Driving IP-based physical security through global standardization

ONVIF™

Base Test Specification

Version 17.01

January 2017

Driving IP-based physical security through global standardization

Driving IP-based physical security through global standardization

Revision History

Ver.	Date	Description
1.02.4	29th Jun, 2011	First issue of Base Test Specification
11.12	22nd Dec, 2011	New version numbering scheme has been applied. Requirement level terms have been removed. Refine / add capabilities related test cases. Add namespace handling test cases in Event and Discovery test Add several Relay Output test cases
12.06	18th Jun, 2012	GetServices and Device Management Service Capabilities test cases have been added. HTTP Digest Security test cases have been added. Deprecated most Event test cases with new ones that include validation of WS-Addressing elements. Event Service Capabilities test cases have been added.
12.12	20th Dec, 2012	BASIC NOTIFICATION INTERFACE – NOTIFY and BASIC NOTIFICATION INTERFACE - NOTIFY FILTER were updated with Test Case ID change.
13.06	Jun, 2013	SET NETWORK DEFAULT GATEWAY CONFIGURATION - IPV4 and SET NETWORK DEFAULT GATEWAY CONFIGURATION – IPV6 were updated with Test Cases ID change. BASIC NOTIFICATION INTERFACE - INVALID MESSAGE CONTENT FILTER, BASIC NOTIFICATION INTERFACE - INVALID MESSAGE CONTENT FILTER, BASIC NOTIFICATION INTERFACE - INVALID TOPIC EXPRESSION, BASIC NOTIFICATION INTERFACE – UNSUBSCRIBE, BASIC NOTIFICATION INTERFACE - RESOURCE UNKNOWN, REALTIME PULLPOINT SUBSCRIPTION - INVALID MESSAGE CONTENT FILTER, REALTIME PULLPOINT SUBSCRIPTION - INVALID TOPIC EXPRESSION, REALTIME PULLPOINT SUBSCRIPTION – UNSUBSCRIBE, REALTIME PULLPOINT SUBSCRIPTION - TIMEOUT were updated with Test Cases ID change. Annex A.17 was updated. Annex A.18 was added. IO COMMAND SETRELAYOUTPUTSETTINGS – INVALID TOKEN and IO COMMAND SETRELAYOUTPUTSTATE – INVALID TOKEN were updated with Test Cases ID change. IPV4 LINK LOCAL ADDRESS was updated.
13.12	Dec, 2013	New Seek test cases were added: SEEK EVENTS SEEK EVENTS – REVERSE SEEK EVENTS – BEGIN OF BUFFER New IP Filtering test cases were added: GET IP ADDRESS FILTER SET IP ADDRESS FILTER – IPv4 ADD IP ADDRESS FILTER – IPv4 REMOVE IP ADDRESS FILTER – IPv4 New capabilities test case was added:

		GET SERVICES – ADVANCED SECURITY SERVICE New event test case was added: REALTIME PULLPOINT SUBSCRIPTION – PULLMESSAGES AS KEEP-ALIVE The following test cases were updated with Id change IPV4 LINK LOCAL ADDRESS GET NETWORK PROTOCOLS CONFIGURATION SET NETWORK PROTOCOLS CONFIGURATION SET NETWORK PROTOCOLS CONFIGURATION - UNSUPPORTED PROTOCOLS SYSTEM COMMAND SETSYSTEMDATEANDTIME SYSTEM COMMAND SETSYSTEMDATEANDTIME USING NTP BASIC NOTIFICATION INTERFACE - INVALID TOPIC EXPRESSION REALTIME PULLPOINT SUBSCRIPTION - INVALID TOPIC EXPRESSION APP_MAX_DELAY was renamed to DISCOVERY_TIMEOUT. Description of DISCOVERY_TIMEOUT was updated. Description in Annex A.7 was updated.
14.06	Jun, 2014	The following test cases were updated with Id change: DEVICE SCOPES CONFIGURATION REALTIME PULLPOINT SUBSCRIPTION – PULLMESSAGES AS KEEP-ALIVE SEEK EVENTS SEEK EVENTS – REVERSE
14.12	Dec, 2014	The following test cases were updated with Id change: SECURITY COMMAND CREATEUSERS REALTIME PULLPOINT SUBSCRIPTION – PULLMESSAGES AS KEEP-ALIVE The following new test cases were added: GET SERVICES – ACCESS RULES SERVICE GET SERVICES – CREDENTIALS SERVICE GET SYSTEM URIS START SYSTEM RESTORE START SYSTEM RESTORE – INVALID BACKUP FILE GET REMOTE USER SET REMOTE USER The following new annexes were added: Annex A.20 TooManyUsers fault check Annex A.21 Get service capabilities
15.06	Jun, 2015	The following test cases were added: GET SERVICES – SCHEDULE SERVICE The following test cases were updated: IPV4 LINK LOCAL ADDRESS The following new annexes were updated: A.17 Action URI's for Event Service Messages
16.01	Dec, 2015	The following test cases were added:

		REALTIME PULLPOINT SUBSCRIPTION - SET SYNCHRONIZATION POINT BASIC NOTIFICATION INTERFACE - SET SYNCHRONIZATION POINT The tests: DEVICE-3-1-14, DEVICE-3-1-8, DEVICE-3-1-15 and IPCONFIG-1-1-5 have been updated.
16.06	March 8, 2016	The tests EVENT-3-1-26, EVENT-3-1-27 have been added
16.06	March 15, 2016	The tests EVENT-3-1-28 - EVENT-3-1-31 have been added
16.06	May 15	Auxiliary operation section and test case added
16.07	July 8, 2016	used GetServiceCapabilities to find out which auxiliary commands are available (not GetCapabilities)
16.12	December 08, 2016	REALTIME PULLPOINT SUBSCRIPTION – DIGITAL INPUT EVENT was moved from ONVIF Base Test specification to ONVIF Device IO Test specification.
17.01	January 09, 2017	6.8 Monitoring Events Item was added (moved from ONVIF Profile Q Test Specification)

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 implementations from different vendors. The set of ONVIF test specifications describes the test cases need to verify the [ONVIF Network Interface Specs] and [ONVIF Conformance] requirements. 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 Base Test Specification acts as a supplementary document to the [ONVIF Network Interface Specs], illustrating test cases that need to be executed and passed. And also this specification also 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. As the test tool performs as a Client during testing, this test tool is referred as ONVIF Client hereafter.

1.1 Scope

This ONVIF Base 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 core services which are defined in [ONVIF Network Interface Specs].

The principal intended purposes are:

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

This specification does not address the following:

- Product use cases and non-functional (performance and regression) testing.
- SOAP Implementation Interoperability test i.e. Web Services Interoperability Basic Profile version 2.0 (WS-I BP2.0).
- Network protocol implementation Conformance test for HTTPS, HTTP, RTP and RTSP protocols.
- 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 subset of it.

This ONVIF Base Test Specification covers core parts of functional blocks in [ONVIF Network Interface Specs]. The following sections describe the brief overview and scope of each functional block.

1.1.1 IP Configuration

IP Configuration covers the test cases needed for the verification of IP configuration features as mentioned in [ONVIF Network Interface Specs]. IP configuration section defines the ONVIF IP configuration compliance requirements and recommendations.

The scope of this specification is to cover following configurations.

- IPv4 configuration
    - Static IP configuration
    - Link-local address configuration

- DHCP configuration

- IPv6 configuration
  - Static IP configuration
  - Stateless IP configuration
  - Stateful IP configuration

1.1.2 Device Discovery

Device discovery and location of the device services in the network are achieved using a multicast discovery protocol defined in WS-Discovery. The communication between client and target service is done using Web Services, notably SOAP/UDP.

Device Discovery testing tests the following:

- Device discovery in the ad-hoc network
- Location of one or more device services
- Enable discovery of service by type and within scope
- SOAP 1.2 envelopes
- SOAP 1.2 fault messages

Refer to Table 1 for Device Discovery tests.

Table 1 Device Discovery

Feature	Messages
Device Discovery	Hello Probe Probe Match Bye

1.1.3 Device Management

Device Management covers the test cases for the verification of the device service as mentioned in [ONVIF Network Interface Specs]. The device service is the entry point to all other services provided by a device.

The scope of this specification is to cover interfaces with regard to following subcategories of device service.

- Capabilities
- Network

- System
- Security
- Input/Output(I/O)
- IP Filtering
- Monitoring Events

In addition, the following behavior of a device is confirmed as the representative of all services that are defined by [ONVIF Network Interface Specs].

- Namespace handling

1.1.4 Event Handling

Event handling covers the test cases for the verification of the Event service as mentioned in [ONVIF Network Interface Specs] and [ONVIF Event WSDL].

The event handling test cases cover the following mandatory interfaces:

- Basic Notification Interface
    - This test specification provides test cases to verify the implementation of the Basic Notification Interface of a device. The mechanisms to subscribe and unsubscribe to an event are covered as well as the mechanism to renew a subscription manager and receive events via Notify messages. The correct use of the message content filter is also tested.
- Real Time Pull Point Notification Interface
    - Test cases to verify the implementation of the Realtime PullPoint Interface are provided by this test specification. The CreatePullPointSubscription command is tested as well as the PullMessages command in combination with message content filtering to retrieve the events.
- Seek
    - Test cases to verify the implementation of the Seek Interface are provided by this test specification. The Seek command is tested as well as the PullMessages command to retrieve the events from persistent notification storage.

1.1.5 Security

Security covers the test cases needed for the verification of required security features as mentioned in [ONVIF Network Interface Specs]. The scope of this specification is limited to Message level security and Username Token Profile.

2 Terms and Definitions

2.1 Definitions

This section describes terms and definitions used in this document.

Address	An address refers to a URI
Capability	The capability commands allow a client to ask for the services provided by an ONVIF device.
Network	A network is an interconnected group of devices communicating using the Internet protocol.
Proxy Server	A server that services the requests of its clients by forwarding requests to other servers. A Proxy provides indirect network connections to its clients.
SOAP	SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols.
Switching Hub	A device for connecting multiple Ethernet devices together, making them act as a single network segment.
Target Service	An endpoint that makes itself available for discovery.
Test Tool	ONVIF Device Test Tool that tests ONVIF device implementation towards the ONVIF Core Specifications.

2.2 Abbreviations

This section describes abbreviations used in this document.

DUT	Device Under Test
DP	Discovery Proxy
DNS	Domain Name System
DHCP	Dynamic Host Configuration Protocol
HTTP	Hyper Text Transport Protocol
HTTPS	Hyper Text Transport Protocol over Secure Socket Layer
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
NTP	Network Time Protocol
POSIX	Portable Operating System Interface
RTCP	RTP Control Protocol
RTSP	Real Time Streaming Protocol

RTP	Real-time Transport Protocol
SDP	Session Description Protocol
TCP	Transport Control Protocol
UTC	Coordinated Universal Time
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
WSDL	Web Services Description Language
WS-I BP 2.0	Web Services Interoperability Basic Profile version 2.0
XML	eXtensible Markup Language

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 device under test

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.

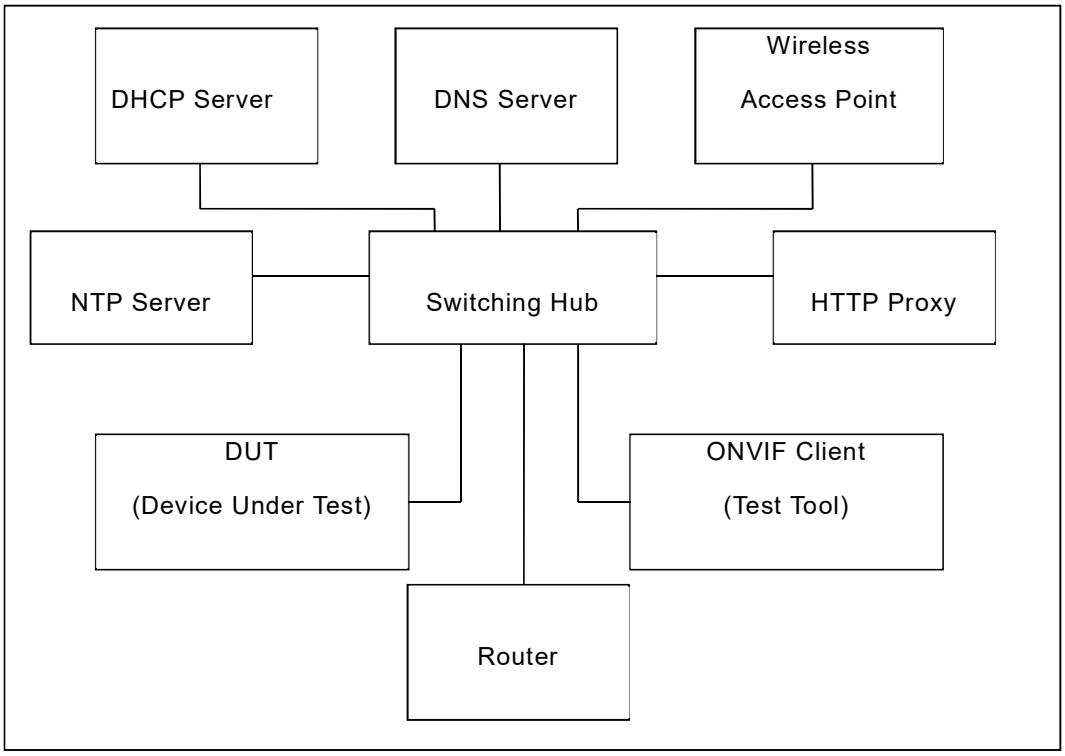


Figure 1: Test Configuration for DUT

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 behaviour of the DUT. It handles both expected and unexpected behaviour.

HTTP Proxy: provides facilitation in case of RTP and RTSP tunnelling 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.

Router: provides router advertisements for IPv6 configuration.

3.2 Prerequisites

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

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

The device under test shall be discovered by the ONVIF Client that exists in the testing environment.

The device under test shall support SetNetworkInterfaces method.

The device under test that supports Link-Local address of IPv4 shall support SetZeroConfiguration method.

The device under test shall be configured with routable IPv4 address.

The following tests are related to IPv4:

- Static IP configuration
- Dynamic IP configuration of Link-Local address
- Dynamic IP configuration (DHCP)

The following tests are related to IPv6:

- Stateless IP configuration which accepts Router Advertisement.
- Stateless IP configuration which uses Neighbor Discovery.
- Stateful IP configuration (DHCPv6)

The device under test shall have at least one network interface that provides IPv4 connectivity. And it

should have at least one network interface that provides IPv6 connectivity.

The device under test that has multiple network interfaces (Wired Ethernet i.e. 802.3af and Wireless Ethernet i.e. 802.11a/b/g/n), initial testing will be performed on the Wired Ethernet network interface. After completion of all testing on the Wired Ethernet network interface, all tests shall be repeated on Wireless Ethernet network interface.

ONVIF Test Specification restricts all testing to Wired Ethernet and/or Wireless Ethernet network interface, other interfaces like USB, Bluetooth etc are outside the scope of the testing.

Refer to Section 4 for IP Configuration Test Cases.

3.3.2 Device Discovery

The device under test shall be discovered by the ONVIF Client that exists in the testing environment.

Failing to discover the device on the network results in failure of the test procedure.

Failing to locate the device services on the network results in failure of the test procedure.

Failing to select the device for interaction results in failure of the test procedure.

Detection of undefined namespace required for Core Specification results in failure of the test procedure.

In certain test cases, the client may check the discovery mode and change it to “discoverable” to perform the test. At the end of the test procedure it resets the discovery mode value.

Refer to Section 5 for Device Discovery Test Cases.

3.3.3 Device Management

The device under test shall demonstrate device, media and event capability. A DUT that does not display mandatory device capability results in failure of test procedure.

Some commands like CreateUsers, SetUser, etc may have restricted access. In this case ONVIF Client should execute the test cases in the administrative mode.

If DUT does not support Media service, then (GET CAPABILITIES for MEDIA) shall be responded with SOAP 1.2 fault message (env:Receiver, ter:ActionNotSupported, ter:NoSuchService).

If DUT does not support PTZ, then (GET CAPABILITIES for PTZ) shall be responded with SOAP 1.2 fault message (env:Receiver, ter:ActionNotSupported, ter:NoSuchService).

Refer to Annex A.2 for valid host name.

The Monitoring Events section covers the test cases needed for check of monitoring property events.

DUT shall give the Event Service entry point by GetServices command. Otherwise these test cases will be skipped.

DUT can support the following property Monitoring Events:

- tns1:Monitoring/ProcessorUsage
- tns1:Monitoring/OperatingTime/LastReset

- tns1:Monitoring/OperatingTime/LastReboot
- tns1:Monitoring/OperatingTime/LastClockSynchronization
- tns1:Monitoring/Backup/Last
- tns1:Device/HardwareFailure/FanFailure
- tns1:Device/HardwareFailure/PowerSupplyFailure
- tns1:Device/HardwareFailure/StorageFailure
- tns1:Device/HardwareFailure/TemperatureCritical

If DUT supports at least one Monitoring event:

- 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.
- The following tests are performed
  - Getting of Processor Usage event and generate Processor Usage property event with initial state
  - Getting of Last Reset event and generate Last Reset property event with initial state
  - Getting of Last Reset event and generate Last Reboot property event with initial state
  - Generate Last Reboot property event after device reboot
  - Getting of Last Clock Synchronization event and generate Last Clock Synchronization property event with initial state
  - Generate Last Clock Synchronization property event after clock synchronization via SetSystemDateAndTime command
  - Generate Last Clock Synchronization property event after clock synchronization with NTP server
  - Getting of Last Backup event and generate Last Backup property event with initial state
  - Getting of Fan Failure event and generate Fan Failed property event with initial state
  - Getting of Power Supply Failure event and generate Power Supply Failed property event with initial state
  - Getting of Storage Failure event and generate Storage Failed property event with initial state

- Getting of Critical Temperature event and generate Critical Temperature property event with initial state

Please refer to Section 6.7 for Monitoring Events Test Cases.

Refer to Section 6 for Device Management Test cases.

3.3.4 Event Handling

Prior to the execution of Event handling test cases, DUT shall be discovered by the ONVIF Client and it shall demonstrate event capabilities to ONVIF Client using the device management service.

If the DUT supports “property events” the ONVIF Client uses the SetSynchronizationPoint method from the event service to trigger events for testing Basic Notification Interface, Realtime PullPoint Interface; the ONVIF Client uses the SetSynchronizationPoint from the Media service to trigger events for testing metadata streaming.

If the DUT does not support “property events”, the event should be triggered manually.

The time of the ONVIF Client and the DUT should be synchronized.

In certain test cases the Test Tool may create a Subscription Manager representing the subscription. In such cases the procedure will make sure that all newly created Subscription Managers are deleted at the end of the test procedure.

In certain test cases the Test Tool may create or change media entities (e.g. add a MetadataConfiguration to a profile). In such cases the procedure will delete those modifications at the end of the test procedure.

Refer to Section 7 for Event handling Test Cases.

3.3.5 Security

The DUT shall support WS-Security User token profile. Consequently, the DUT shall support user profiles that conform to the User token profile and handling of these users via Device Management.

The details of Access rights and Access policies are outside the scope of this document. However, ONVIF Client shall be able to access any given part of any given service supported by the DUT with a user with Administrator rights.

Refer to Section 7.4.1 for Security Test Cases.

3.3.6 Authentication method selection as a testing framework

According to the later version of [ONVIF Network Interface Specs], it requires ONVIF client to support both HTTP digest and WS-UsernameToken functionality as authentication functionality. Therefore, ONVIF Client (ONVIF Device Test Tool in this context) as a testing framework shall properly select authentication method between the two based on the response from DUT toward specific request. The following is the deterministic procedure on which authentication method is to be selected.

Procedure:

1. ONVIF Client invokes a specific command which is under testing without any user credentials (no WS-UsernameToken, no HTTP digest authentication header).

2. If the DUT returns a correct response, then ONVIF Client determines that DUT does not require any user authentication toward the command according to the configured security policy.

3. If the DUT returns HTTP 401 Unauthorized error along with WWW-Authentication: Digest header, then ONVIF Client determines that DUT supports HTTP digest authentication. ONVIF Client shall provide with the proper level of user credential to continue the test procedure.

4. If the DUT returns SOAP fault (Sender/NotAuthorized) message, then ONVIF Client determines that WS-UsernameToken is supported by DUT. ONVIF Client shall provide with the proper level of user credential to continue the test procedure.

