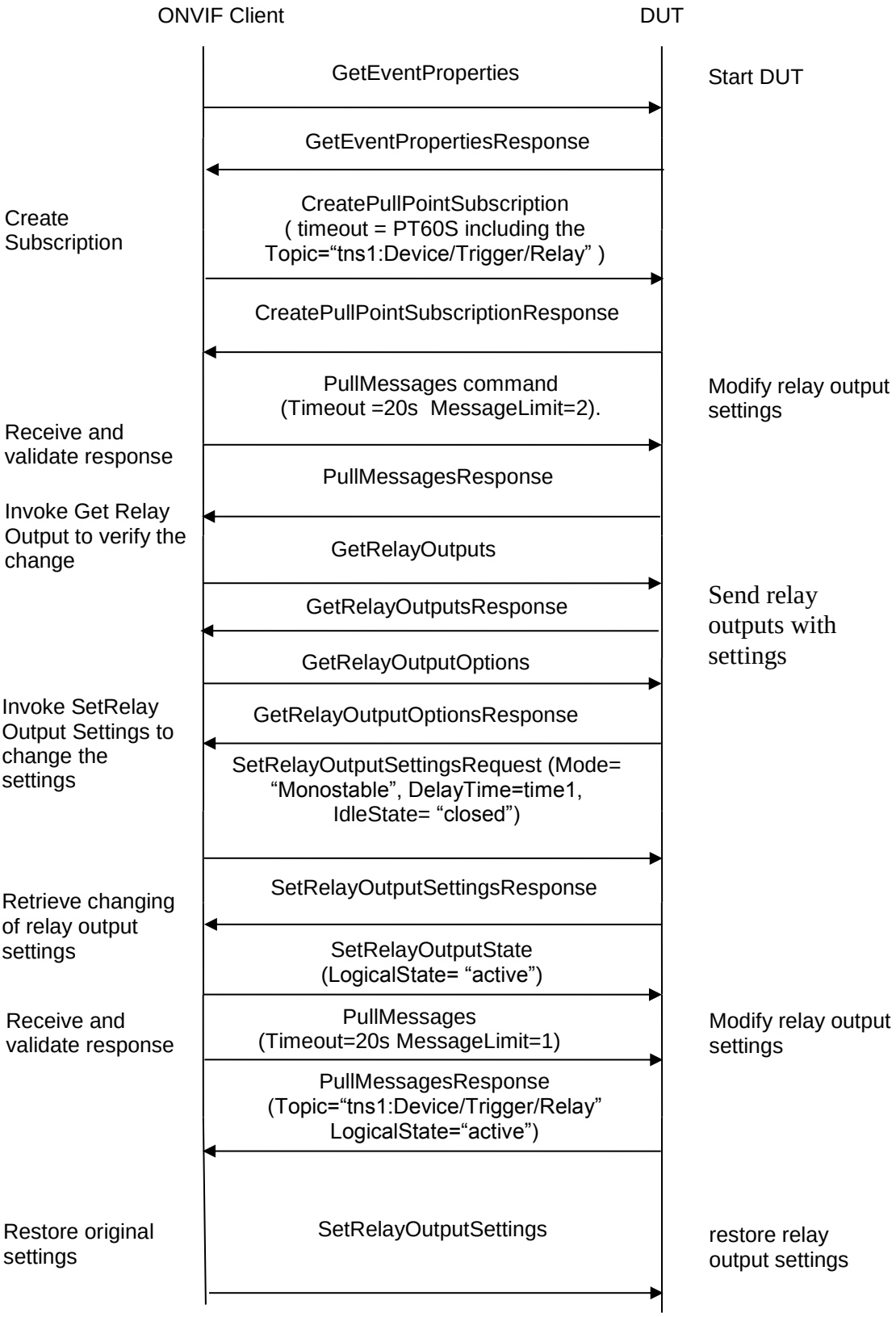
WSDL Reference: deviceio.wsdl

Test Purpose: To verify the behavior of SetRelayOutputState command in the case of monostable mode and closed idle state.