4 IP Configuration Test Cases

4.1 IPv4

4.1.1 IPV4 STATIC IP

Test Label: IP Configuration IPv4 Static IP Configuration

Test Case ID: IPCONFIG-1-1-1

ONVIF Core Specification Coverage: IP Configuration

Command Under Test: None

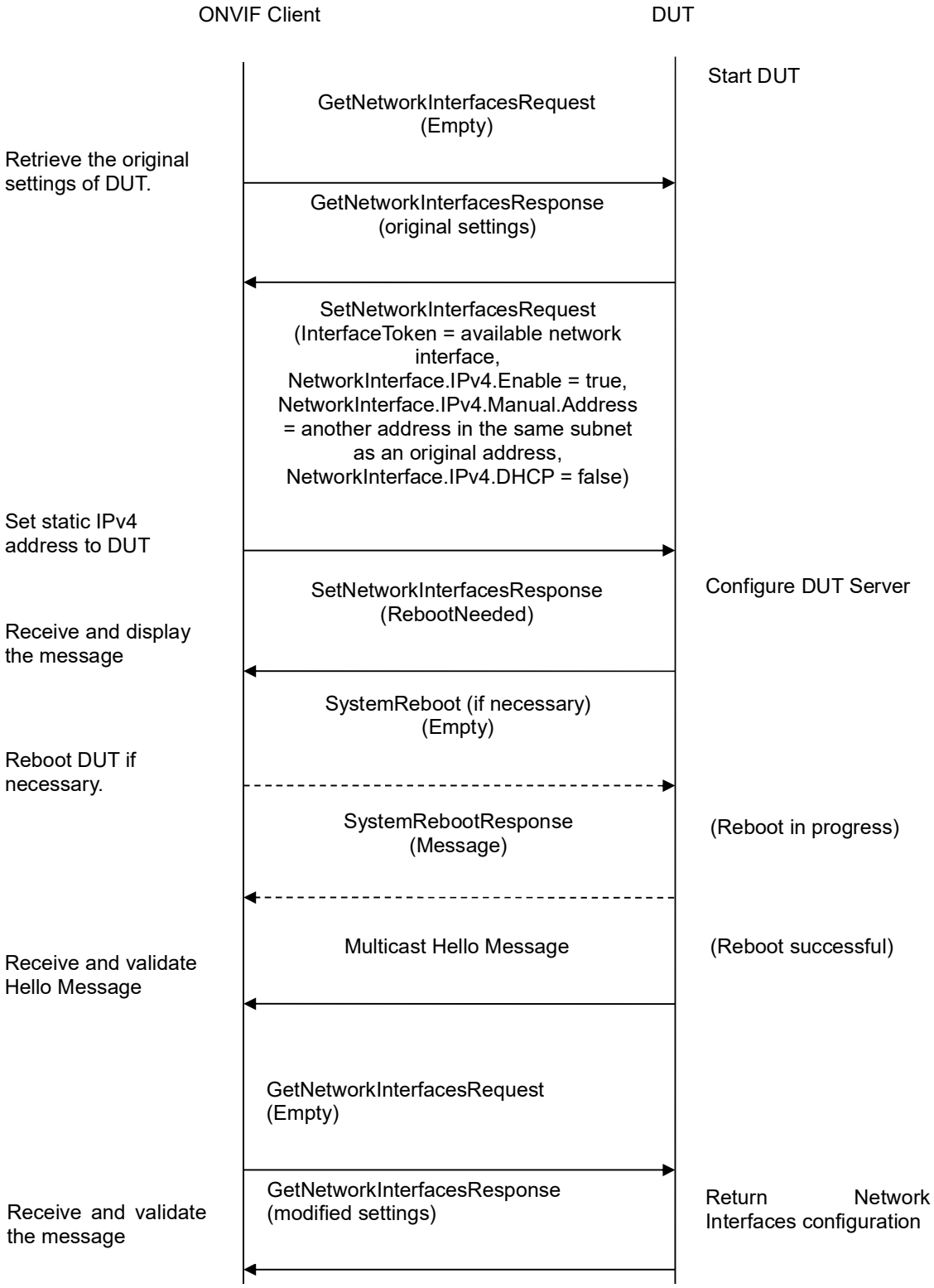
WSDL Reference: devicemgmt.wsdl

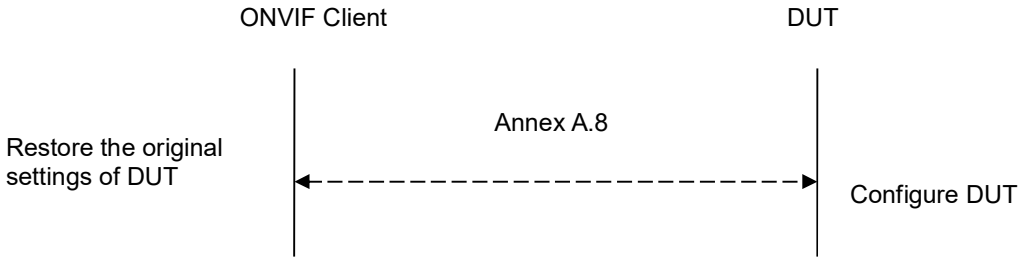
Test Purpose: To test IPv4 Static IP Configuration.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
GetNetworkInterfacesRequest
(Empty)
Retrieve the original settings of DUT.
GetNetworkInterfacesResponse
(original settings)
SetNetworkInterfacesRequest
(InterfaceToken = available network interface,
NetworkInterface.IPv4.Enable = true,
NetworkInterface.IPv4.Manual.Address = another address in the same subnet as an original address,
NetworkInterface.IPv4.DHCP = false)
Set static IPv4 address to DUT
SetNetworkInterfacesResponse
(RebootNeeded)
Configure DUT Server
Receive and display the message
SystemReboot (if necessary)
(Empty)
Reboot DUT if necessary.
SystemRebootResponse
(Message)
(Reboot in progress)
Multicast Hello Message
(Reboot successful)
Receive and validate Hello Message
GetNetworkInterfacesRequest
(Empty)
GetNetworkInterfacesResponse
(modified settings)
Return Network Interfaces configuration
Receive and validate the message



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetNetworkInterfacesRequest message to retrieve the original settings of DUT.
4. ONVIF Client will invoke SetNetworkInterfacesRequest message to set static IPv4 address to DUT (InterfaceToken = available network interface, NetworkInterface.IPv4.Enabled = true, NetworkInterface.IPv4.Manual.Address = another address in the same subnet as an original address, NetworkInterface.IPv4.DHCP = false).
5. DUT will return SetNetworkInterfacesResponse message.
6. If necessary, ONVIF Client will invoke SystemReboot message to restart the DUT. Otherwise, go to step-8.
7. DUT will return SystemRebootResponse message.
8. DUT will send Multicast Hello message from a newly configured address.
9. ONVIF Client will receive and validate Hello message sent from a newly configured address by the DUT.
10. ONVIF Client will invoke GetNetworkInterfacesRequest message to a newly configured address to retrieve the modified settings of the DUT.
11. ONVIF Client will receive and validate GetNetworkInterfacesResponse message sent from a newly configured address by the DUT.
12. ONVIF Client will restore the original settings by following the procedure mentioned in Annex A.8.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetNetworkInterfacesResponse message.

The DUT did not send SystemRebootResponse message.

The DUT did not send Hello message after IP configuration change.

The DUT did not send GetNetworkInterfacesResponse message.

The DUT did not send correct network interface information (i.e. NetworkInterface [token = available network interface token, IPv4.Enabled = true, IPv4.Config.Manual = newly configured address, IPv4.DHCP = false]) in GetNetworkInterfacesResponse message.

4.1.2 IPV4 DHCP

Test Label: IP Configuration IPv4 DHCP Configuration

Test Case ID: IPCONFIG-1-1-3

ONVIF Core Specification Coverage: IP Configuration

Command Under Test: None

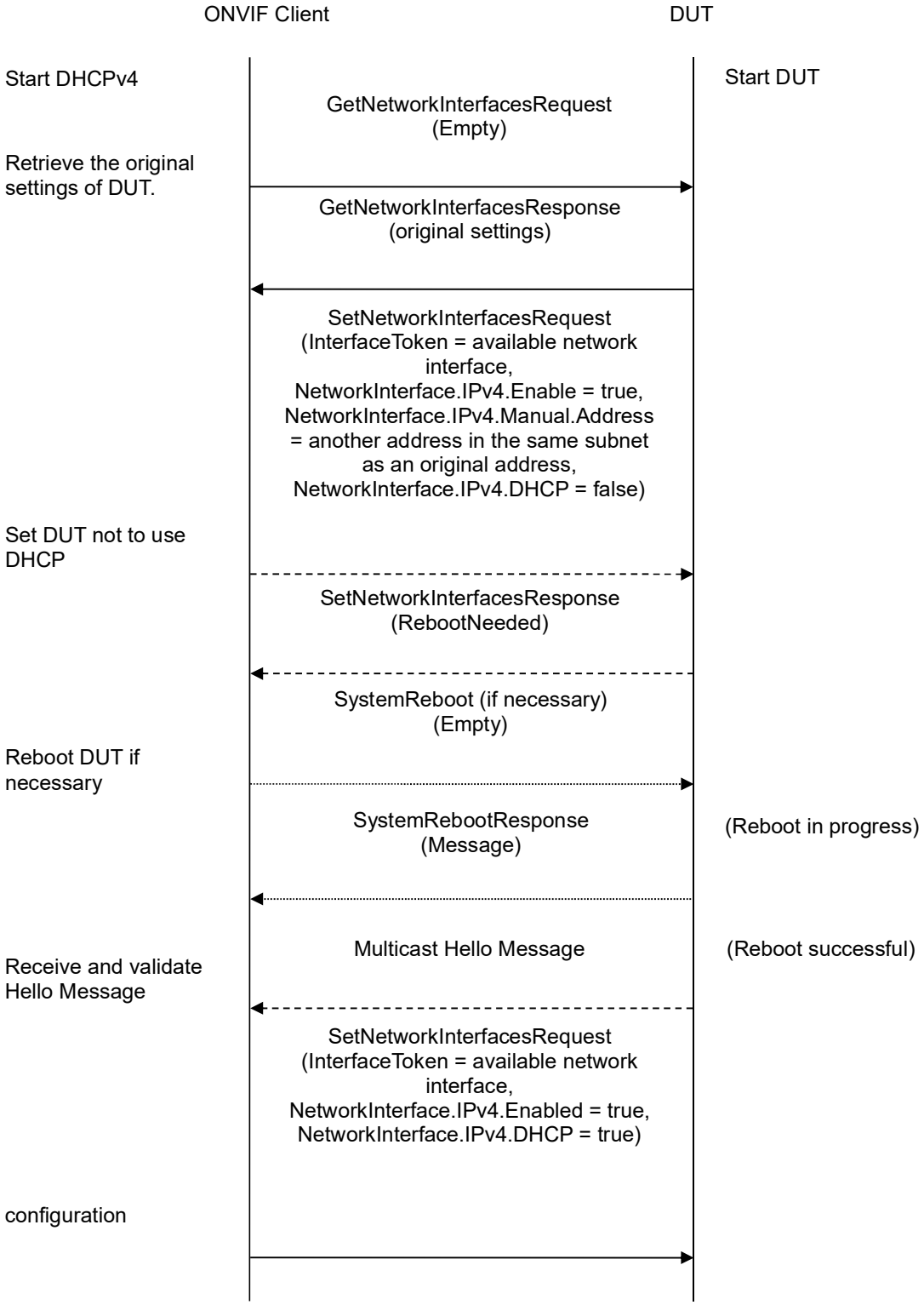
WSDL Reference: devicemgmt.wsdl

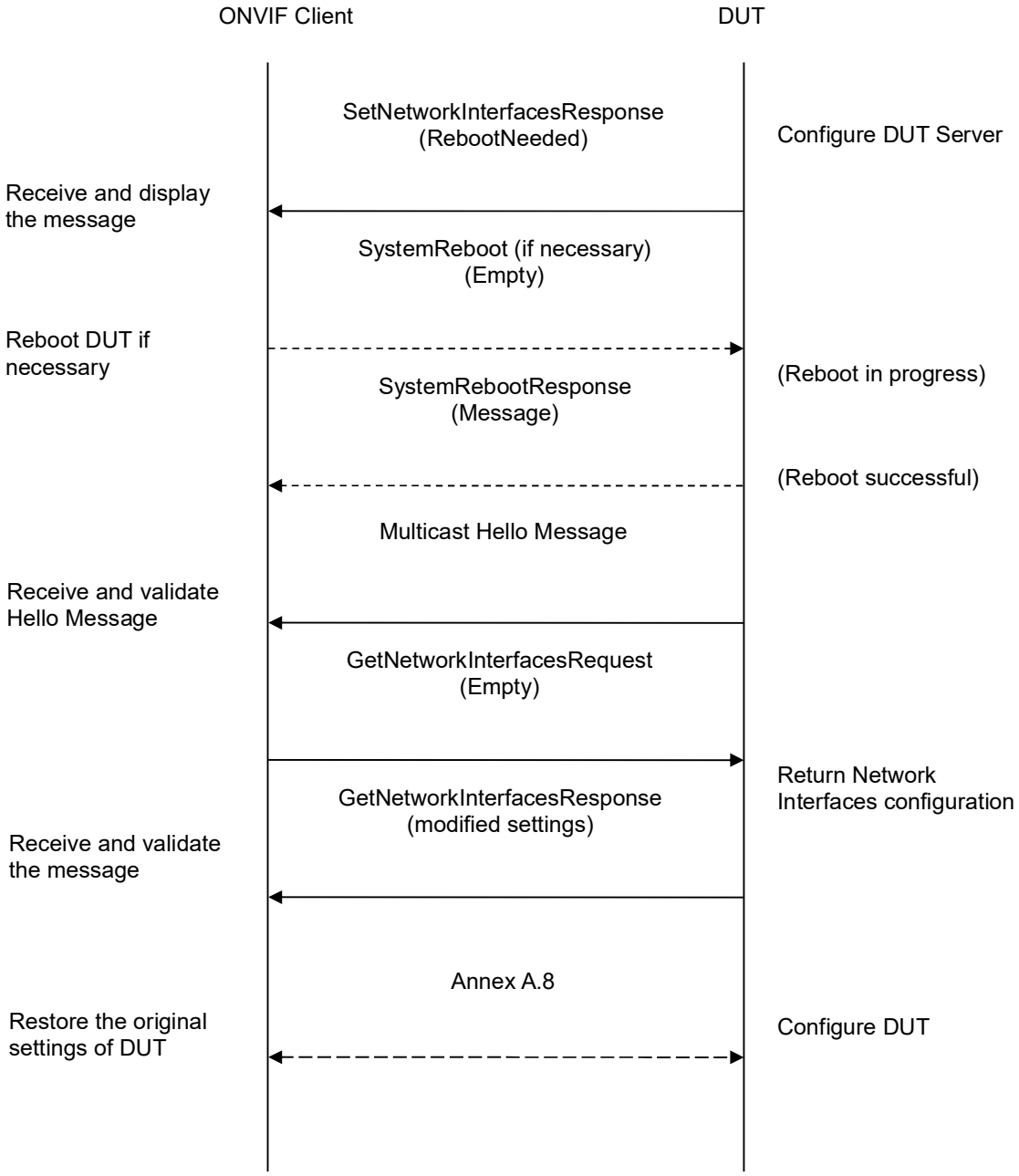
Test Purpose: To test IPv4 DHCP Configuration.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DHCPv4
Start DUT
GetNetworkInterfacesRequest
(Empty)
Retrieve the original settings of DUT.
GetNetworkInterfacesResponse
(original settings)
SetNetworkInterfacesRequest
(InterfaceToken = available network interface,
NetworkInterface.IPv4.Enable = true,
NetworkInterface.IPv4.Manual.Address = another address in the same subnet as an original address,
NetworkInterface.IPv4.DHCP = false)
Set DUT not to use DHCP
SetNetworkInterfacesResponse
(RebootNeeded)
SystemReboot (if necessary)
(Empty)
Reboot DUT if necessary
SystemRebootResponse
(Message)
(Reboot in progress)
Multicast Hello Message
(Reboot successful)
Receive and validate Hello Message
SetNetworkInterfacesRequest
(InterfaceToken = available network interface,
NetworkInterface.IPv4.Enabled = true,
NetworkInterface.IPv4.DHCP = true)
configuration



Test Procedure:

1. Start DHCPv4 server.
2. Start an ONVIF Client.
3. Start the DUT.

4. ONVIF Client will invoke GetNetworkInterfacesRequest message to retrieve the original settings of DUT.
5. If NetworkInterface.IPv4.DHCP == true in the original settings, ONVIF Client will invoke SetNetworkInterfacesRequest message to set DUT not to use DHCP (InterfaceToken = available network interface, NetworkInterface.IPv4.Enabled = true, NetworkInterface.IPv4.Manual.Address = another address in the same subnet as an original address, NetworkInterface.IPv4.DHCP = false, NetworkInterface.IPv6.Enable = false). Otherwise, continue to step-8.
6. If necessary, ONVIF Client will invoke SystemReboot message to restart DUT.
7. DUT will send Multicast Hello message from a newly configured address.
8. ONVIF Client will invoke SetNetworkInterfacesRequest message to set DUT to use DHCP (InterfaceToken = available network interface, NetworkInterface.IPv4.Enabled = true, NetworkInterface.IPv4.Manual = empty, NetworkInterface.IPv4.DHCP = true, NetworkInterface.IPv6.Enable = false).
9. DUT will return SetNetworkInterfacesResponse message.
10. If necessary, ONVIF Client will invoke SystemReboot message to restart DUT. Otherwise, go to step-12.
11. DUT will return SystemRebootResponse message.
12. DUT will send Multicast Hello message from a newly configured address.
13. ONVIF Client will receive and validate Hello message sent from a newly configured address by DUT.
14. ONVIF Client will invoke GetNetworkInterfacesRequest message to a newly configured address to retrieve the modified settings of DUT.
15. ONVIF Client will receive and validate GetNetworkInterfacesResponse message sent from a newly configured address by DUT.
16. If the modified settings are different from original settings then ONVIF Client will restore the original settings by following the procedure mentioned in Annex A.8.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetNetworkInterfacesResponse message.

The DUT did not send SystemRebootResponse message.

The DUT did not send Hello message after IP configuration change.

The DUT did not send GetNetworkInterfacesResponse message.

The DUT did not send correct network interface information (i.e. NetworkInterface [token = available network interface token, IPv4.Enabled = true, IPv4.Config.FromDHCP = new IP address]) in GetNetworkInterfacesResponse message.

4.1.3 IPV4 LINK LOCAL ADDRESS

Test Label: IP Configuration IPv4 Link-Local Address Configuration

Test Case ID: IPCONFIG-1-1-5

ONVIF Core Specification Coverage: IP Configuration Set zero configuration

Command Under Test: SetZeroConfiguration, GetZeroConfiguration

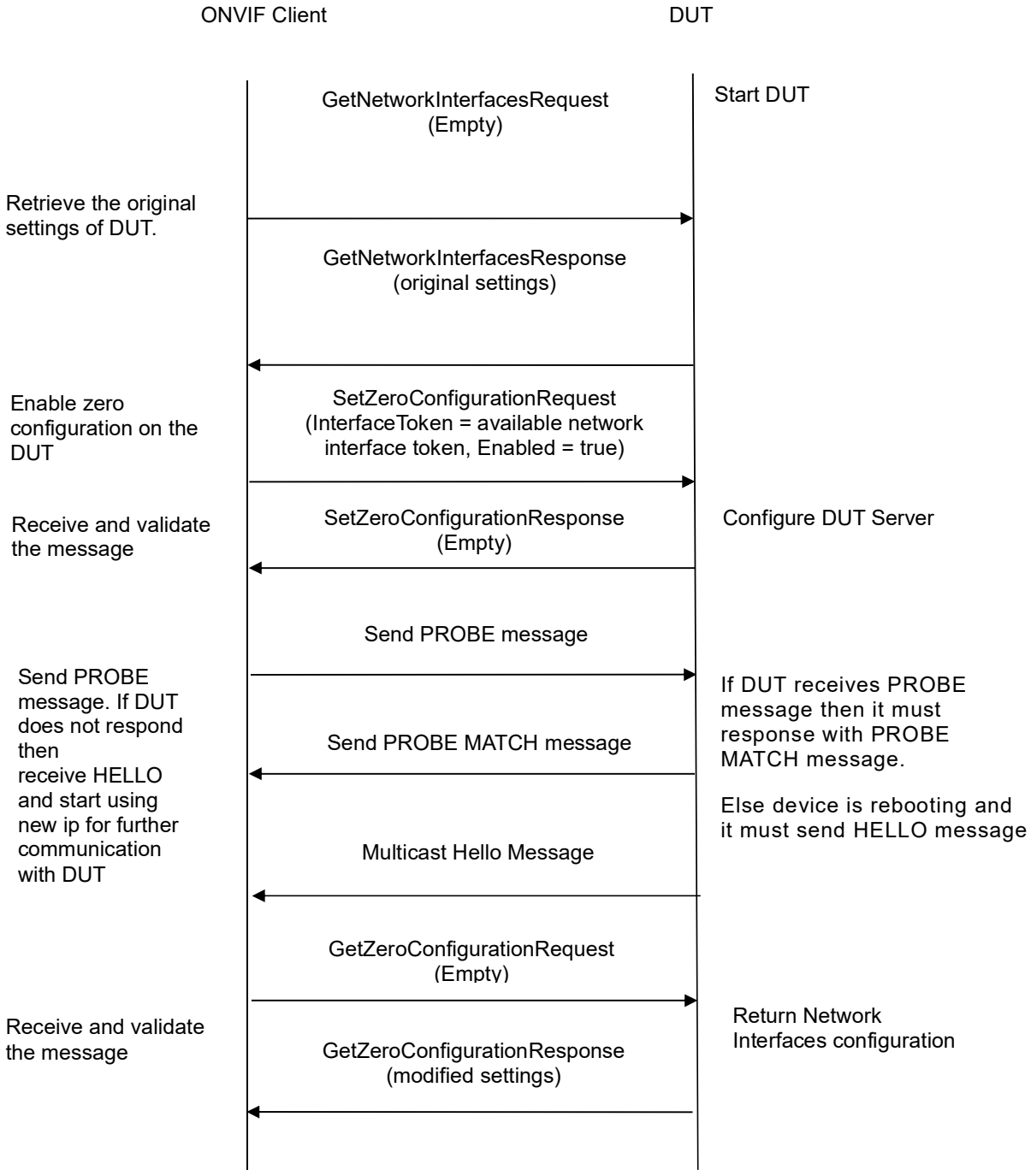
WSDL Reference: devicemgmt.wsdl

Test Purpose: To test IPv4 Link-Local Address Configuration.

Pre-Requisite: Dynamic IP configuration as per RFC 3927 is supported by DUT. Routable IPv4 address is configured.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

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

3. ONVIF Client will invoke GetNetworkInterfacesRequest message to retrieve the original settings of DUT.
4. DUT will return GetNetworkInterfacesResponse message with current settings of network interfaces.
5. Select Network Interface.
6. ONVIF Client will invoke GetZeroConfigurationRequest message to retrieve initial network zero configuration from the DUT.
7. ONVIF Client will invoke SetZeroConfigurationRequest message to set DUT to use dynamic IP configuration of IPv4 link-local address (InterfaceToken = networkInterface1, Enabled = true).
8. DUT will return SetZeroConfigurationResponse message.
9. ONVIF Client will wait operation delay timeout.
10. ONVIF Client will send PROBE message and if DUT will respond to this message then go to the step 12.
11. ONVIF Client will wait for Hello message sent from newly configured address by the DUT. Then ONVIF Client will start using this newly configured address for further communications with DUT.
12. ONVIF Client will invoke GetZeroConfigurationRequest message (InterfaceToken = networkInterface1) to a newly configured address to retrieve the modified settings of DUT.
13. ONVIF Client will receive and validate GetZeroConfigurationResponse message sent from a newly configured address by DUT.
14. ONVIF Client validates that GetZeroConfigurationResponse contains Enabled = true and Addresses contains Link Local address.
15. ONVIF Client will restore the original settings of network zero configuration if they were changed at step 7.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send **SetZeroConfigurationResponse** message

The DUT did not send **GetNetworkInterfacesResponse** message.

The DUT did not send **GetZeroConfigurationResponse** message.

The DUT did not send **GetZeroConfigurationResponse** message with Enable = true and with Link Local address at step 11.

4.2 IPv6

4.2.1 IPV6 STATIC IP

Test Label: IP Configuration IPv6 Static IP Configuration

Test Case ID: IPCONFIG-2-1-1

ONVIF Core Specification Coverage: IP Configuration

Command Under Test: None

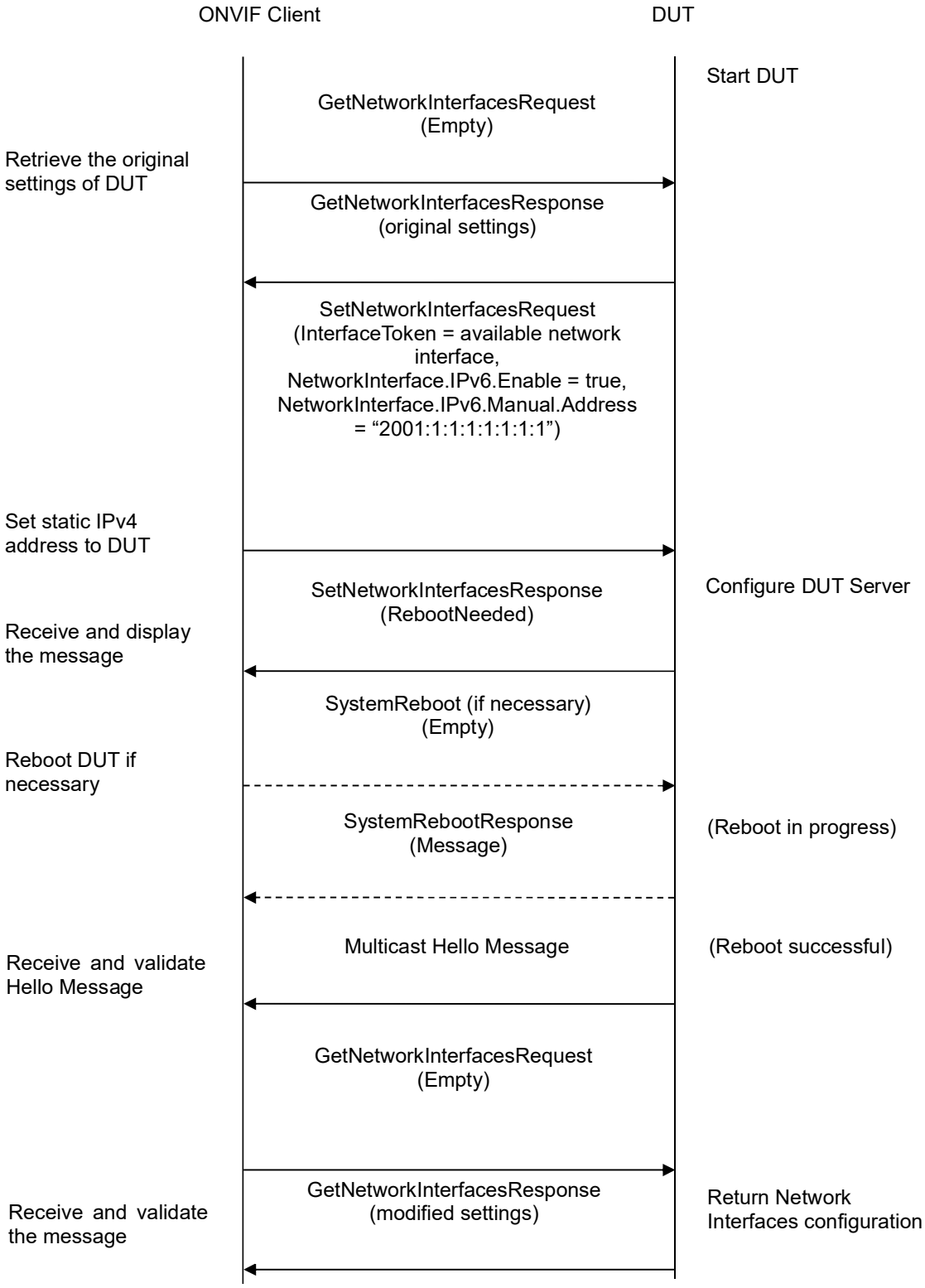
WSDL Reference: devicemgmt.wsdl

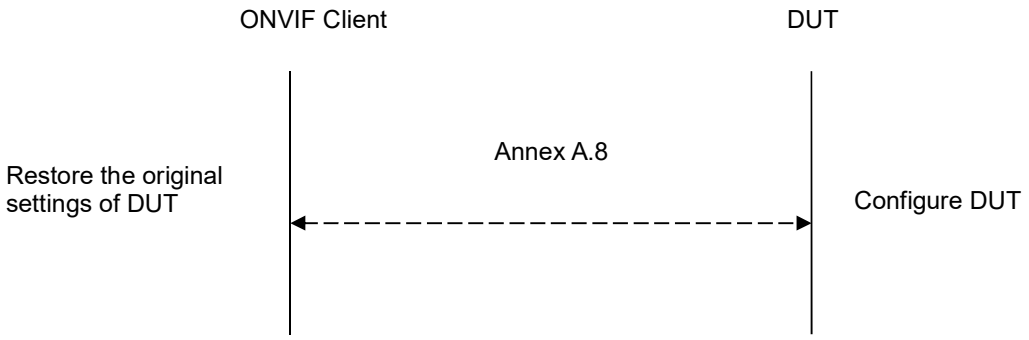
Test Purpose: To test IPv6 Static IP Configuration.

Pre-Requisite: IPv6 is implemented by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
GetNetworkInterfacesRequest
(Empty)
Retrieve the original settings of DUT
GetNetworkInterfacesResponse
(original settings)
SetNetworkInterfacesRequest
(InterfaceToken = available network interface,
NetworkInterface.IPv6.Enable = true,
NetworkInterface.IPv6.Manual.Address = “2001:1:1:1:1:1:1:1”)
Set static IPv4 address to DUT
SetNetworkInterfacesResponse
(RebootNeeded)
Configure DUT Server
Receive and display the message
SystemReboot (if necessary)
(Empty)
Reboot DUT if necessary
SystemRebootResponse
(Message)
(Reboot in progress)
Multicast Hello Message
(Reboot successful)
Receive and validate Hello Message
GetNetworkInterfacesRequest
(Empty)
GetNetworkInterfacesResponse
(modified settings)
Return Network Interfaces configuration
Receive and validate the message



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetNetworkInterfacesRequest message to retrieve the original settings of the DUT.
4. ONVIF Client will invoke SetNetworkInterfacesRequest message to set static IPv6 address to the DUT (InterfaceToken = available network interface, NetworkInterface.IPv6.Enabled = true, NetworkInterface.IPv6.Manual.Address = “2001:1:1:1:1:1:1:1”).
5. The DUT will return SetNetworkInterfacesResponse message.
6. If necessary, ONVIF Client will invoke SystemReboot message to restart the DUT. Otherwise, go to step 8.
7. The DUT will return SystemRebootResponse message.
8. The DUT will send Multicast Hello message from a newly configured address.
9. ONVIF Client will receive and validate Hello message sent from a newly configured address by DUT.
10. ONVIF Client will invoke GetNetworkInterfacesRequest message to a newly configured address to retrieve the modified settings of the DUT.
11. ONVIF Client will receive and validate GetNetworkInterfacesResponse message sent from a newly configured address by DUT.
12. ONVIF Client will restore the original settings by following the procedure mentioned in Annex A.8.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetNetworkInterfacesResponse message.

The DUT did not send SystemRebootResponse message.

The DUT did not send Hello message after IP configuration change.

The DUT did not send GetNetworkInterfacesResponse message.

The DUT did not send correct network interface information (i.e. NetworkInterface [token = available network interface token, IPv6.Enabled = true, IPv6.Config.Manual = “2001:1:1:1:1:1:1:1”]) in GetNetworkInterfacesResponse message.

4.2.2 IPV6 STATELESS IP CONFIGURATION - ROUTER ADVERTISEMENT

Test Label: IP Configuration IPv6 Stateless IP Configuration Which Accepts Router Advertisement.

Test Case ID: IPCONFIG-2-1-2

ONVIF Core Specification Coverage: IP Configuration

Command Under Test: None

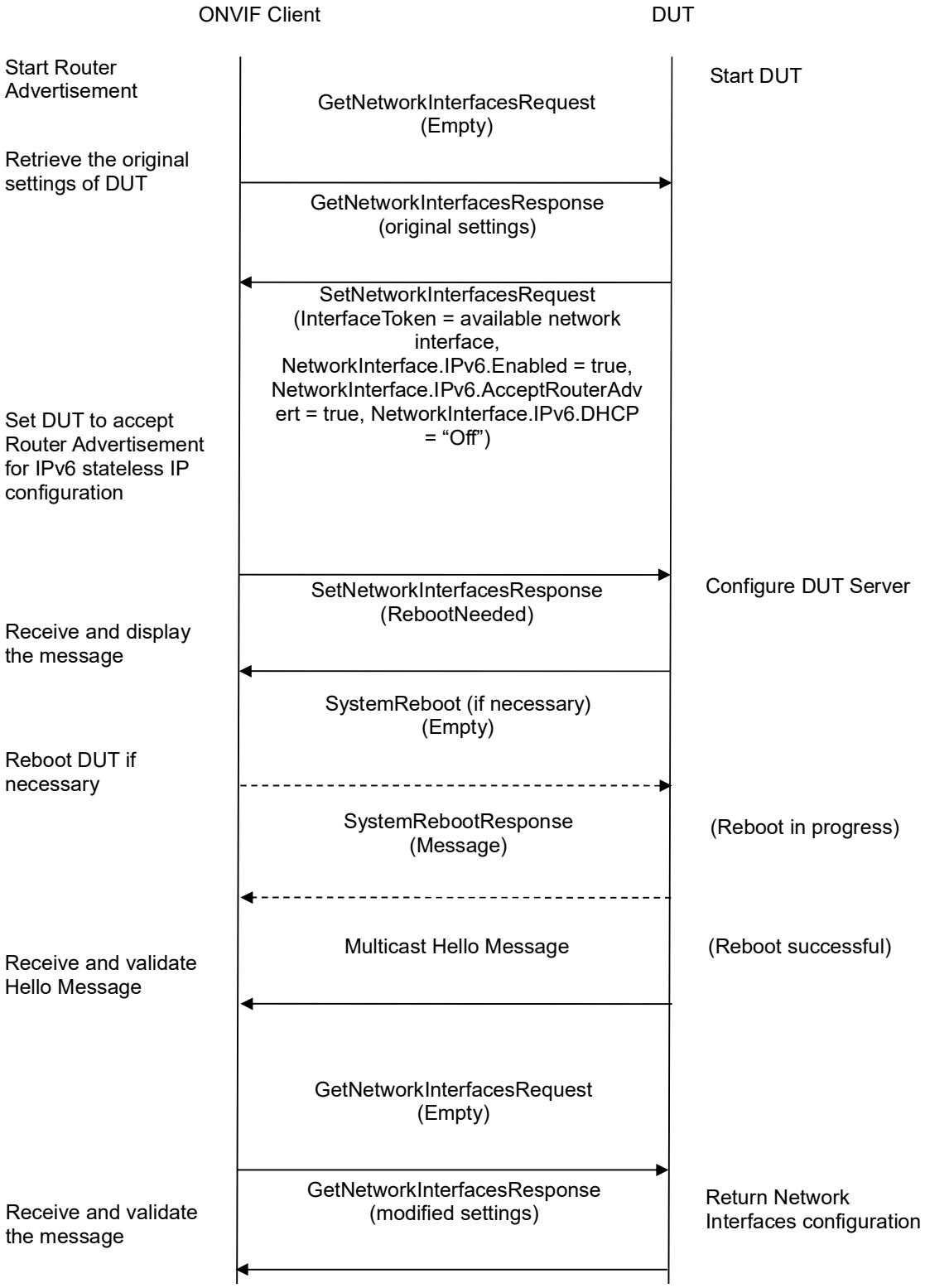
WSDL Reference: devicemgmt.wsdl

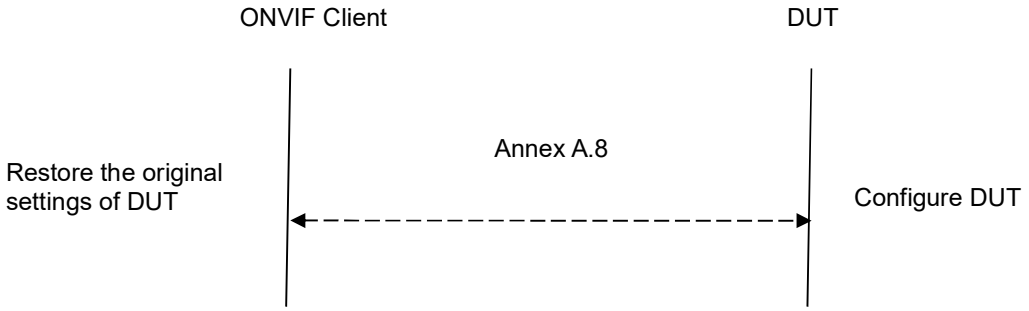
Test Purpose: To test IPv6 Stateless IP Configuration which Accepts Router Advertisement.

Pre-Requisite: IPv6 is implemented by the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start Router Advertisement
Start DUT
GetNetworkInterfacesRequest (Empty)
Retrieve the original settings of DUT
GetNetworkInterfacesResponse (original settings)
SetNetworkInterfacesRequest (InterfaceToken = available network interface, NetworkInterface.IPv6.Enabled = true, NetworkInterface.IPv6.AcceptRouterAdvert = true, NetworkInterface.IPv6.DHCP = “Off”)
Set DUT to accept Router Advertisement for IPv6 stateless IP configuration
SetNetworkInterfacesResponse (RebootNeeded)
Configure DUT Server
Receive and display the message
SystemReboot (if necessary) (Empty)
Reboot DUT if necessary
SystemRebootResponse (Message)
(Reboot in progress)
Multicast Hello Message
(Reboot successful)
Receive and validate Hello Message
GetNetworkInterfacesRequest (Empty)
GetNetworkInterfacesResponse (modified settings)
Return Network Interfaces configuration
Receive and validate the message



Test Procedure:

1. Start Router Advertisement
2. Start an ONVIF Client.
3. Start the DUT
4. ONVIF Client will invoke GetNetworkInterfacesRequest message to retrieve the original settings of DUT
5. ONVIF Client will invoke SetNetworkInterfacesRequest message to set DUT accept Router Advertisement for IPv6 stateless IP configuration (InterfaceToken = available network interface, NetworkInterface.IPv6.Enabled = true, NetworkInterface.IPv6.AcceptRouterAdvert = true, NetworkInterface.IPv6.DHCP = "Off").
6. The DUT will return SetNetworkInterfacesResponse message.
7. If necessary, ONVIF Client will invoke SystemReboot message to restart DUT. Otherwise, continue to step-9.
8. The DUT will return SystemRebootResponse message.
9. The DUT will send Multicast Hello message from newly configured address.
10. ONVIF Client will receive and validate Hello message sent from newly configured address by DUT.
11. ONVIF Client will invoke GetNetworkInterfacesRequest message to newly configured address to retrieve the modified settings of DUT.
12. ONVIF Client will receive and validate GetNetworkInterfacesResponse message sent from newly configured address by DUT.
13. ONVIF Client will restore the original settings by following the procedure mentioned in Annex A.8.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetNetworkInterfacesResponse message.

The DUT did not send SystemRebootResponse message.

The DUT did not send Hello message after IP configuration change.

The DUT did not send GetNetworkInterfacesResponse message.

The DUT did not send correct network interface information (i.e. NetworkInterface [token = available network interface token, IPv6.Enabled = true, IPv6.Config.FromRA = new IP address]) in GetNetworkInterfacesResponse message.

4.2.3 IPV6 STATELESS IP CONFIGURATION - NEIGHBOUR DISCOVERY

Test Label: IP Configuration IPv6 Stateless IP Configuration Which Uses Neighbor Discovery.

Test Case ID: IPCONFIG-2-1-3

ONVIF Core Specification Coverage: IP Configuration

Command Under Test: None

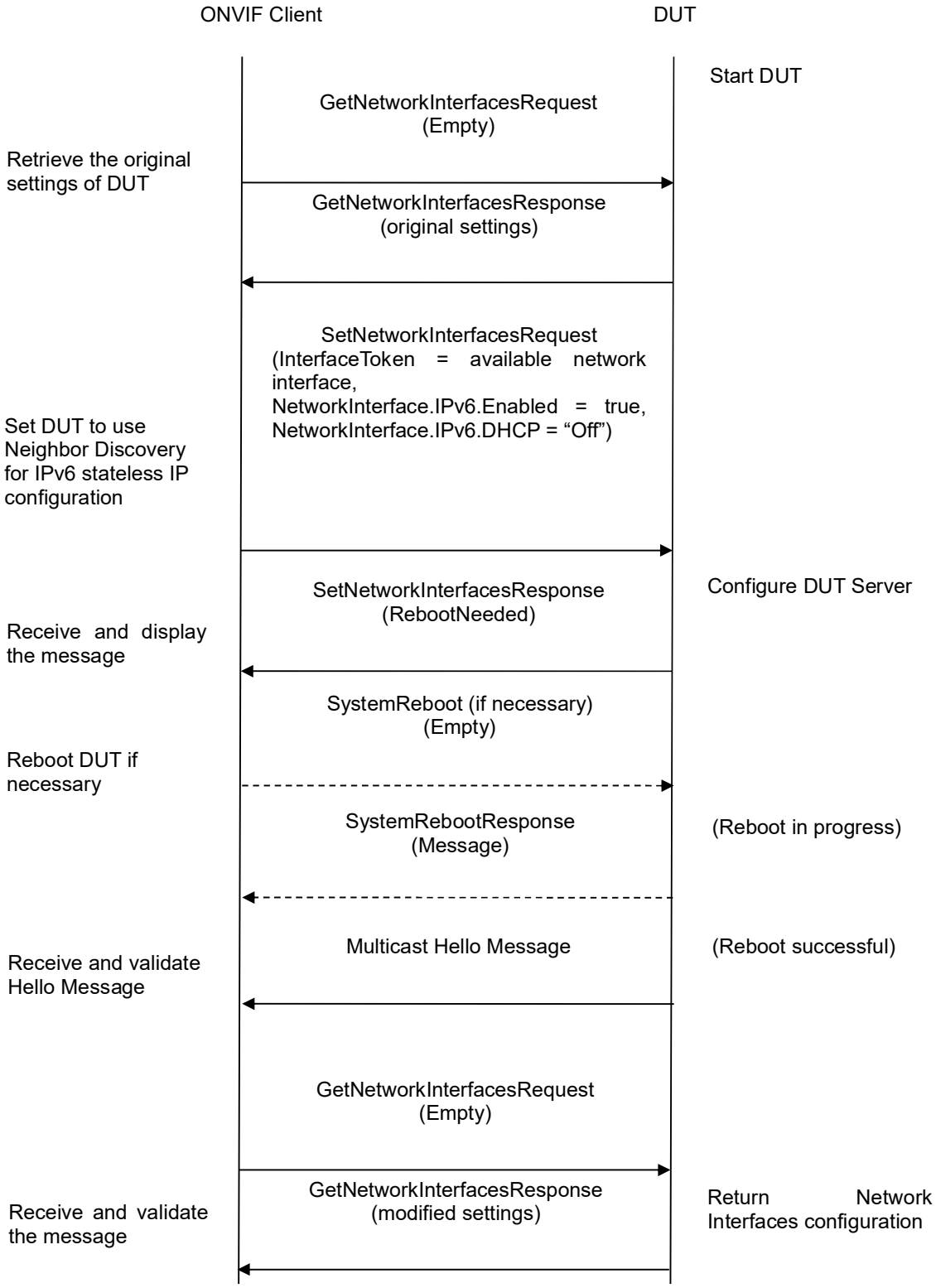
WSDL Reference: devicemgmt.wsdl

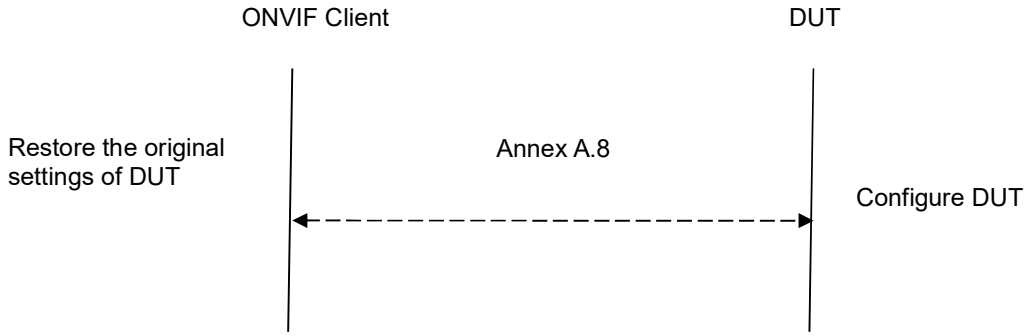
Test Purpose: To test IPv6 Stateless IP Configuration which uses Neighbor Discovery.