Pre-Requisite: Device IO service is supported by DUT. Relay Outputs supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetRelayOutputsRequest message to retrieve a list of all available relay outputs and their settings
4. The DUT sends the GetRelayOutputsResponse message with list of all available relay outputs and their settings.
5. ONVIF Client selects first relay output from GetRelayOutputsResponse, saves this relay output in **RelayOutput1** variable. Then it runs the following steps:

    5.1. ONVIF Client saves backup copy of **RelayOutput1** variable in **BackupRelayOutput1** variable.

    5.2. ONVIF Client invokes GetRelayOutputOptionsRequest message RelayOutputToken=**RelayOutput1** token as input parameter.

    5.3. The DUT sends GetRelayOutputOptionsResponse. ONVIF client verifies the GetRelayOutputOptionsResponse message.

    5.4. ONVIF Client finds option with Mode= “Monostable” in GetRelayOutputOptionsResponse and populates **time1** variable. If Discrete= “True”, ONVIF Client populates **time1** with the value from DelayTimes closest to 5 seconds. If Discrete= “False” then **time1** should be populated with “5 seconds”.

    5.5. If there is no option with Mode= “Monostable” in GetRelayOutputOptionsResponse then ONVIF Client skips other steps and pass the test.

    5.6. ONVIF Client changes the values of the following properties: Mode= “Monostable”, DelayTime=**time1**, IdleState= “closed” of **RelayOutput1**variable**.**

    5.7. ONVIF Client invokes SetRelayOutputSettingsRequest message RelayOutput=**RelayOutput1**as input parameter.

    5.8. The DUT sends SetRelayOutputSettingsResponse. ONVIF client verifies the SetRelayOutputSettingsResponse message.

    5.9. ONVIF Client invokes SetRelayOutputStateRequest message with RelayOutputToken = **RelayOutput1** token and LogicalState= “active” as input parameters.

    5.10. The DUT sends SetRelayOutputStateResponse message. ONVIF Client verifies the response.

    5.11. ONVIF Client invokes SetRelayOutputSettingsRequest message RelayOutput=**BackupRelayOutput1**as input parameter to restore the configuration changes.

Test Result:

PASS –

DUT passes all assertions.

The GetRelayOutputOptionsResponse did not contain option with Mode= “Monostable”

FAIL –

The DUT did not send GetEventPropertiesResponse message

GetEventPropertiesResponse did not contain Topic="tns1:Device/Trigger/Relay"

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not send PullMessagesResponse message.

PullMessagesResponse did not contain NotificationMessage with contain Topic="tns1:Device/Trigger/Relay"

The DUT sent PullMessagesResponse with incorrect value in LogicalState in NotificationMessage.

The DUT did not send GetRelayOutputsResponse message.

The DUT sent empty list of RelayOutputs in GetRelayOutputsResponse.

The DUT did not send SetRelayOutputSettingsResponse message.

The DUT did not send SetRelayOutputStateResponse message.

The DUT did not send valid SetRelayOutputStateResponse message.

The DUT did not change relay state as expected.

Driving IP-based physical security through global standardization

5 Digital Input Configuration

5.1.1 IO GETDIGITALINPUTS

Test Label: Device IO GetDigitalInputs

Test Case ID: DEVICEIO-3-1-1

ONVIF Core Specification Coverage: None

Command Under Test: GetDigitalInputs

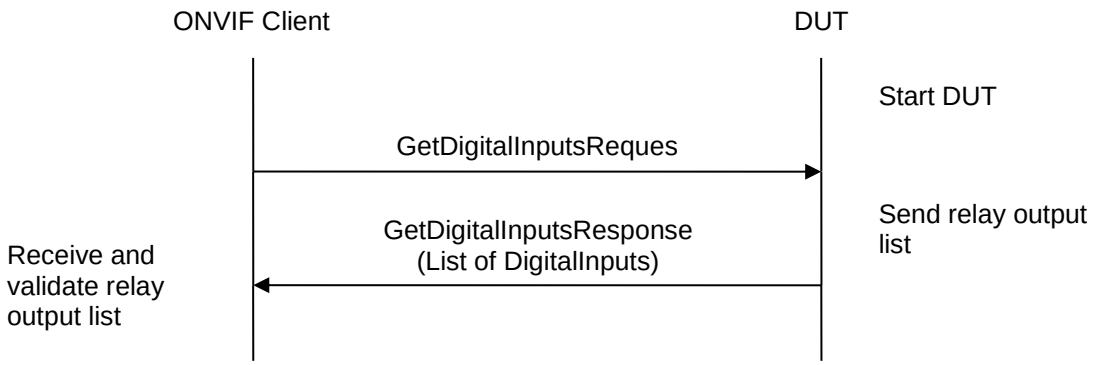
WSDL Reference: deviceio.wsdl

Test Purpose: To verify the DUT returns proper message for GetDigitalInputs request.