Pre-Requisite: IPv6 is implemented by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
GetNetworkInterfacesRequest
(Empty)
Retrieve the original settings of DUT
GetNetworkInterfacesResponse
(original settings)
SetNetworkInterfacesRequest
(InterfaceToken = available network interface,
NetworkInterface.IPv6.Enabled = true,
NetworkInterface.IPv6.DHCP = “Off”)
Set DUT to use Neighbor Discovery for IPv6 stateless IP configuration
SetNetworkInterfacesResponse
(RebootNeeded)
Configure DUT Server
Receive and display the message
SystemReboot (if necessary)
(Empty)
Reboot DUT if necessary
SystemRebootResponse
(Message)
(Reboot in progress)
Multicast Hello Message
(Reboot successful)
Receive and validate Hello Message
GetNetworkInterfacesRequest
(Empty)
GetNetworkInterfacesResponse
(modified settings)
Return Network Interfaces configuration
Receive and validate the message



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetNetworkInterfacesRequest message to retrieve the original settings of DUT.
4. ONVIF Client will invoke SetNetworkInterfacesRequest message to set DUT to use Neighbor Discovery for IPv6 stateless IP configuration (InterfaceToken = available network interface, NetworkInterface.IPv6.Enabled = true, NetworkInterface.IPv6.DHCP = "Off").
5. DUT will return SetNetworkInterfacesResponse message.
6. If necessary, ONVIF Client will invoke SystemReboot message to restart DUT. Otherwise, go to step-8.
7. DUT will return SystemRebootResponse message.
8. DUT will send Multicast Hello message from a newly configured address.
9. ONVIF Client will receive and validate Hello message sent from a newly configured address by the DUT.
10. ONVIF Client will invoke GetNetworkInterfacesRequest message to a newly configured address to retrieve the modified settings of DUT.
11. ONVIF Client will receive and validate GetNetworkInterfacesResponse message sent from a newly configured address by DUT.
12. ONVIF Client will restore the original settings by following the procedure mentioned in Annex A.8.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetNetworkInterfacesResponse message.

The DUT did not send SystemRebootResponse message.

The DUT did not send Hello message after IP configuration change.

The DUT did not send GetNetworkInterfacesResponse message.

The DUT did not send correct network interface information (i.e. NetworkInterface [token = available network interface token, IPv6.Enabled = true, IPv6.Config.LinkLocal = new IP address]) in GetNetworkInterfacesResponse message.

4.2.4 IPV6 STATEFUL IP CONFIGURATION

Test Label: IP Configuration IPv6 Stateful IP Configuration (DHCPv6)

Test Case ID: IPCONFIG-2-1-4

ONVIF Core Specification Coverage: IP Configuration

Command Under Test: None

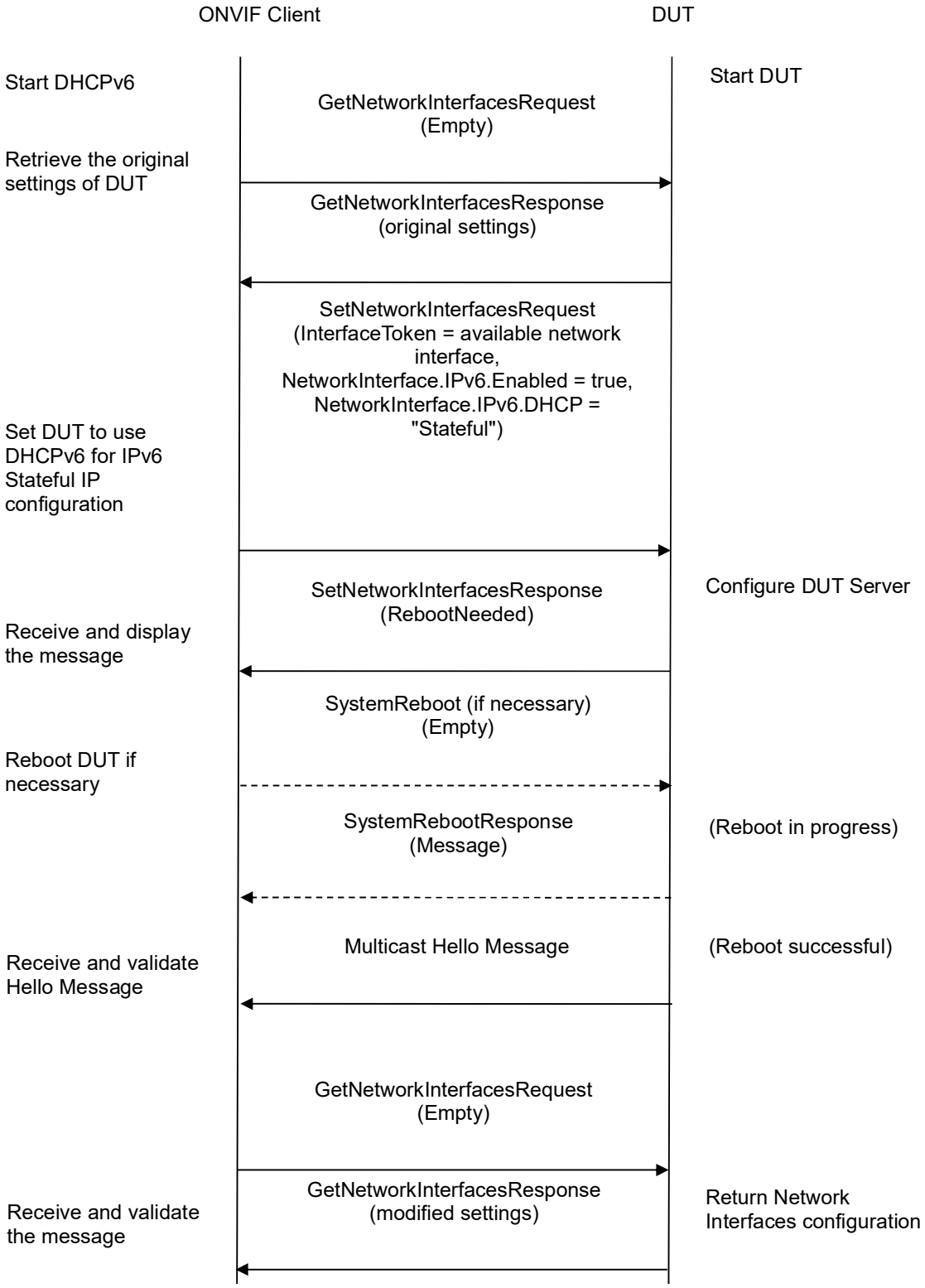
WSDL Reference: devicemgmt.wsdl

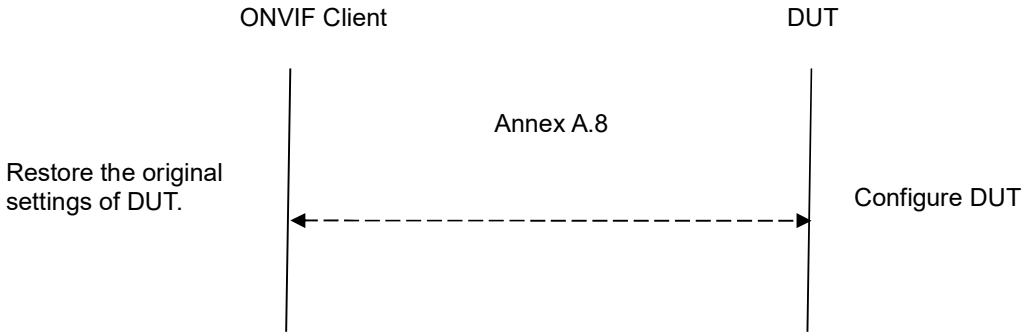
Test Purpose: To test IPv6 Stateful IP Configuration (DHCPv6)

Pre-Requisite: IPv6 is implemented by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DHCPv6
Start DUT
GetNetworkInterfacesRequest
(Empty)
Retrieve the original settings of DUT
GetNetworkInterfacesResponse
(original settings)
SetNetworkInterfacesRequest
(InterfaceToken = available network interface,
NetworkInterface.IPv6.Enabled = true,
NetworkInterface.IPv6.DHCP = "Stateful")
Set DUT to use DHCPv6 for IPv6 Stateful IP configuration
SetNetworkInterfacesResponse
(RebootNeeded)
Configure DUT Server
Receive and display the message
SystemReboot (if necessary)
(Empty)
Reboot DUT if necessary
SystemRebootResponse
(Message)
(Reboot in progress)
Multicast Hello Message
(Reboot successful)
Receive and validate Hello Message
GetNetworkInterfacesRequest
(Empty)
GetNetworkInterfacesResponse
(modified settings)
Return Network Interfaces configuration
Receive and validate the message



Test Procedure:

1. Start DHCPv6 server.
2. Start an ONVIF Client.
3. Start the DUT.
4. ONVIF Client will invoke GetNetworkInterfacesRequest message to retrieve the original settings of DUT.
5. ONVIF Client will invoke SetNetworkInterfacesRequest message to set DUT to accept Router Advertisement for IPv6 stateful IP configuration (InterfaceToken = available network interface, NetworkInterface.IPv6.Enabled = true, NetworkInterface.IPv6.DHCP = "Stateful").
6. The DUT will return SetNetworkInterfacesResponse message.
7. If necessary, ONVIF Client will invoke SystemReboot message to restart DUT. Otherwise, go to step 9.
8. The DUT will return SystemRebootResponse message.
9. The DUT will send Multicast Hello message from a newly configured address.
10. ONVIF Client will receive and validate Hello message sent from a newly configured address by the DUT.
11. ONVIF Client will invoke GetNetworkInterfacesRequest message to a newly configured address to retrieve the modified settings of the DUT.
12. ONVIF Client will receive and validate GetNetworkInterfacesResponse message sent from a newly configured address by the DUT.
13. ONVIF Client will restore the original settings by following the procedure mentioned in Annex A.8.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetNetworkInterfacesResponse message.

The DUT did not send SystemRebootResponse message.

The DUT did not send Hello message after IP configuration change.

The DUT did not send GetNetworkInterfacesResponse message.

The DUT did not send correct network interface information (i.e. NetworkInterface [token = available network interface token, IPv6.Enabled = true, IPv6.Config.FromDHCP = new IP address]) in GetNetworkInterfacesResponse message.

5 Device Discovery Test Cases

This section covers tests designed for ONVIF Device Discovery Feature.

5.1 HELLO MESSAGE

Test Label: Device Discovery Multicast HELLO Message Transmission.

Test Case ID: DISCOVERY-1-1-1

ONVIF Core Specification Coverage: Hello, Reboot

Command Under Test: Hello

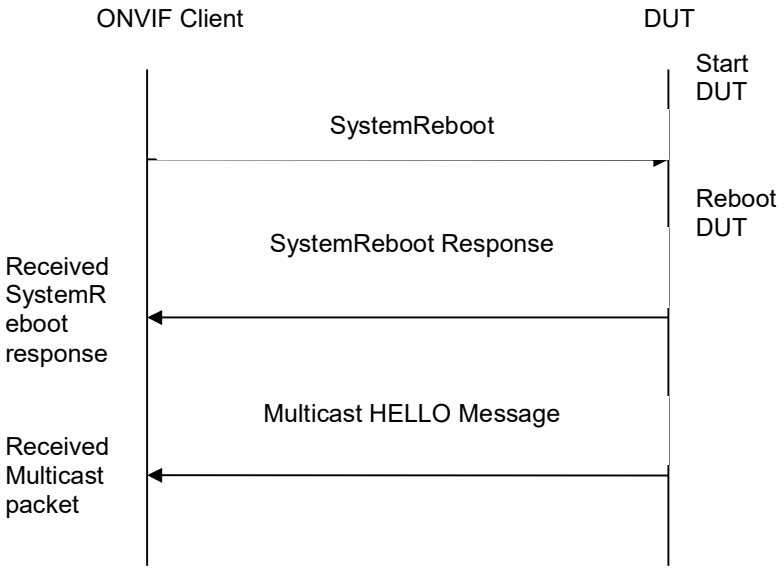
WSDL Reference: ws-discovery.wsdl, devicemgmt.wsdl

Test Purpose: To verify that the DUT transmits HELLO message with the correct multicast parameters (address, and port number) when it is connected to the network.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client
2. Start the DUT
3. ONVIF Client invokes SystemReboot message to reboot the DUT.
4. DUT sends SystemRebootResponse message.
5. ONVIF Client waits for the user-defined boot time to receive HELLO message from DUT.

6. Verify that the DUT transmits the HELLO message with multicast address 239.255.255.250, and port number 3702.
7. In case of device discovery with IPv6 interface, verify that the DUT transmits the HELLO message with multicast address FF02::C (link-local scope), and port number 3702.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SystemRebootResponse message.

The DUT did not send the multicast HELLO message.

5.2 HELLO MESSAGE VALIDATION

Test Label: Device Discovery HELLO Message Validation

Test Case ID: DISCOVERY-1-1-2

ONVIF Core Specification Coverage: Endpoint reference, Hello, Types, Scopes, Reboot

Command Under Test: Hello

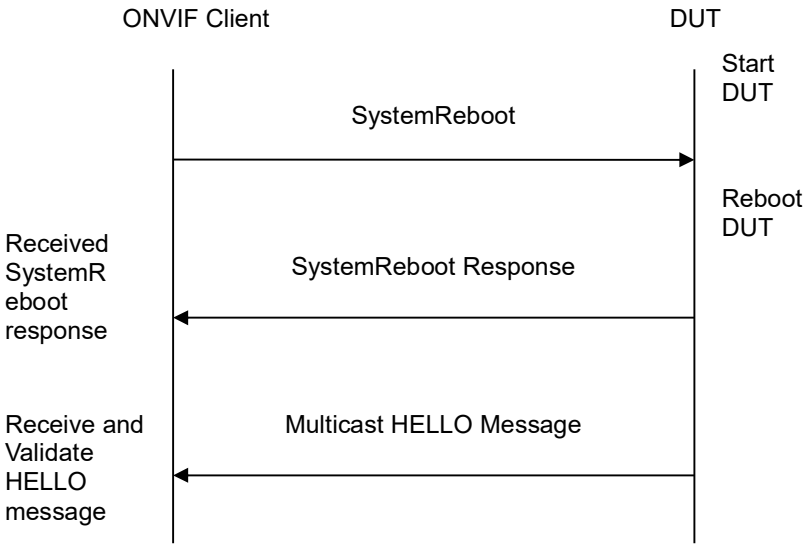
WSDL Reference: ws-discovery.wsdl, devicemgmt.wsdl

Test Purpose: To verify the mandatory XML elements Device type, Scope types, Endpoint Reference and Meta data version in the HELLO message.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes SystemReboot message to reboot the DUT.
4. The DUT sends SystemRebootResponse message.
5. ONVIF Client waits for the user defined boot time to receive HELLO message from DUT.
6. ONVIF Client will verify the mandatory XML elements in the DUT HELLO message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SystemRebootResponse message.

The DUT did not send multicast HELLO message.

The DUT did not send HELLO message with one or more mandatory XML elements (EndpointReference, Types, and Scopes).

The DUT did not send HELLO message with mandatory device type and scope types (type, location, hardware and name).

The DUT did not send HELLO message with a namespace of the Types value.

Note: See Annex A.1 for Device and Scope Types definition.

5.3 SEARCH BASED ON DEVICE SCOPE TYPES

Test Label: Device Discovery Search based on device scope types.

Test Case ID: DISCOVERY-1-1-3

ONVIF Core Specification Coverage: Services overview, Types, Scopes, Probe and Probe Match, Get scope parameters

Command Under Test: Probe, ProbeMatch

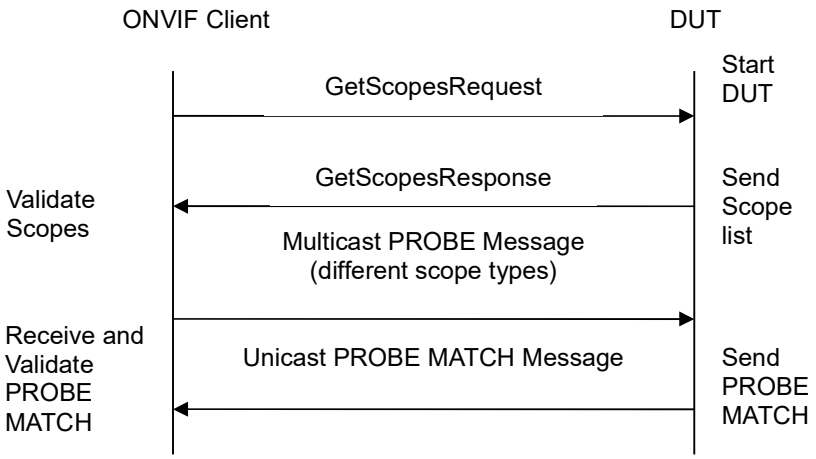
WSDL Reference: ws-discovery.wsdl, devicemgmt.wsdl

Test Purpose: To search the DUT based on the mandatory scope types (type, location, hardware and name).

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT
3. ONVIF Client will invoke GetScopesRequest message to retrieve existing scopes list.
4. DUT replies with the list of scopes types in GetScopesResponse message.
5. ONVIF Client will transmit the multicast PROBE message with different scope types (type, location, hardware and name).
6. ONVIF Client will verify the PROBE MATCH message sent by DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetScopesResponse message.

The DUT scope list does not have one or more mandatory scope entries.

The DUT did not send mandatory XML elements (device type, scope list, service address and scope matching rule) in the PROBE MATCH message.

The DUT did not send PROBE MATCH message within the time out period of DISCOVERY_TIMEOUT.

The DUT did not send PROBE MATCH message with a namespace of the Types value.

The DUT did not send PROBE MATCH message with fixed entry point to <d:XAddrs> element (“http://<onvif_host>/onvif/device_service”).

In case of IPv6, the DUT did not send PROBE MATCH message with <d:XAddrs> including IPv6 address.

The DUT did not send PROBE MATCH message with a correct XML element RelatesTo that has the same value as MessageID of PROBE message (not to omit "urn" namespace).

Note: See Annex A.1 for Device and Scope Types definition, Annex A.7 for the value of DISCOVERY_TIMEOUT.

5.4 SEARCH WITH OMITTED DEVICE AND SCOPE TYPES

Test Label: Device Discovery Search with omitted device type and scope types.

Test Case ID: DISCOVERY-1-1-4

ONVIF Core Specification Coverage: Services overview, Types, Scopes, Probe and Probe Match

Command Under Test: Probe, ProbeMatch

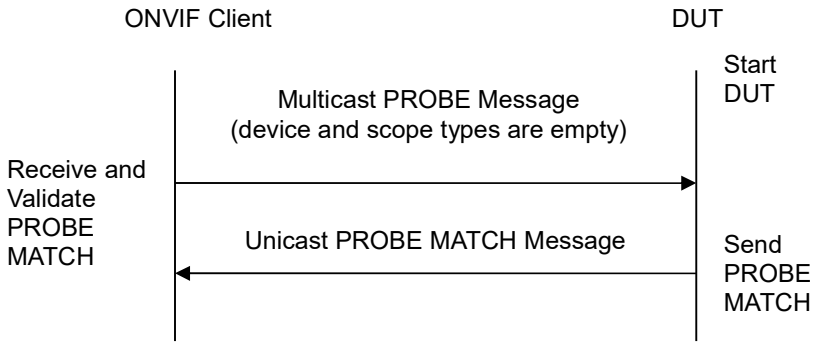
WSDL Reference: ws-discovery.wsdl

Test Purpose: To search the DUT with device and scope types being omitted.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will transmit multicast PROBE message with device type and scope type inputs omitted.
4. ONVIF Client will verify the PROBE MATCH message sent by the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send PROBE MATCH message within the time out period of DISCOVERY_TIMEOUT.

The DUT did not send mandatory XML elements (device type, scope list, service address and scope matching rule) in the PROBE MATCH message.

The DUT did not send PROBE MATCH message with a namespace of the Types value.

The DUT did not send PROBE MATCH message with fixed entry point to <d:XAddrs> element (“http://<onvif_host>/onvif/device_service”).

In case of IPv6, the DUT did not send PROBE MATCH message with <d:XAddrs> of including IPv6 address.

The DUT did not send PROBE MATCH message with a correct XML element RelatesTo that it has same value as MessageID of PROBE message (not to omit "urn" namespace).

Note: See Annex A.1 for Device and Scope Types definition, Annex A.7 for the value of DISCOVERY_TIMEOUT.

5.5 RESPONSE TO INVALID SEARCH REQUEST

Test Label: Device Discovery DUT does not respond to invalid multicast PROBE message.

Test Case ID: DISCOVERY-1-1-5

ONVIF Core Specification Coverage: Probe and Probe Match

Command under Test: Probe

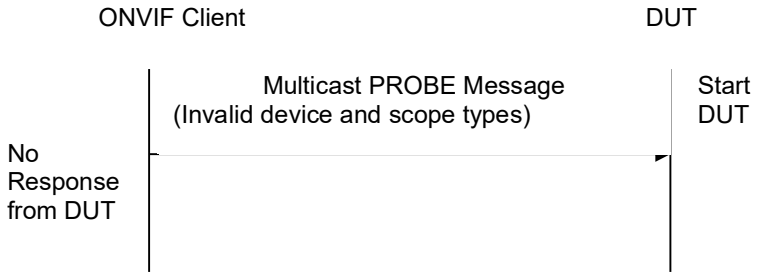
WSDL Reference: ws-discovery.wsdl

Test Purpose: To verify that the DUT does not reply to the invalid multicast PROBE message (invalid device and scope types).

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will transmit multicast PROBE message with invalid device and scope types.
4. Verify that the DUT did not send PROBE MATCH message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did send PROBE MATCH message.

Note: See Annex A.1 for Invalid Device and Scope Types definition.

5.6 SEARCH USING UNICAST PROBE MESSAGE

Test Label: Device Discovery Search by Unicast PROBE message.

Test Case ID: DISCOVERY-1-1-6

Test Purpose: To verify DUT behavior for Unicast PROBE message.

Note: All Tests 5.3, 5.4, 5.5 to be repeated with Unicast PROBE message.

5.7 BYE MESSAGE

Test Label: Device Discovery BYE Message Transmission.

Test Case ID: DISCOVERY-1-1-8

ONVIF Core Specification Coverage: Bye, Reboot

Command Under Test: Bye

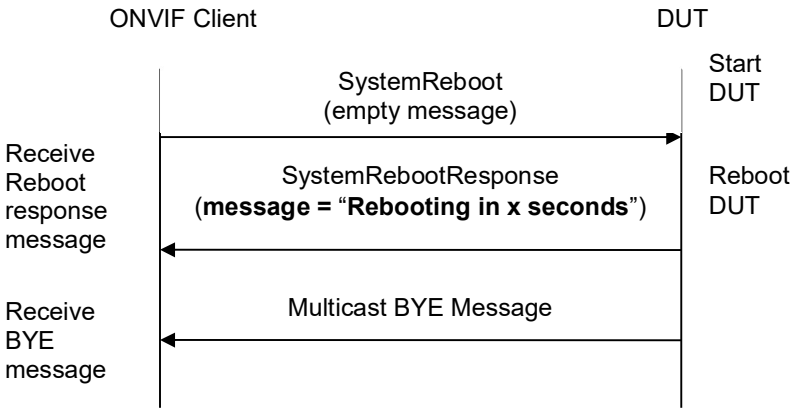
WSDL Reference: ws-discovery.wsdl, devicemgmt.wsdl

Test Purpose: To verify that the DUT transmits BYE message before the system reboot.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke SystemReboot message to reboot the DUT.
4. Verify that the DUT sends SystemRebootResponse message (example message string = "Rebooting in x seconds").
5. Verify that the DUT issued a BYE message.
6. ONVIF Client waits for the user defined boot time before proceeding to execute the next test case.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SystemRebootResponse message.

The DUT did not send BYE message.

5.8 DISCOVERY MODE CONFIGURATION

Test Label: Device Discovery mode configurations.

Test Case ID: DISCOVERY-1-1-9

ONVIF Core Specification Coverage: Probe and Probe Match, Reboot, Get discovery mode, Set

discovery mode

Command Under Test: GetDiscoveryMode, SetDiscoveryMode

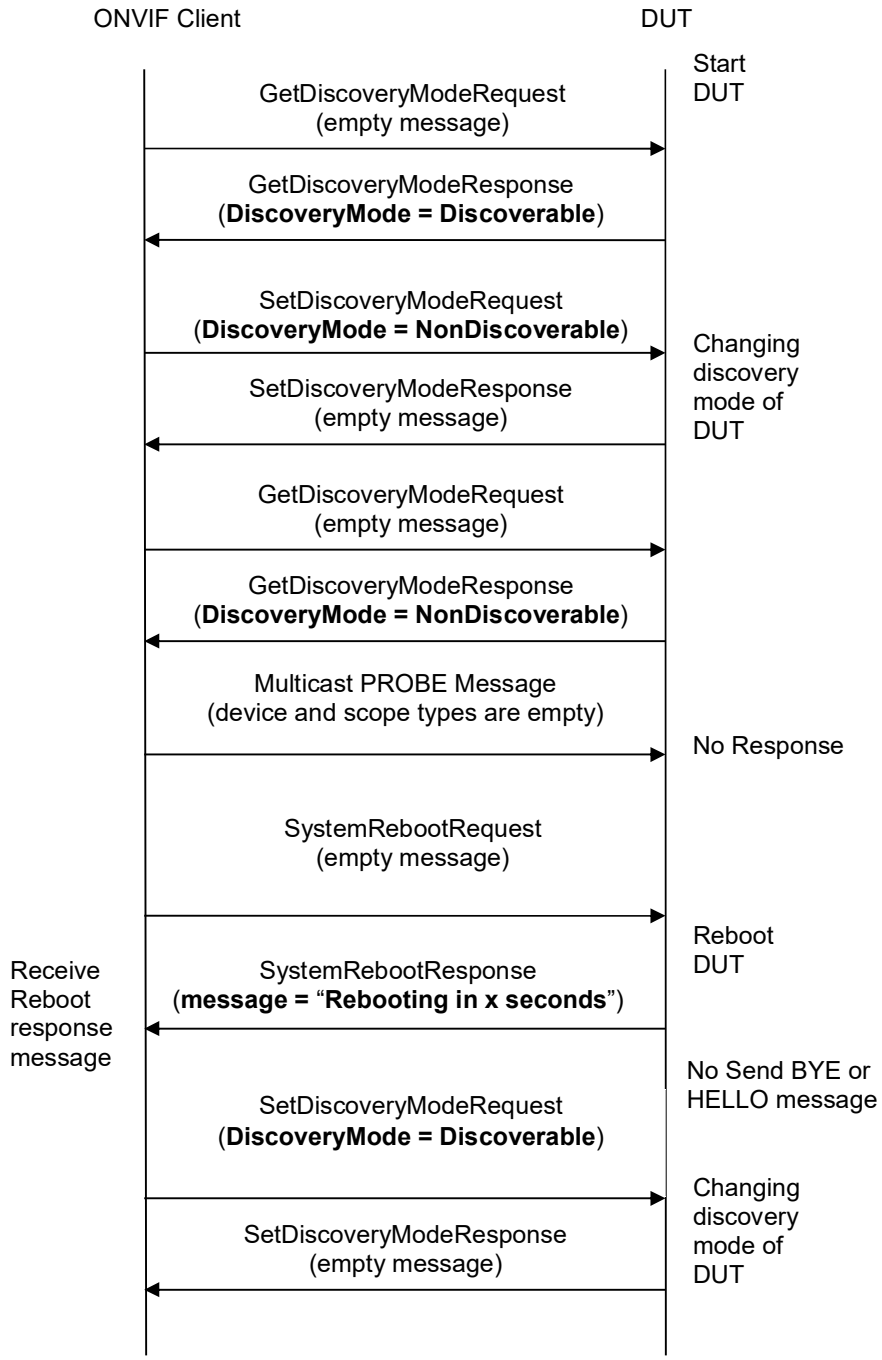
WSDL Reference: ws-discovery.wsdl, devicemgmt.wsdl

Test Purpose: To verify DUT behavior for Discovery mode configuration.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetDiscoveryMode message to verify discovery mode of the DUT.

4. Verify that the DUT sends GetDiscoveryModeResponse message (DiscoveryMode = Discoverable).
5. ONVIF Client will invoke SetDiscoveryMode message to set discovery mode of the DUT to Non-Discoverable.
6. Verify that the DUT sends SetDiscoveryModeResponse message.
7. ONVIF Client will invoke GetDiscoveryMode message to verify discovery mode of the DUT.
8. Verify that the DUT sends GetDiscoveryModeResponse message (DiscoveryMode = NonDiscoverable).
9. ONVIF Client will transmit multicast PROBE message with device type and scope type inputs omitted.
10. Verify that the DUT did not send PROBE MATCH message.
11. ONVIF Client will invoke SystemReboot message to reboot the DUT.
12. Verify that the DUT sends SystemRebootResponse message (example message string = “Rebooting in x seconds”).
13. Verify that the DUT did not send BYE or HELLO message.
14. ONVIF Client waits for the user defined boot time before proceeding to execute the next step.
15. ONVIF Client will invoke SetDiscoveryMode message to set discovery mode of the DUT to Discoverable.
16. Verify that the DUT sends SetDiscoveryModeResponse message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetDiscoveryModeResponse message.

The DUT GetDiscoveryModeResponse did not have a Discovery mode parameter of Discoverable after executing Test Procedure 3.

The DUT did not send SetDiscoveryModeResponse message.

The DUT GetDiscoveryModeResponse did not have a Discovery mode parameter of NonDiscoverable after executing Test Procedure 7.

The DUT sent PROBE MATCH message within the time out period of DISCOVERY_TIMEOUT.

The DUT did not send SystemRebootResponse message.

The DUT sent BYE or HELLO message after executing Test Procedure 12.

Note: See Annex A.7 for the value of DISCOVERY_TIMEOUT.

5.9 SOAP FAULT MESSAGE

Test Label: Device Discovery generates SOAP 1.2 fault message for Invalid Unicast PROBE Message.

Test Case ID: DISCOVERY-1-1-10

ONVIF Core Specification Coverage: SOAP Fault Messages

Command under Test: Probe

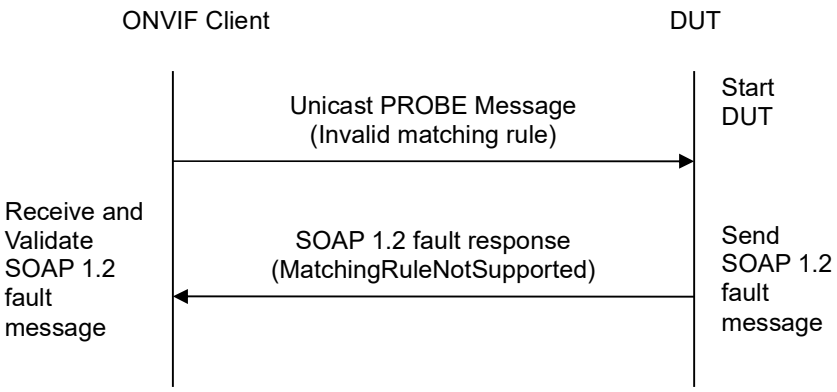
WSDL Reference: ws-discovery.wsdl, devicemgmt.wsdl

Test Purpose: To verify that the DUT generates a SOAP 1.2 fault message to the invalid Unicast PROBE message (Invalid matching rule).

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will transmit Unicast PROBE message with invalid matching type rule.
4. Verify that the DUT generates a SOAP 1.2 fault message (MatchingRuleNotSupported).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SOAP 1.2 fault message.

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

Note: See Annex A.4 for Invalid SOAP 1.2 fault message definition. Refer RFC 3986 for scope matching definitions.

5.10 DEVICE SCOPES CONFIGURATION

Test Label: Device Discovery Device Scope configurations.

Test Case ID: DISCOVERY-1-1-11

ONVIF Core Specification Coverage: Hello, Probe and Probe Match, Get scope parameters, Set scope parameters, Add scope parameters, Remove scope parameters

Command Under Test: AddScopes, SetScopes, RemoveScopes

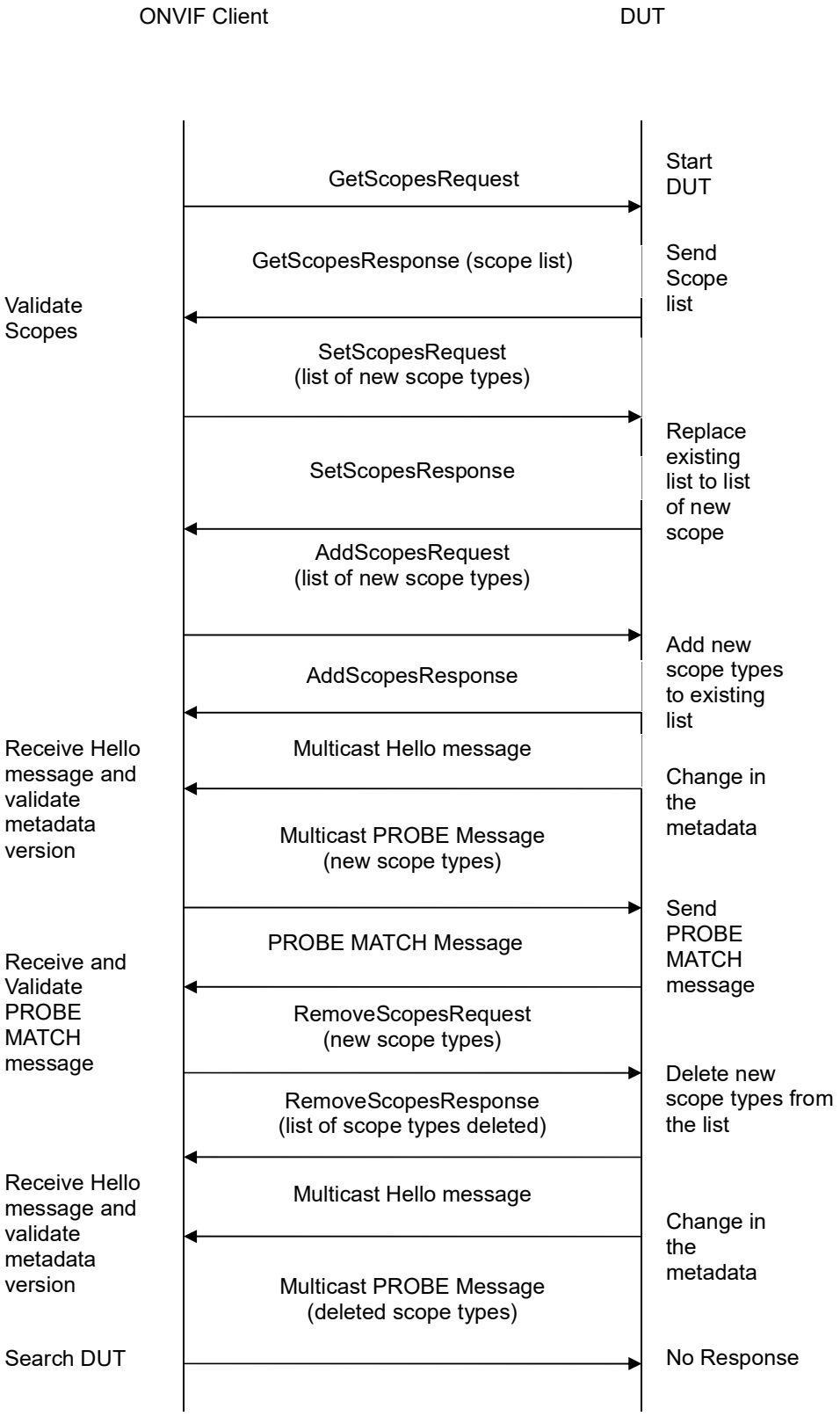
WSDL Reference: ws-discovery.wsdl, devicemgmt.wsdl

Test Purpose: To verify DUT behavior for scope parameter configuration.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
GetScopesRequest
Start DUT
GetScopesResponse (scope list)
Send Scope list
Validate Scopes
SetScopesRequest (list of new scope types)
SetScopesResponse
Replace existing list to list of new scope
AddScopesRequest (list of new scope types)
AddScopesResponse
Add new scope types to existing list
Receive Hello message and validate metadata version
Multicast Hello message
Change in the metadata
Multicast PROBE Message (new scope types)
PROBE MATCH Message
Send PROBE MATCH message
Receive and Validate PROBE MATCH message
RemoveScopesRequest (new scope types)
RemoveScopesResponse (list of scope types deleted)
Delete new scope types from the list
Receive Hello message and validate metadata version
Multicast Hello message
Change in the metadata
Multicast PROBE Message (deleted scope types)
Search DUT
No Response

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetScopesRequest** message to retrieve existing scope types.
4. The DUT replies with the list of scopes types in the **GetScopesResponse** message.
5. ONVIF Client will invoke **SetScopesRequest** message to replace existing list with a list of new scope.
6. The DUT replies with **SetScopesResponse** message indicating success.
7. ONVIF Client waits for 2 seconds.
8. ONVIF Client will invoke **AddScopesRequest** message to add new scope types to the existing scope list.
9. The DUT replies with **AddScopesResponse** message indicating success.
10. The DUT sends Multicast Hello message to indicate the change in the metadata (i.e. addition of new scope types to the existing list).
11. ONVIF Client will invoke Multicast PROBE message to search the DUT with newly added scope types.
12. Verify that the DUT issued a PROBE MATCH message.
13. ONVIF Client will invoke **RemoveScopesRequest** message to delete the newly configured scope types.
14. The DUT replies with **RemoveScopesResponse** message indicating success.
15. The DUT sends Multicast Hello message to indicate the change in the metadata (i.e. deletion of scope types from the existing list).
16. ONVIF Client will invoke Multicast PROBE message to search the DUT with deleted scope types.
17. Verify that the DUT did not send PROBE MATCH message.

Test Result:

PASS –

The DUT passed all assertions

FAIL –

The DUT did not send **GetScopesResponse** message.

The DUT scope list does not have one or more mandatory scope entry.

The DUT did not send **SetScopesResponse** message after executing Test Procedure 5.

The DUT did not send **AddScopesResponse** message.

The DUT did not send multicast Hello message after the change in its metadata (addition/deletion of scope types) with a complete list of current scopes.

The DUT did not send mandatory XML elements (device, new scope type, service address and scope matching rule) in the PROBE MATCH message.

The DUT did not send **RemoveScopesResponse** message.

The DUT did not send PROBE MATCH message within the time out period of DISCOVERY_TIMEOUT after executing Test Procedure steps 10 and 15.

Note:

The DUT may return SOAP Fault 1.2 “TooManyScopes” for the SetScopes (step 5) or AddScopes (step 8) command. Such SOAP 1.2 fault message shall be treated as PASS case for this test.

Whenever there is a change in the metadata of the Target Service, “MetadataVersion” is incremented by >=1.

See Annex A.4 for Invalid SOAP 1.2 fault message definition and Annex A.7 for the value of DISCOVERY_TIMEOUT.

5.11 Namespace Handling

5.11.1 DISCOVERY - NAMESPACES (DEFAULT NAMESPACES FOR EACH TAG)

Test Label: Device Discovery Different Namespaces Definition Test (Default Namespaces for Each Tag).

Test Case ID: DISCOVERY-2-1-1

ONVIF Core Specification Coverage: None

Command Under Test: None

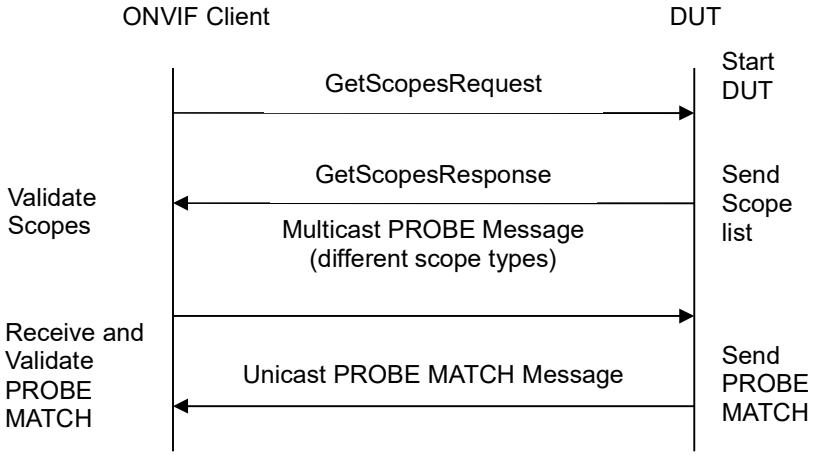
WSDL Reference: ws-discovery.wsdl, devicemgmt.wsdl

Test Purpose: To verify that DUT accepts requests for Device Discovery with different namespaces definition.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT
3. ONVIF Client will invoke GetScopesRequest message to retrieve existing scopes list.
4. DUT replies with the list of scopes types in GetScopesResponse message.
5. ONVIF Client will transmit the multicast PROBE message with different scope types (type, location, hardware and name).
6. ONVIF Client will verify the PROBE MATCH message sent by DUT.

Test Result:

PASS –

The DUT passed all assertions

FAIL –

The DUT did not send GetScopesResponse message.

The DUT scope list does not have one or more mandatory scope entry.

The DUT did not send mandatory XML elements (device type, scope list, service address and scope matching rule) in the PROBE MATCH message.

The DUT did not send PROBE MATCH message within the timeout period of DISCOVERY_TIMEOUT.

The DUT did not send PROBE MATCH message with a namespace of the Types value.

The DUT did not send PROBE MATCH message with fixed entry point to <d:XAddrs> element (“http://<onvif_host>/onvif/device_service”).

In case of IPv6, the DUT did not send PROBE MATCH message with <d:XAddrs> of including IPv6 address.

The DUT did not send PROBE MATCH message with a correct XML element RelatesTo that has the same value as MessageID of PROBE message (not to omit "urn" namespace).

Note: See Annex A.1 for Device and Scope Types definition, Annex A.7 for the value of DISCOVERY_TIMEOUT.

Note: All requests to the DUT shall have default namespaces definition in each tag (see examples in Annex A.11).

5.11.2 DISCOVERY - NAMESPACES (DEFAULT NAMESPACES FOR PARENT TAG)

Test Label: Device Discovery Different Namespaces Definition Test (Default Namespaces for Parent Tag).