Pre-Requisite: Device IO service is supported by DUT. Digital Inputs is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client sends GetDigitalInputsRequest message to DUT to retrieve the list of supported digital input configurations.
4. The DUT sends GetDigitalInputsResponse with the list of supported DigitalInputs.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send GetDigitalInputsResponse message.

The GetDigitalInputsResponse message did not contain Digital Inputs.

5.1.2 IO GETDIGITALINPUTS – VERIFY QUANTITY

Test Label: Device IO GetDigitalInputs– Verify Quantity

Test Case ID: DEVICEIO-3-1-2

ONVIF Core Specification Coverage: None

Command Under Test: GetDigitalInputs, GetServiceCapabilities

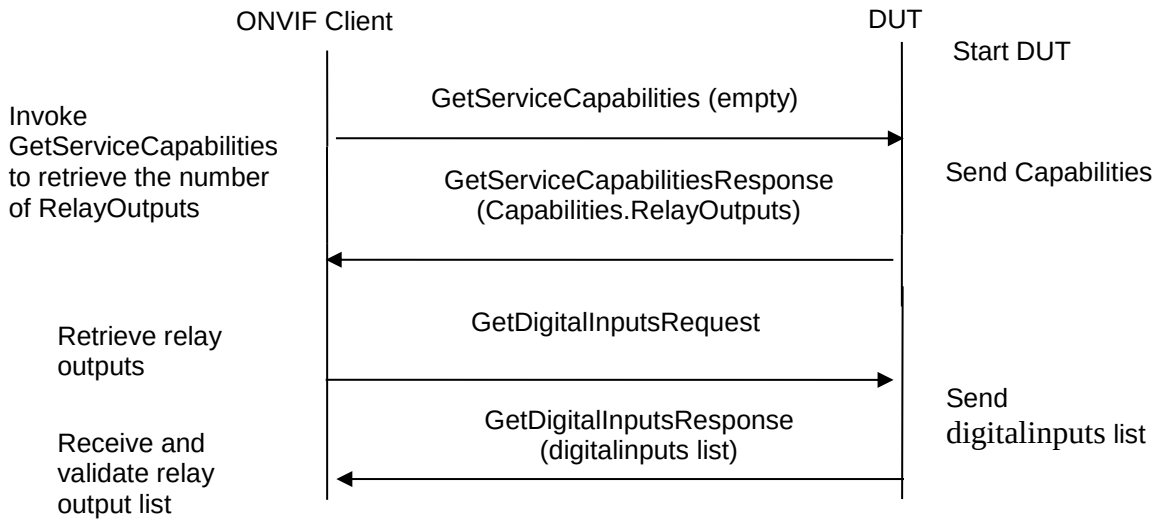
WSDL Reference:deviceio.wsdl

Test Purpose: To verify the DUT returns proper message for GetDigitalInputs request.

Pre-Requisite: Device IO service is supported by DUT. Digital Inputs is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetServiceCapabilitiesRequest (empty message).
4. The DUT sends GetServiceCapabilitiesResponse with the capabilities of the device IO service.
5. ONVIF Client sends GetDigitalInputsRequest message to DUT to retrieve the list of supported digital input configurations.

6. The DUT sends GetDigitalInputsResponse with the list of supported DigitalInputs.
7. ONVIF Client verifies the number of digital inputs in GetDigitalInputsResponse. This number should be equal to the Capabilities.DigitalInputsnumber in GetServiceCapabilitiesResponse.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send GetDigitalInputsResponse message.

The DUT sent incorrect GetDigitalInputsResponse message.

The DUT did not send GetServiceCapabilitiesResponse message.

The DUT sent empty list of DigitalInputsin GetDigitalInputsResponse.

The number of Digital Inputs in GetDigitalInputsResponse message is not equal to Capabilities.DigitalInputs number from GetServiceCapabilitiesResponse message.