Test Case ID: DISCOVERY-2-1-2

ONVIF Core Specification Coverage: None

Command Under Test: None

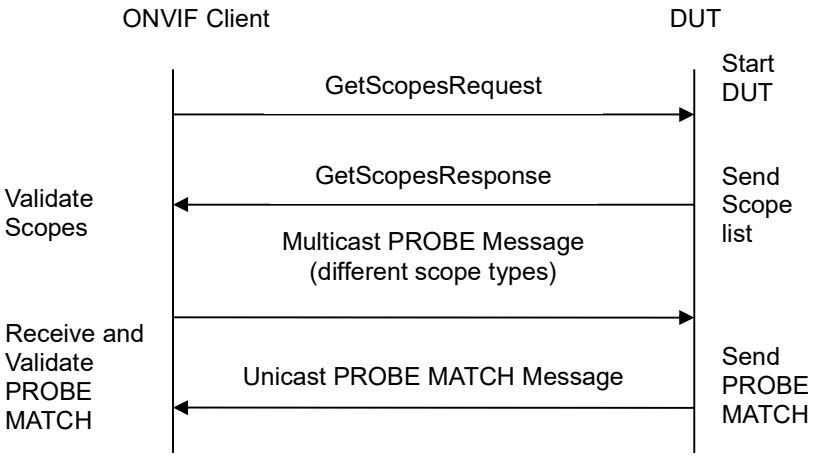
WSDL Reference: ws-discovery.wsdl, devicemgmt.wsdl

Test Purpose: To verify that the DUT accepts requests for Device Discovery with different namespaces definition.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT
3. ONVIF Client will invoke GetScopesRequest message to retrieve existing scopes list.
4. ONVIF Client replies with the list of scopes types in GetScopesResponse message.
5. ONVIF Client will transmit the multicast PROBE message with different scope types (type, location, hardware and name).
6. ONVIF Client will verify the PROBE MATCH message sent by DUT.

Test Result:

PASS –

The DUT passed all assertions

FAIL –

The DUT did not send GetScopesResponse message.

The DUT scope list does not have one or more mandatory scope entry.

The DUT did not send mandatory XML elements (device type, scope list, service address and scope matching rule) in the PROBE MATCH message.

The DUT did not send PROBE MATCH message within the time out period of DISCOVERY_TIMEOUT.

The DUT did not send PROBE MATCH message with a namespace of the Types value.

The DUT did not send PROBE MATCH message with fixed entry point to <d:XAddrs> element (“http://<onvif_host>/onvif/device_service”).

In case of IPv6, the DUT did not send PROBE MATCH message with <d:XAddrs> of including IPv6 address.

The DUT did not send PROBE MATCH message with a correct XML element RelatesTo that has the same value as MessageID of PROBE message (not to omit "urn" namespace).

Note: See Annex A.1 for Device and Scope Types definition, Annex A.7 for the value of DISCOVERY_TIMEOUT.

Note: All requests to the DUT shall have default namespaces definition in parent tag (see examples in Annex A.11).

5.11.3 DISCOVERY - NAMESPACES (NOT STANDARD PREFIXES)

Test Label: Device Discovery Different Namespaces Definition Test (Not Standard Prefixes).

Test Case ID: DISCOVERY-2-1-3

ONVIF Core Specification Coverage: None

Command Under Test: None

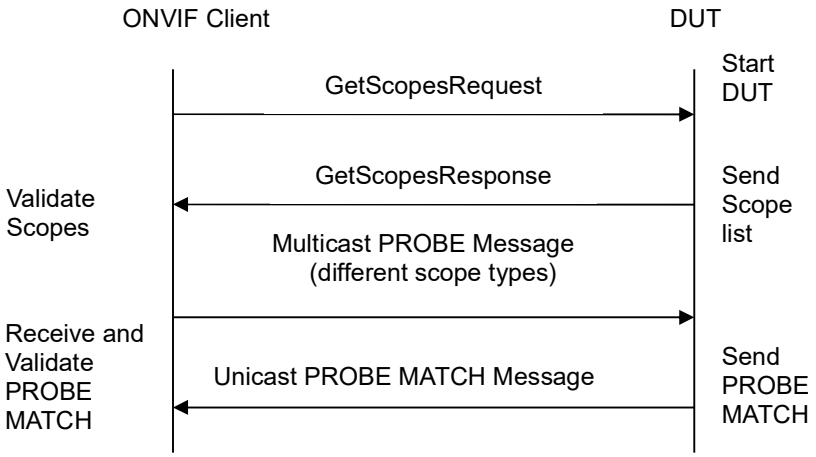
WSDL Reference: ws-discovery.wsdl, devicemgmt.wsdl

Test Purpose: To verify that the DUT accepts requests for Device Discovery with different namespaces definition.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT
3. ONVIF Client will invoke GetScopesRequest message to retrieve existing scopes list.
4. DUT replies with the list of scopes types in GetScopesResponse message.

5. ONVIF Client will transmit the multicast PROBE message with different scope types (type, location, hardware and name).
6. ONVIF Client will verify the PROBE MATCH message sent by DUT.

Test Result:

PASS –

The DUT passed all assertions

FAIL –

The DUT did not send GetScopesResponse message.

The DUT scope list does not have one or more mandatory scope entry.

The DUT did not send mandatory XML elements (device type, scope list, service address and scope matching rule) in the PROBE MATCH message.

The DUT did not send PROBE MATCH message within the time out period of DISCOVERY_TIMEOUT.

The DUT did not send PROBE MATCH message with a namespace of the Types value.

The DUT did not send PROBE MATCH message with fixed entry point to <d:XAddrs> element (“http://<onvif_host>/onvif/device_service”).

In case of IPv6, the DUT did not send PROBE MATCH message with <d:XAddrs> of including IPv6 address.

The DUT did not send PROBE MATCH message with a correct XML element RelatesTo that it has same value as MessageID of PROBE message (not to omit "urn" namespace).

Note: See Annex A.1 for Device and Scope Types definition, Annex A.7 for the value of DISCOVERY_TIMEOUT.

Note: All requests to the DUT shall have namespaces definition with non-standard prefixes (see examples in Annex A.11).

5.11.4 DISCOVERY - NAMESPACES (DIFFERENT PREFIXES FOR THE SAME NAMESPACE)

Test Label: Device Discovery Different Namespaces Definition Test (Different Prefixes for the Same Namespace).

Test Case ID: DISCOVERY-2-1-4

ONVIF Core Specification Coverage: None

Command Under Test: None

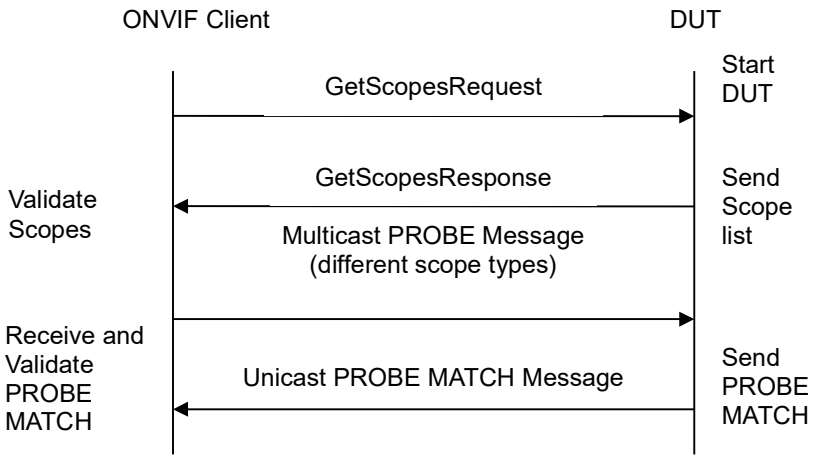
WSDL Reference: ws-discovery.wsdl, devicemgmt.wsdl

Test Purpose: To verify that the DUT accepts requests for Device Discovery with different namespaces definition.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT
3. ONVIF Client will invoke GetScopesRequest message to retrieve existing scopes list.
4. DUT replies with the list of scopes types in GetScopesResponse message.
5. ONVIF Client will transmit the multicast PROBE message with different scope types (type, location, hardware and name).
6. ONVIF Client will verify the PROBE MATCH message sent by DUT.

Test Result:

PASS –

The DUT passed all assertions

FAIL –

The DUT did not send GetScopesResponse message.

The DUT scope list does not have one or more mandatory scope entry.

The DUT did not send mandatory XML elements (device type, scope list, service address and scope matching rule) in the PROBE MATCH message.

The DUT did not send PROBE MATCH message within the time out period of DISCOVERY_TIMEOUT.

The DUT did not send PROBE MATCH message with a namespace of the Types value.

The DUT did not send PROBE MATCH message with fixed entry point to <d:XAddrs> element (“http://<onvif_host>/onvif/device_service”).

In case of IPv6, the DUT did not send PROBE MATCH message with <d:XAddrs> of including IPv6 address.

The DUT did not send PROBE MATCH message with a correct XML element RelatesTo that has the same value as MessageID of PROBE message (not to omit "urn" namespace).

Note: See Annex A.1 for Device and Scope Types definition, Annex A.7 for the value of DISCOVERY_TIMEOUT.

Note: All requests to the DUT shall have namespaces definition with different prefixes for the same namespace (see examples in Annex A.11).

5.11.5 DISCOVERY - NAMESPACES (THE SAME PREFIX FOR DIFFERENT NAMESPACES)

Test Label: Device Discovery Different Namespaces Definition Test (the Same Prefix for Different Namespaces).

Test Case ID: DISCOVERY-2-1-5

ONVIF Core Specification Coverage: None

Command Under Test: None

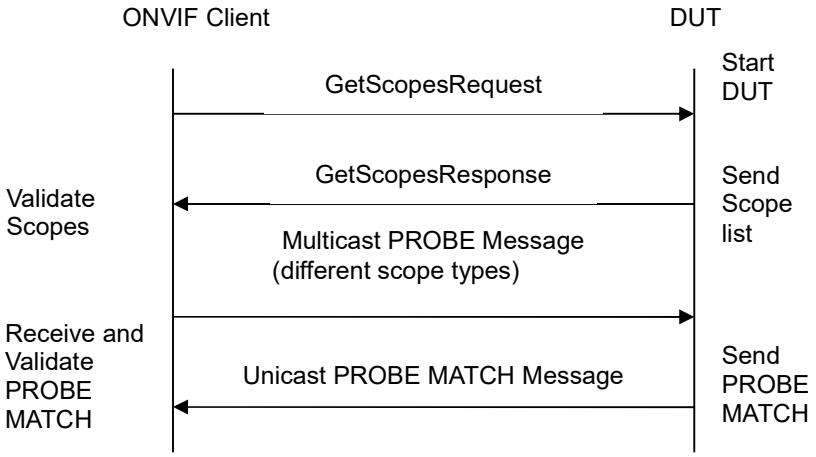
WSDL Reference: ws-discovery.wsdl, devicemgmt.wsdl

Test Purpose: To verify that the DUT accepts requests for Device Discovery with different namespaces definition.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

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

3. ONVIF Client will invoke GetScopesRequest message to retrieve existing scopes list.
4. DUT replies with the list of scopes types in GetScopesResponse message.
5. ONVIF Client will transmit the multicast PROBE message with different scope types (type, location, hardware and name).
6. ONVIF Client will verify the PROBE MATCH message sent by DUT.

Test Result:

PASS –

The DUT passed all assertions

FAIL –

The DUT did not send GetScopesResponse message.

The DUT scope list does not have one or more mandatory scope entry.

The DUT did not send mandatory XML elements (device type, scope list, service address and scope matching rule) in the PROBE MATCH message.

The DUT did not send PROBE MATCH message within the time out period of DISCOVERY_TIMEOUT.

The DUT did not send PROBE MATCH message with a namespace of the Types value.

The DUT did not send PROBE MATCH message with fixed entry point to <d:XAddrs> element (“http://<onvif_host>/onvif/device_service”).

In case of IPv6, the DUT did not send PROBE MATCH message with <d:XAddrs> of including IPv6 address.

The DUT did not send PROBE MATCH message with a correct XML element RelatesTo that it has same value as MessageID of PROBE message (not to omit "urn" namespace).

Note: See Annex A.1 for Device and Scope Types definition, Annex A.7 for the value of DISCOVERY_TIMEOUT.

Note: All requests to the DUT shall have namespaces definition with the same prefixes for different namespaces (see examples in Annex A.11).

6 Device Management Test Cases

6.1 Capabilities

6.1.1 GET WSDL URL

Test Label: Device Management WSDL URL.

Test Case ID: DEVICE-1-1-1

ONVIF Core Specification Coverage: Get WSDL URL

Command under test: GetWsdlUrl

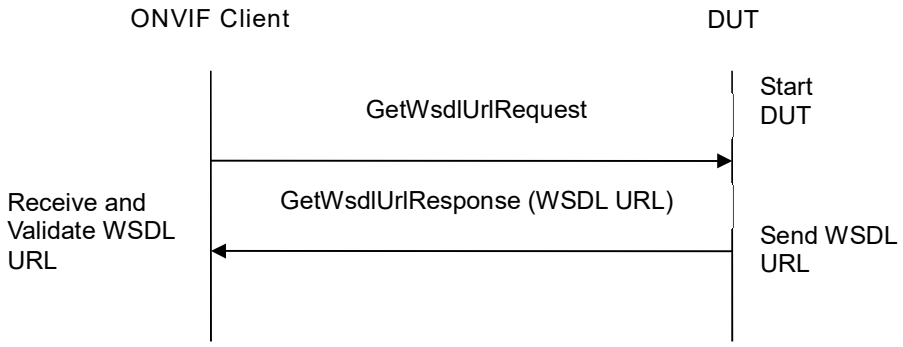
WSDL Reference: devicemgmt.wsdl

Test Purpose: To retrieve complete XML schema and WSDL definitions of the DUT.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetWsdlUrlRequest message to retrieve XML schema and WSDL definitions of the DUT.
4. Verify that DUT sends GetWsdlUrlResponse message (WSDL URL).
5. Validate the WSDL URL returned from the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetWsdlUrlResponse message.

6.1.2 ALL CAPABILITIES

Test Label: Device Management All Capabilities Verification.

Test Case ID: DEVICE-1-1-2

ONVIF Core Specification Coverage: Capability exchange

Command under test: GetCapabilities

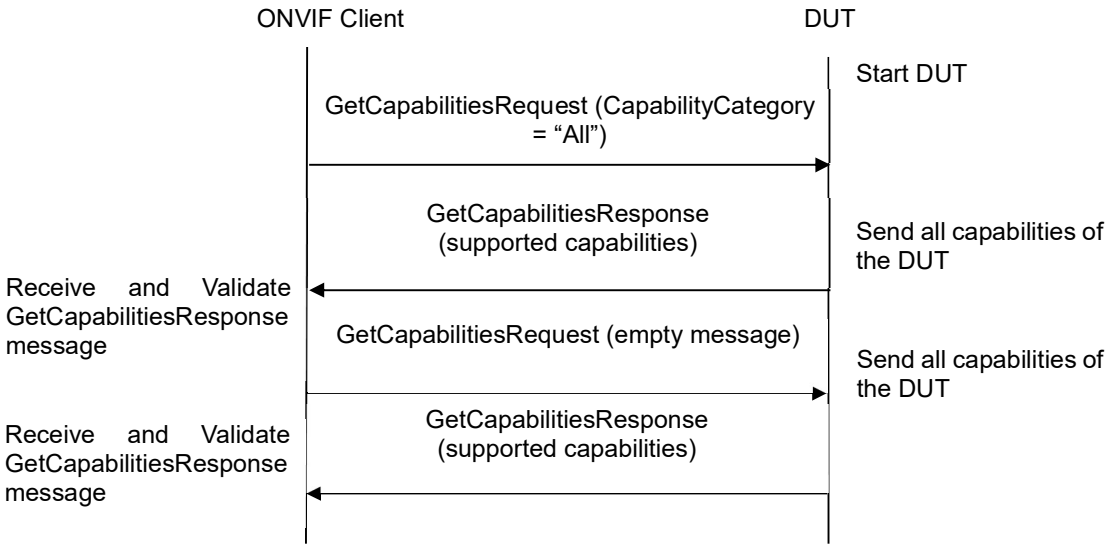
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify all Capabilities of the DUT.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetCapabilitiesRequest message (CapabilityCategory = “All”) to retrieve all capabilities of the DUT.
4. Verify the Capabilities Response from the DUT and support for Device and Events capabilities.

5. Verify support of Media, DeviceIO, PTZ, Imaging, and Analytics if corresponding services are supported.
6. ONVIF Client will invoke GetCapabilitiesRequest message (empty message) to retrieve all capabilities of the DUT.
7. Verify the Capabilities Response from the DUT and support for Device and Events capabilities.
8. Verify support of Media, DeviceIO, PTZ, Imaging, and Analytics if corresponding services are supported.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetCapabilitiesResponse message at step 4 and step 7.

The DUT did not return Device and Events capabilities.

The DUT did not return Media capabilities, if Media service is supported.

The DUT did not return DeviceIO capabilities, if DeviceIO service is supported.

The DUT did not return PTZ capabilities, if PTZ service is supported.

The DUT did not return Analytics capabilities, if Analytics service is supported.

The DUT did not return Imaging capabilities, if Imaging service is supported.

6.1.3 DEVICE CAPABILITIES

Test Label: Device Management Device Capabilities Verification.

Test Case ID: DEVICE-1-1-3

ONVIF Core Specification Coverage: Capability exchange.

Command under test: GetCapabilities

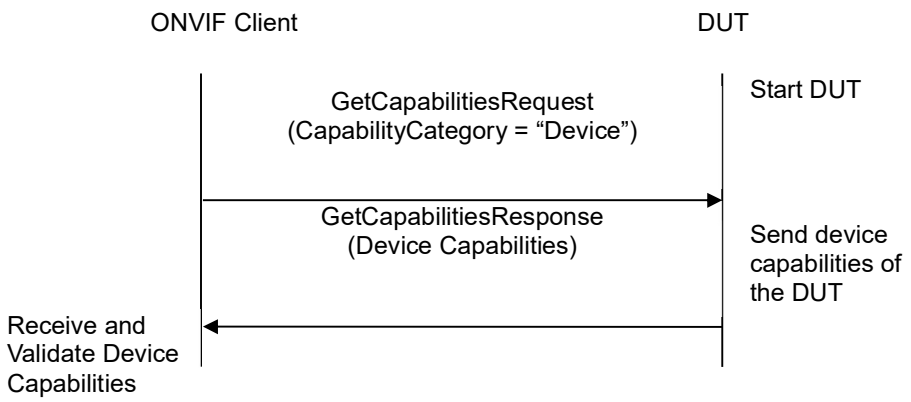
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify Device Capabilities of the DUT.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT.

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetCapabilitiesRequest message (CapabilityCategory = “Device”) to retrieve Device Capabilities of the DUT.
4. DUT sends its device capabilities in the GetCapabilitiesResponse message.
5. Verify the address of the device service in the GetCapabilitiesResponse message.
6. Verify Network, System, IO and Security capabilities if supported by the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetCapabilitiesResponse message.

The DUT did not send the address of the device service.

6.1.4 MEDIA CAPABILITIES

Test Label: Device Management Media Capabilities Verification.

Test Case ID: DEVICE-1-1-4

ONVIF Core Specification Coverage: Capability exchange

Command under test: GetCapabilities

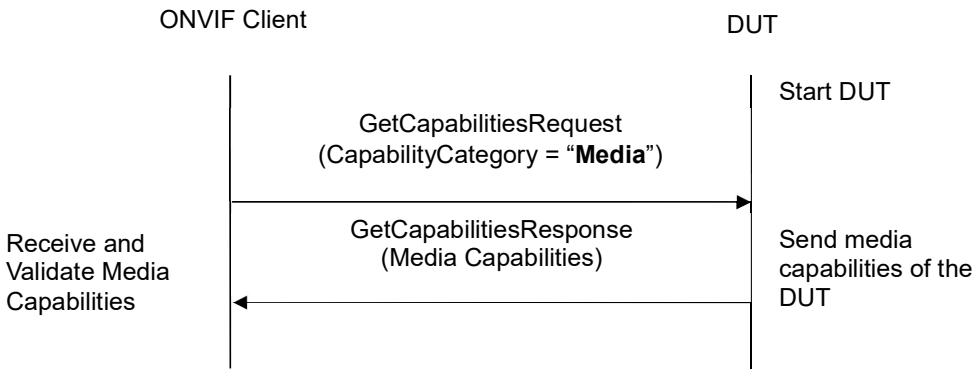
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify Media Capabilities of the DUT.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT.

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetCapabilitiesRequest message (CapabilityCategory = "Media") to retrieve Media Capabilities of the DUT.
4. Verify the GetCapabilitiesResponse message; it should either have address of the Media service or SOAP 1.2 fault message (ActionNotSupported/NoSuchService), if the Media Service is not supported.
5. Verify the address of the media service in the GetCapabilitiesResponse message.
6. Verify Real time streaming capabilities if supported by the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetCapabilitiesResponse message or DUT did not generate the SOAP 1.2 fault message (ActionNotSupported/NoSuchService), if the Media Service is not supported.

The DUT did not send the address of the media service, if the Media Service is supported.

6.1.5 EVENT CAPABILITIES

Test Label: Device Management DUT Event Capabilities Verification.

Test Case ID: DEVICE-1-1-5

ONVIF Core Specification Coverage: Capability exchange.

Command Under Test: GetCapabilities.