5.1.3 IOGET DIGITAL INPUT CONFIGURATION OPTIONS

Test Label: Device IO Get Digital Input Configuration Options

Test Case ID: DEVICEIO-3-1-3

ONVIF Core Specification Coverage: None

Command Under Test: GetDigitalInputs, GetDigitalInputConfigurationOptions

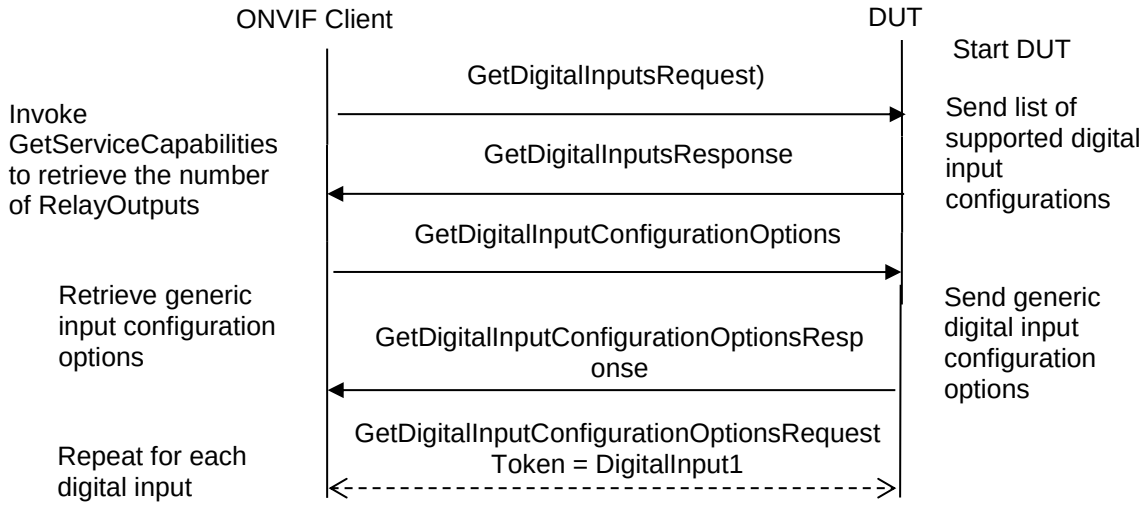
WSDL Reference:deviceio.wsdl

Test Purpose: To verify the behavior of GetDigitalInputConfigurationOptions command.

Pre-Requisite: Device IO service is supported by DUT. DigitalInputs is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetDigitalInputsRequest message to DUT to retrieve the list of supported digital input configurations.
4. The DUT sends GetDigitalInputsResponse with the list of supported digital inputs.
5. ONVIF Client invokes GetDigitalInputConfigurationOptions(Empty) message to DUT to retrieve the generic input configuration options.
6. The DUT sends GetDigitalInputConfigurationOptionsResponse with generic digital input configuration options.
7. For each digital input in GetDigitalInputsResponse, ONVIF Client saves this digital input in **DigitalInput1** variable and runs the following steps:
   7.1. ONVIF Client invokes GetDigitalInputConfigurationOptionsRequest with Token = **DigitalInput1** token as input argument.
   7.2. The DUT sends GetDigitalInputConfigurationOptionsResponse with configuration options for the given token.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send GetDigitalInputsResponse message.

The GetDigitalInputsResponse message did not contain Digital Inputs.

The DUT Did not send GetDigitalInputConfigurationOptionsResponse.

The GetDigitalInputConfigurationOptionsResponse did not contain Digital Input Options.

5.1.4 IO DIGITAL INPUT CONFIGURATION

Test Label: Device IO Digital Input Configuration

Test Case ID: DEVICEIO-3-1-4

ONVIF Core Specification Coverage: None

Command Under Test: GetDigitalInputs, GetDigitalInputConfigurationOptions, SetDigitalInputConfigurations