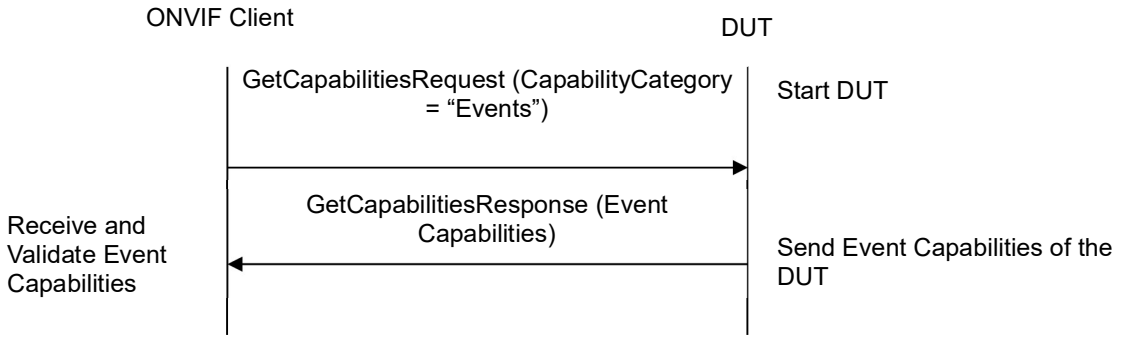
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify event capabilities of the DUT.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT.

Test Sequence:



Test Procedure:

1. Start the DUT.
2. Start an ONVIF Client.
3. ONVIF Client will invoke GetCapabilitiesRequest message (CapabilityCategory = “Events”) to retrieve Event Capabilities of the DUT.
4. Verify the address of the event service in the GetCapabilitiesResponse message.
5. Verify the Subscription policies if supported by the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetCapabilitiesResponse message.

The DUT did not send the address of the event service.

6.1.6 PTZ CAPABILITIES

Test Label: Device Management PTZ Capabilities Verification.

Test Case ID: DEVICE-1-1-6

ONVIF Core Specification Coverage: Capability exchange.

Command Under Test: GetCapabilities.

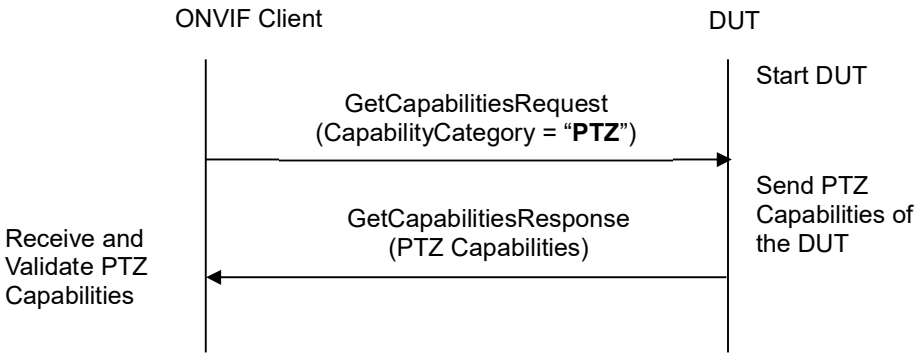
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify PTZ capabilities of the DUT.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT.

Test Sequence:



Test Procedure:

1. Start the DUT.
2. Start an ONVIF Client.
3. ONVIF Client will invoke GetCapabilitiesRequest message (CapabilityCategory = "PTZ") to retrieve PTZ Capabilities of the DUT.
4. Verify the GetCapabilitiesResponse; message should either have address of the PTZ service or SOAP 1.2 fault message (ActionNotSupported/NoSuchService), if the PTZ service is not supported.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetCapabilitiesResponse message.

The DUT did not send the address of the PTZ service, if PTZ supported or DUT did not generate the SOAP 1.2 fault message (ActionNotSupported/NoSuchService), if the PTZ is not supported.

6.1.7 SOAP FAULT MESSAGE

Test Label: Device Management generates a SOAP 1.2 fault message for Invalid GetCapabilitiesRequest Message.

Test Case ID: DEVICE-1-1-9

Command under test: GetCapabilities

ONVIF Core Specification Coverage: Capability exchange.

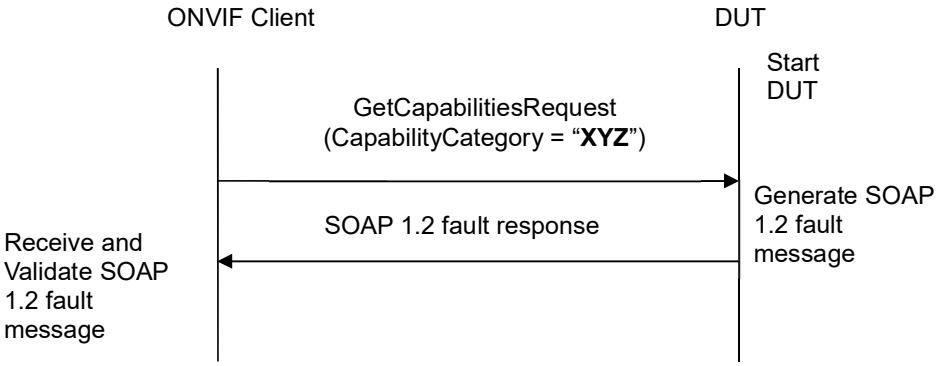
WSDL Reference: devicemgmt.wsdl

Pre-Requisite: None

Test Purpose: To verify that the DUT generates SOAP 1.2 fault message to the invalid GetCapabilitiesRequest message (invalid capability category).

Test Configuration: ONVIF Client and DUT.

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will send GetCapabilitiesRequest message with an invalid capability category.
4. Verify that the DUT generates a SOAP 1.2 fault message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SOAP 1.2 fault message.

The DUT did not send the correct SOAP 1.2 fault message (fault code, namespace etc).

Note: See Annex A.4 for Invalid SOAP 1.2 fault message definition.

6.1.8 IMAGING CAPABILITIES

Test Label: Device Management Imaging Capabilities Verification.

Test Case ID: DEVICE-1-1-10

ONVIF Core Specification Coverage: Capability exchange

Command under test: GetCapabilities

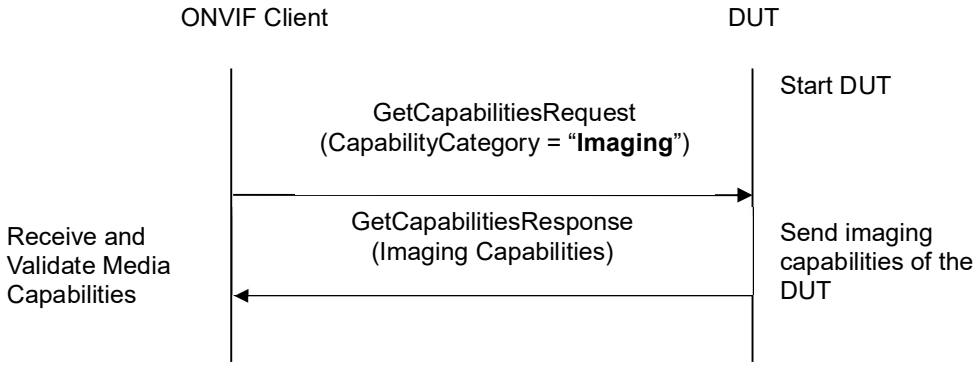
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify Imaging Capabilities of the DUT.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT.

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetCapabilitiesRequest message (CapabilityCategory = "Imaging") to retrieve Imaging Capabilities of the DUT.
4. Verify the GetCapabilitiesResponse message; it should either have address of the Imaging service or SOAP 1.2 fault message (ActionNotSupported/NoSuchService), if the Imaging Service is not supported.
5. Verify the address of the imaging service in the GetCapabilitiesResponse message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetCapabilitiesResponse message or DUT did not generate the SOAP 1.2 fault message (ActionNotSupported/NoSuchService), if the Imaging Service is not supported.

The DUT did not send the address of the imaging service, if the Imaging Service is supported.

6.1.9 ANALYTICS CAPABILITIES

Test Label: Device Management Analytics Capabilities Verification.

Test Case ID: DEVICE-1-1-11

ONVIF Core Specification Coverage: Capability exchange

Command under test: GetCapabilities

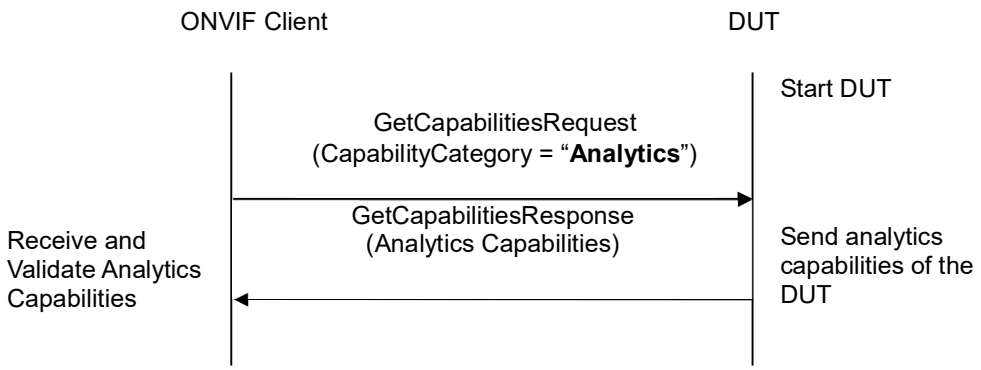
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify Analytics Capabilities of the DUT.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT.

Test Sequence:



Test Procedure:

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

3. ONVIF Client will invoke GetCapabilitiesRequest message (CapabilityCategory = “Analytics”) to retrieve Analytics Capabilities of the DUT.
4. Verify the GetCapabilitiesResponse message; it should either have address of the Analytics service or SOAP 1.2 fault message (ActionNotSupported/NoSuchService), if the Analytics Service is not supported.
5. Verify the address of the analytics service in the GetCapabilitiesResponse message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetCapabilitiesResponse message or DUT did not generate the SOAP 1.2 fault message (ActionNotSupported/NoSuchService), if the Analytics Service is not supported.

The DUT did not send the address of the analytics service, if the Analytics Service is supported.

6.1.10 GET SERVICES – DEVICE SERVICE

Test Label: Device Management Get Services Verification.

Test Case ID: DEVICE-1-1-13

ONVIF Core Specification Coverage: Capability exchange

Command under test: GetServices

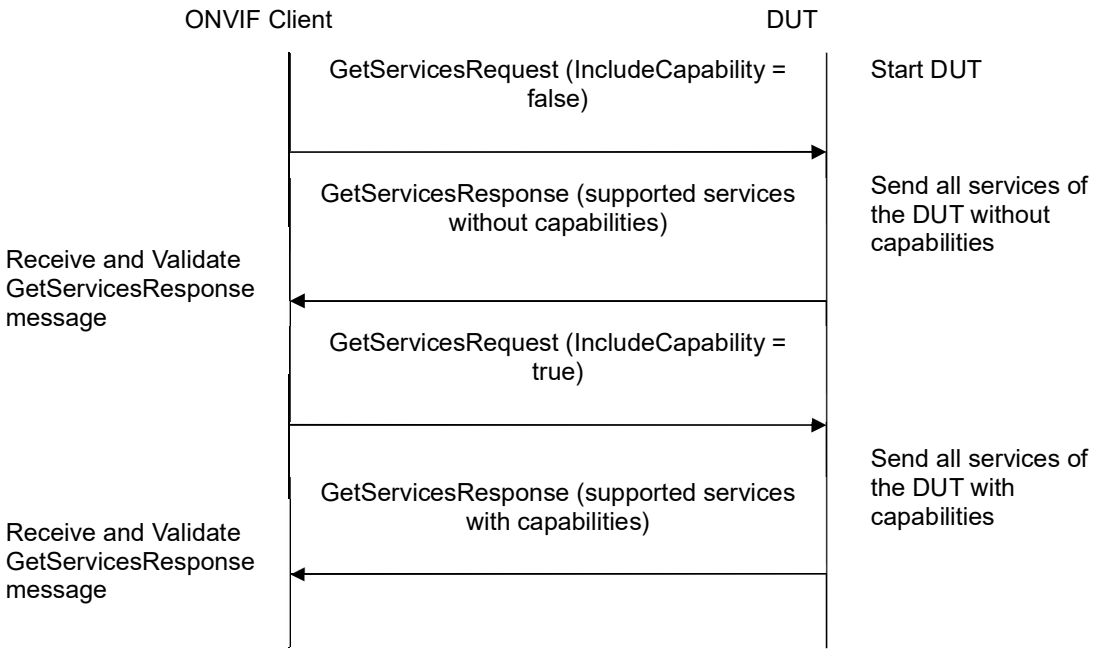
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify that Device Management Service is received using GetServicesRequest.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = false) to retrieve all services of the DUT without service capabilities.
4. Verify the GetServicesResponse message from the DUT.
5. Verify that there are the following services in GetServicesResponse message: Device Management service.
6. Verify that no Capabilities are returned for Device Management service.
7. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = true) to retrieve all services of the DUT with service capabilities.
8. Verify the GetServicesResponse message from the DUT.
9. Verify that there are the following services in GetServicesResponse message: Device Management service.
10. Verify that Capabilities element is returned for Device service and it contains a corresponding element with Device Management service Capabilities. Check that this element is valid according to the corresponding WSDL.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetServicesResponse message at step 4 and step 8.

The DUT did not return Device Management services.

The DUT sent Capabilities at step 4.

The DUT did not send Capabilities at step 8.

The DUT sent Capabilities with wrong element (element does not correspond the service or an element is invalid according to WSDL).

Note: Service will be defined as Device Management service if it has Namespace element that is equal to “http://www.onvif.org/ver10/device/wsdl”.

6.1.11 GET SERVICES – MEDIA SERVICE

Test Label: Device Management Get Services Verification.

Test Case ID: DEVICE-1-1-14

ONVIF Core Specification Coverage: Capability exchange

Command under test: GetServices

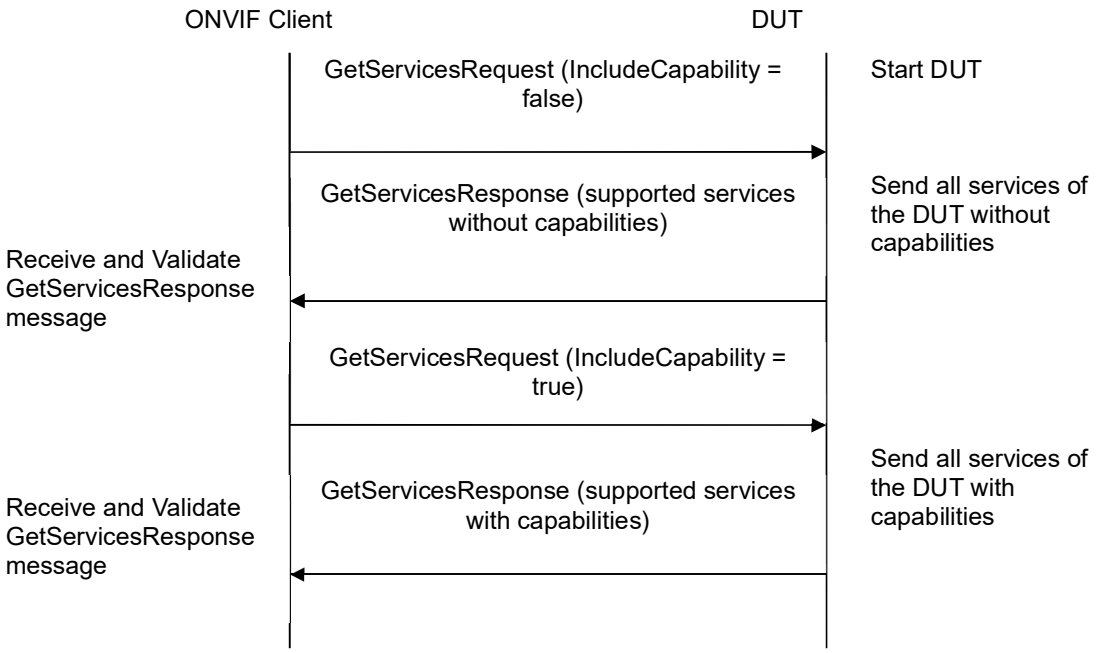
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify that Media Service is received using GetServicesRequest.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = false) to retrieve all services of the DUT without service capabilities.
4. Verify the GetServicesResponse message from the DUT.
5. Verify that there are the following services in GetServicesResponse message: Media service.
6. Verify that no Capabilities are returned for Media service.
7. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = true) to retrieve all services of the DUT with service capabilities.
8. Verify the GetServicesResponse message from the DUT.
9. Verify that there are the following services in GetServicesResponse message: Media service.
10. Verify that Capabilities element is returned for Media service and it contains the corresponding element with Media service Capabilities. Check that this element is valid according to the corresponding WSDL.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetServicesResponse message at step 4 and step 8.

The DUT did not return Media services.

The DUT sent Capabilities at step 4.

The DUT did not send Capabilities at step 8.

The DUT sent Capabilities with wrong element (element does not correspond the service or an element is invalid according to WSDL).

Note: Service will be defined as Media service, if it has Namespace element that is equal to "http://www.onvif.org/ver10/media/wsdl".

6.1.12 GET SERVICES – PTZ SERVICE

Test Label: Device Management Get Services Verification.

Test Case ID: DEVICE-1-1-15

ONVIF Core Specification Coverage: Capability exchange

Command under test: GetServices

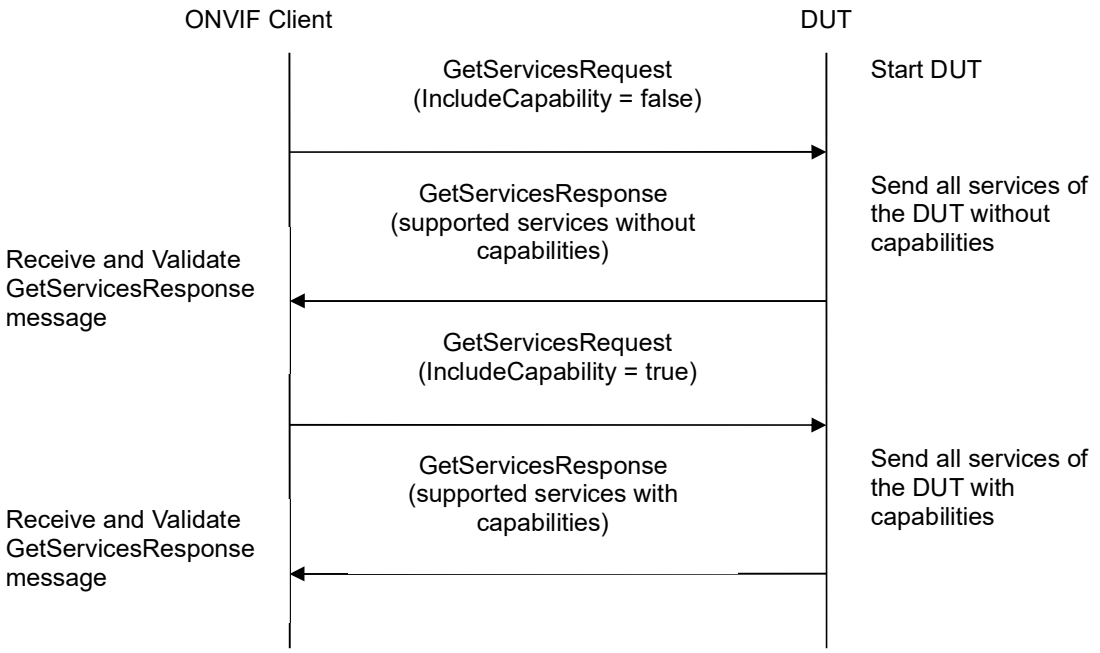
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify that PTZ Service is received using GetServicesRequest.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = false) to retrieve all services of the DUT without service capabilities.
4. Verify the GetServicesResponse message from the DUT.
5. Verify that there are the following services in GetServicesResponse message: PTZ service.
6. Verify that no Capabilities are returned for PTZ service.
7. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = true) to retrieve all services of the DUT with service capabilities.
8. Verify the GetServicesResponse message from the DUT.
9. Verify that there are the following services in GetServicesResponse message: PTZ service.
10. Verify that Capabilities element is returned for PTZ service and it contains the corresponding element with PTZ service Capabilities. Check that this element is valid according to the corresponding WSDL.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetServicesResponse message at step 4 and step 8.

The DUT did not return PTZ services.

The DUT sent Capabilities at step 4.

The DUT did not send Capabilities at step 8.

The DUT sent Capabilities with wrong element (element does not correspond the service or an element is invalid according to WSDL).

Note: Service will be defined as PTZ service if it has Namespace element that is equal to "http://www.onvif.org/ver20/ptz/wsdl".

6.1.13 GET SERVICES – EVENT SERVICE

Test Label: Device Management Get Services Verification.