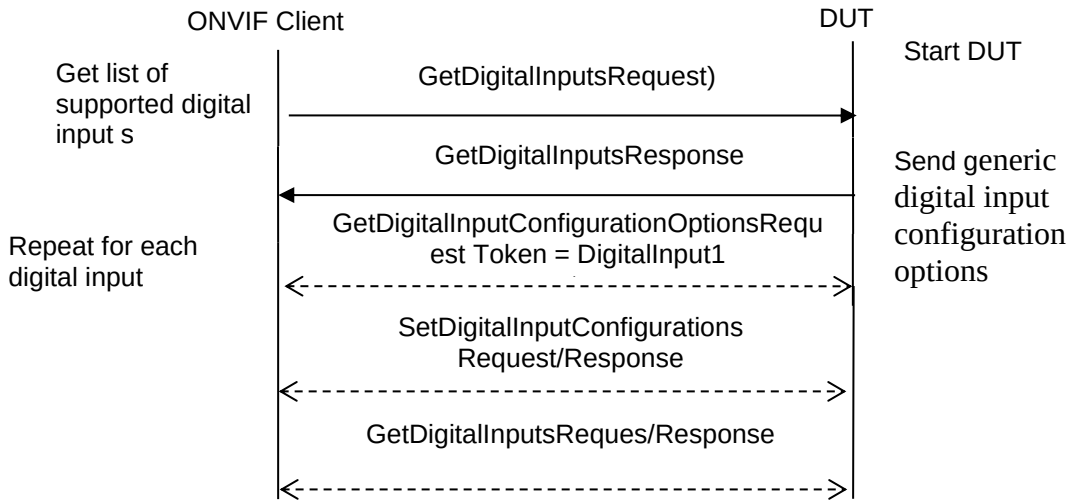
WSDL Reference:deviceio.wsdl

Test Purpose: To verify the behavior o fGetDigitalInputs, GetDigitalInputConfigurationOptions, SetDigitalInputConfigurations commands.

Pre-Requisite: Device IO service is supported by DUT. Digital Inputs is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetDigitalInputsRequest message to DUT to retrieve the list of supported digital input configurations.
4. The DUT sends GetDigitalInputsResponse with the list of supported DigitalInputs.
5. For each digital input in GetDigitalInputsResponse, ONVIF Client saves this digital input in **DigitalInput1** variable and runs the following steps:

5.1. ONVIF Client invokes GetDigitalInputConfigurationOptionsRequest with Token = **DigitalInput1** token as input argument.

5.2. The DUT sends GetDigitalInputConfigurationOptionsResponse with configuration options for the given token.

5.3. If GetDigitalInputConfigurationOptionsResponse contains DigitalInputOptionsIdleState = “closed” then run the following steps:

5.3.1. ONVIF Client changes **DigitalInput1** IdleState property to “closed”.

5.3.2. ONVIF Client invokes SetDigitalInputConfigurationsRequest with **DigitalInput1** as input argument.

5.3.3. The DUT sends SetDigitalInputConfigurationsResponse. ONVIF Client verifies the response.

5.3.4. ONVIF Client invokes GetDigitalInputsRequest.

5.3.5. The DUT sends GetDigitalInputsResponse with the list of Digital Inputs.

5.3.6. ONVIF Client verifies that the GetDigitalInputsResponse contains digital input with token = **DigitalInput1** token, also it verifies that IdleState value equals to the value set up in the step 5.3.1.

5.4. If GetDigitalInputConfigurationOptionsResponse contains DigitalInputOptions.IdleState = “open” then run the following steps:

5.4.1. ONVIF Client changes **DigitalInput 1.**IdleState property to “open”.

5.4.2. ONVIF Client invokes SetDigitalInputConfigurationsRequest with **DigitalInput1** as input argument.

5.4.3. The DUT sends SetDigitalInputConfigurationsResponse. ONVIF Client verifies the response.

5.4.4. ONVIF Client invokes GetDigitalInputsRequest.

5.4.5. The DUT sends GetDigitalInputsResponse with the list of Digital Inputs.

5.4.6. ONVIF Client verifies that the GetDigitalInputsResponse contains digital input with token = **DigitalInput1** token, also it verifies that IdleState value equals to the value set up in the step 5.4.1.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send GetDigitalInputsResponse message.

The GetDigitalInputsResponse message did not contain Digital Inputs.

The DUT Did not send GetDigitalInputConfigurationOptionsResponse.

The GetDigitalInputConfigurationOptionsResponse did not contain Digital Input Options.

The DUT did not send SetDigitalInputConfigurationsResponse.

The DUT did not change IdleState.