Test Case ID: DEVICE-1-1-16

ONVIF Core Specification Coverage: Capability exchange

Command under test: GetServices

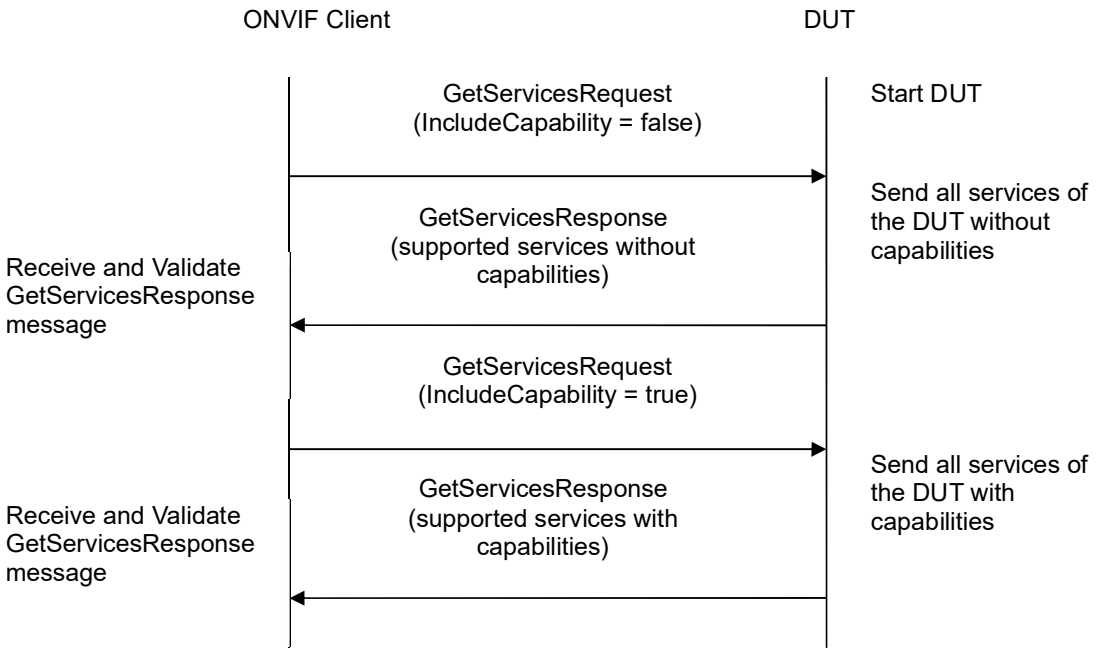
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify that Event Service is received using GetServicesRequest.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = false) to retrieve all services of the DUT without service capabilities.
4. Verify the GetServicesResponse message from the DUT.
5. Verify that there are the following services in GetServicesResponse message: Event service.
6. Verify that no Capabilities are returned for Event service.
7. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = true) to retrieve all services of the DUT with service capabilities.
8. Verify the GetServicesResponse message from the DUT.
9. Verify that there are the following services in GetServicesResponse message: Event service.
10. Verify that Capabilities element is returned for Event service and it contains the corresponding element with Event service Capabilities. Check that this element is valid according to the corresponding WSDL.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetServicesResponse message at step 4 and step 8.

The DUT did not return Event services.

The DUT sent Capabilities at step 4.

The DUT did not send Capabilities at step 8.

The DUT sent Capabilities with wrong element (element does not correspond the service or an element is invalid according to WSDL).

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

6.1.14 GET SERVICES – IMAGING SERVICE

Test Label: Device Management Get Services Verification.

Test Case ID: DEVICE-1-1-17

ONVIF Core Specification Coverage: Capability exchange

Command under test: GetServices

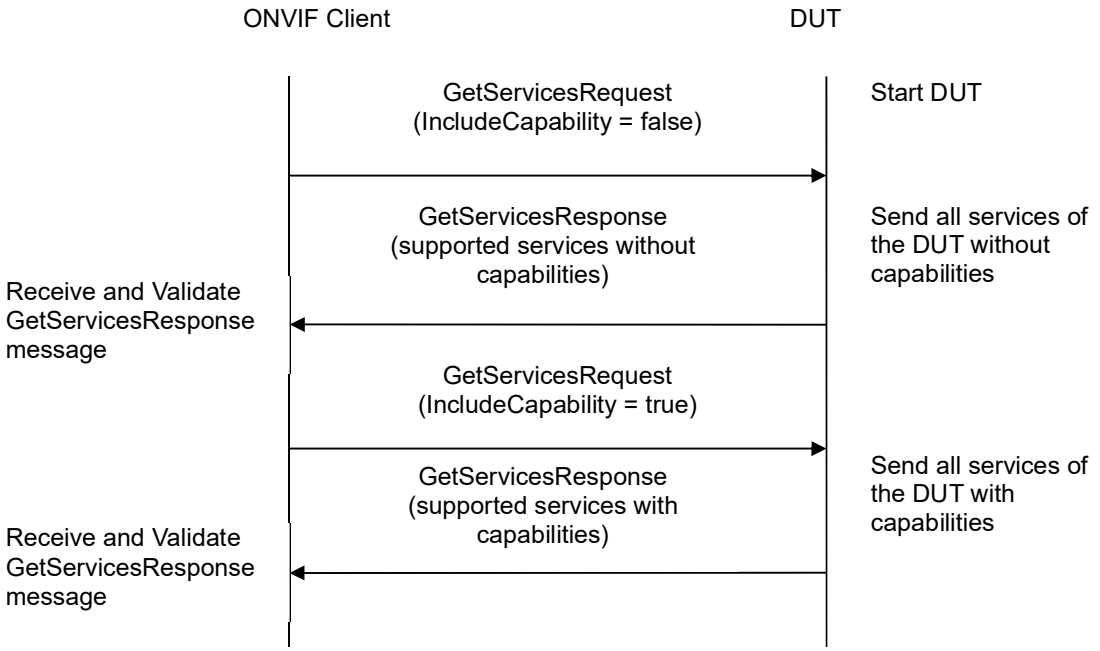
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify that Imaging Service is received using GetServicesRequest.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = false) to retrieve all services of the DUT without service capabilities.
4. Verify the GetServicesResponse message from the DUT.
5. Verify that there are the following services in GetServicesResponse message: Imaging service.
6. Verify that no Capabilities are returned for PTZ service.
7. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = true) to retrieve all services of the DUT with service capabilities.
8. Verify the GetServicesResponse message from the DUT.
9. Verify that there are the following services in GetServicesResponse message: Imaging service.
10. Verify that Capabilities element is returned for PTZ service and it contains the corresponding element with Imaging service Capabilities. Check that this element is valid according to the corresponding WSDL.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetServicesResponse message at step 4 and step 8.

The DUT did not return Imaging services.

The DUT sent Capabilities at step 4.

The DUT did not send Capabilities at step 8.

The DUT sent Capabilities with wrong element (element does not correspond the service or an element is invalid according to WSDL).

Note: Service will be defined as Imaging service if it has Namespace element that is equal to "http://www.onvif.org/ver20/imaging/wsdl".

6.1.15 DEVICE SERVICE CAPABILITIES

Test Label: Device Management Service Capabilities Verification.

Test Case ID: DEVICE-1-1-18

ONVIF Core Specification Coverage: Capability exchange

Command under test: GetServiceCapabilities (for Device Management service)

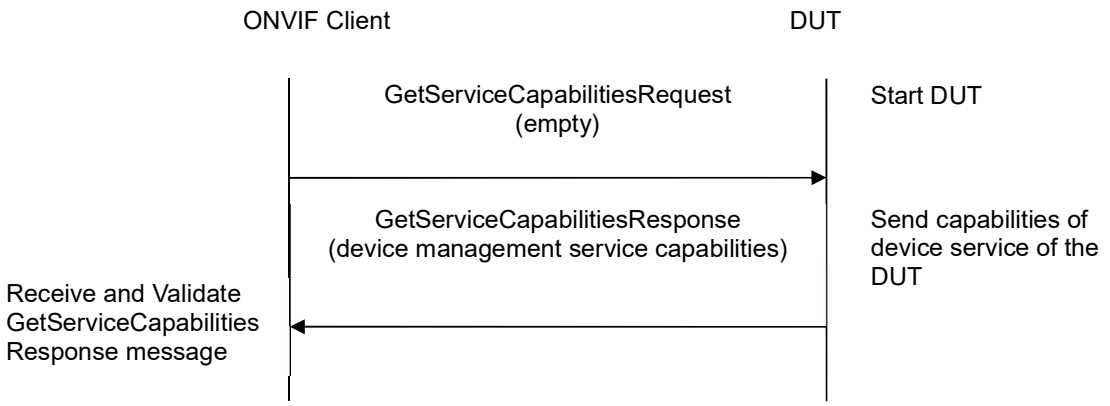
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify Device Management Service Capabilities of the DUT.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetServiceCapabilitiesRequest message to retrieve device management service capabilities of the DUT.
4. Verify the GetServiceCapabilitiesResponse from the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send a valid GetServiceCapabilitiesResponse message.

6.1.16 GET SERVICES AND GET DEVICE SERVICE CAPABILITIES CONSISTENCY

Test Label: Get Services and Device Management Service Capabilities Consistency Verification.

Test Case ID: DEVICE-1-1-19

ONVIF Core Specification Coverage: Capability exchange

Command under test: GetServices, GetServiceCapabilities (for Device Management Service)

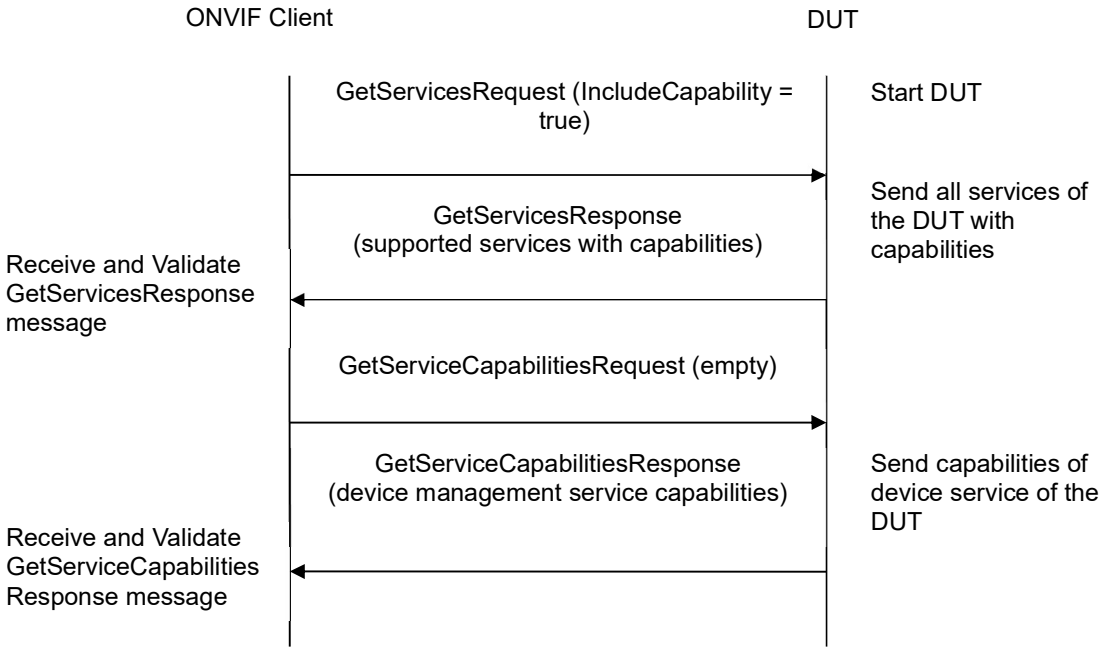
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify Get Services and Device Management Service Capabilities consistency.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetServicesRequest message (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 GetServiceCapabilitiesRequest message to retrieve Device Management service capabilities of the DUT.
6. Verify the GetServiceCapabilitiesResponse message from the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send a valid GetServicesResponse message.

The DUT did not send a valid GetServiceCapabilitiesResponse message.

The DUT sent different Capabilities in GetServicesResponse message and in GetServiceCapabilitiesResponse message.

Note: Service will be defined as Device Management service if it has Namespace element that is equal to “http://www.onvif.org/ver10/device/wsdl”.

6.1.17 GET SERVICES – REPLAY SERVICE

Test Label: Device Management Get Services Verification for Replay Service.

Test Case ID: DEVICE-1-1-20

ONVIF Core Specification Coverage: Capability exchange

Command under test: GetServices

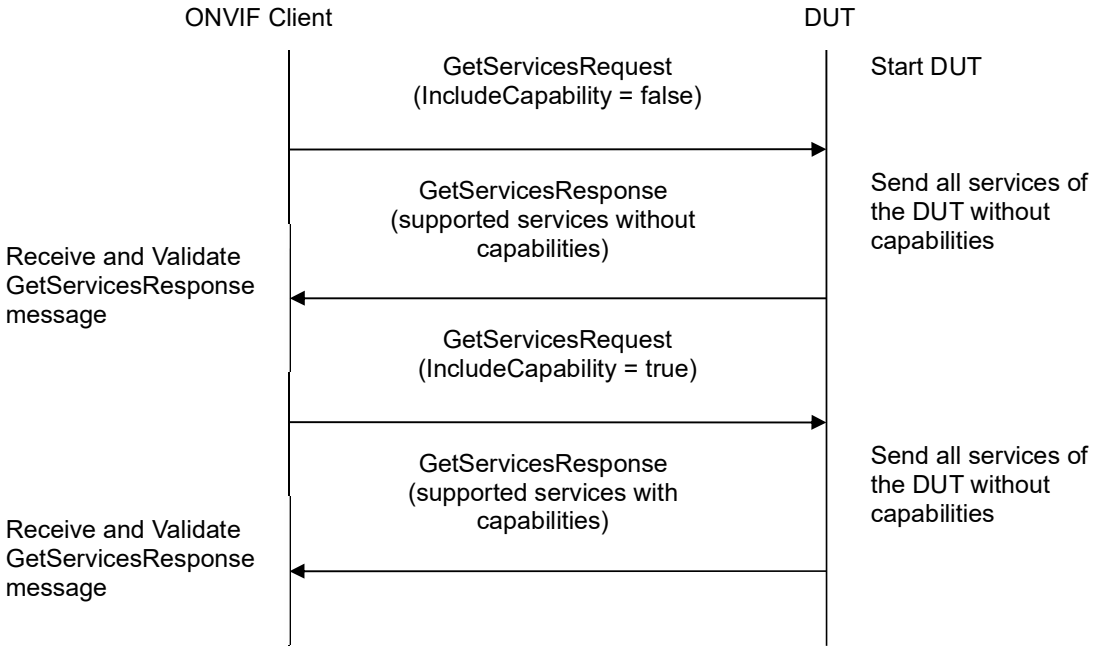
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify getting Replay Service with using GetServicesRequest.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = false) to retrieve all services of the DUT without service capabilities.
4. Verify the GetServicesResponse message from the DUT.
5. Verify that there are the following services in GetServicesResponse message: Replay service.
6. Verify that no Capabilities are returned for Replay service.

7. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = true) to retrieve all services of the DUT with service capabilities.
8. Verify the GetServicesResponse message from the DUT.
9. Verify that there are the following services in GetServicesResponse message: Replay service.
10. Verify that Capabilities element is returned for Replay service and it contains the corresponding element with Replay service Capabilities. Check that this element is valid according to the corresponding WSDL.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetServicesResponse message at step 4 and step 8.

The DUT did not return Replay services.

The DUT sent Capabilities at step 4.

The DUT did not send Capabilities at step 8.

The DUT sent Capabilities with wrong element (element does not correspond the service or an element is invalid according to WSDL).

Note: Service will be defined as Replay service if it has Namespace element that is equal to "http://www.onvif.org/ver10/replay/wsdl".

6.1.18 GET SERVICES – RECORDING SEARCH SERVICE

Test Label: Device Management Get Services Verification for Recording Search Service.

Test Case ID: DEVICE-1-1-21

ONVIF Core Specification Coverage: Capability exchange

Command under test: GetServices

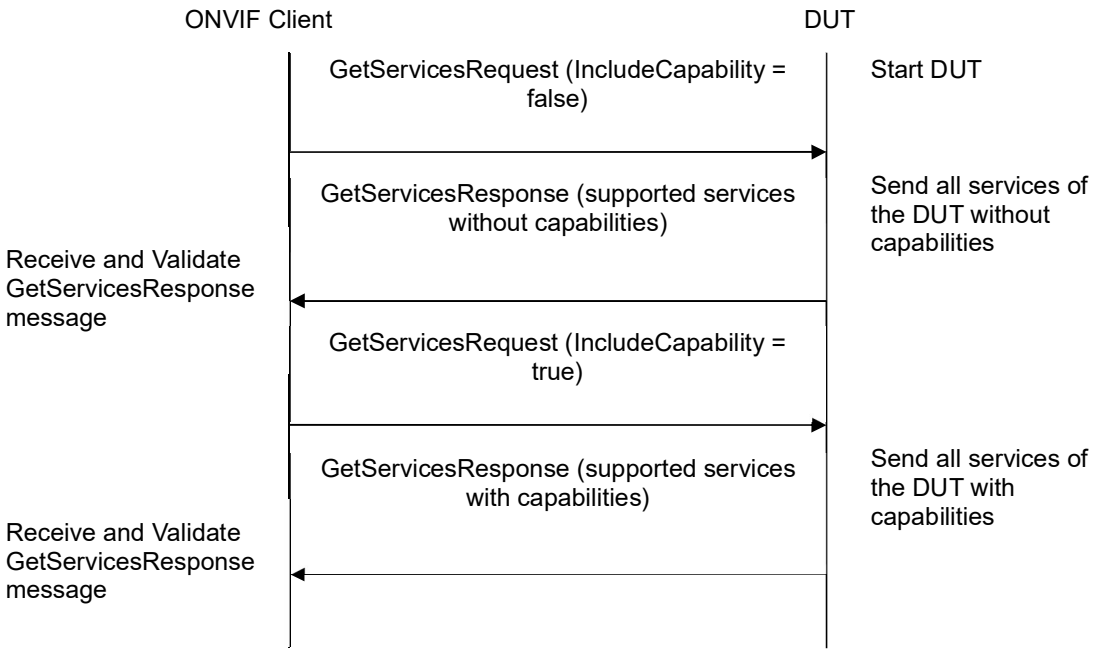
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify that Recording Search Service is received using GetServicesRequest.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = false) to retrieve all services of the DUT without service capabilities.
4. Verify the GetServicesResponse message from the DUT.
5. Verify that there are the following services in GetServicesResponse message: Recording Search service.
6. Verify that no Capabilities are returned for Recording Search service.
7. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = true) to retrieve all services of the DUT with service capabilities.
8. Verify the GetServicesResponse message from the DUT.
9. Verify that there are the following services in GetServicesResponse message: Recording Search service.
10. Verify that Capabilities element is returned for Recording Search service and it contains the corresponding element with Recording Search service Capabilities. Check that this element is valid according to the corresponding WSDL.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetServicesResponse message at step 4 and step 8.

The DUT did not return Recording Search services.

The DUT sent Capabilities at step 4.

The DUT did not send Capabilities at step 8.

The DUT sent Capabilities with wrong element (element does not correspond the service or element is invalid according to the WSDL).

Note: Service will be defined as Recording Search service if it has Namespace element that is equal to “http://www.onvif.org/ver10/search/wsdl”.

6.1.19 GET SERVICES – RECORDING CONTROL SERVICE

Test Label: Device Management Get Services Verification for Recording Control Service.

Test Case ID: DEVICE-1-1-22

ONVIF Core Specification Coverage: Capability exchange (ONVIF Core Specification)

Command under test: GetServices

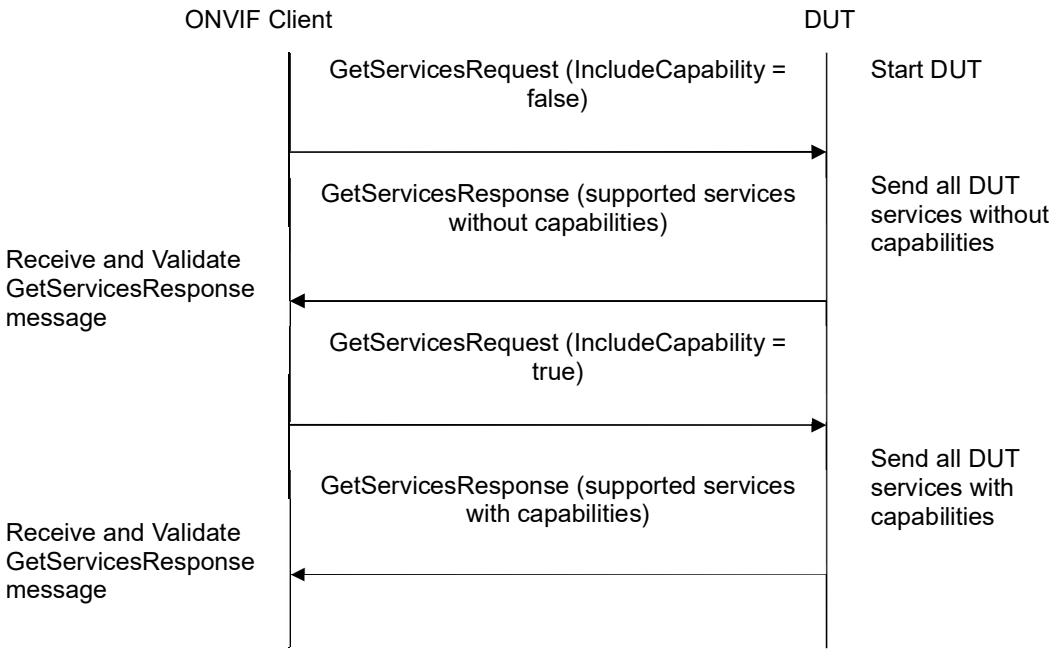
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify that Recording Control Service is received using GetServicesRequest.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = false) to retrieve all DUT services without service capabilities.
4. Verify the GetServicesResponse message from the DUT.
5. Verify that there are the following services in GetServicesResponse message: Recording Control service.
6. Verify that no Capabilities are returned for Recording Control service.
7. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = true) to retrieve all DUT services with service capabilities.
8. Verify the GetServicesResponse message from the DUT.
9. Verify that there are the following services in GetServicesResponse message: Recording Control service.
10. Verify that Capabilities element returned for Recording Control service and it contains a corresponding element with Recording Control service Capabilities. Check that this element is valid according to the corresponding WSDL.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetServicesResponse message at step 4 and step 8.

The DUT did not return Recording Control services.

The DUT sent Capabilities at step 4.

The DUT did not send Capabilities at step 8.

The DUT sent Capabilities with wrong element (element does not correspond the service or an element is invalid according to WSDL).

Note: Service will be defined as Recording Control service if it has Namespace element that is equal to “http://www.onvif.org/ver10/recording/wsdl”.

6.1.20 GET SERVICES – RECEIVER SERVICE

Test Label: Device Management Get Services Verification for Receiver Service.

Test Case ID: DEVICE-1-1-23

ONVIF Core Specification Coverage: Capability exchange (ONVIF Core Specification)

Command under test: GetServices

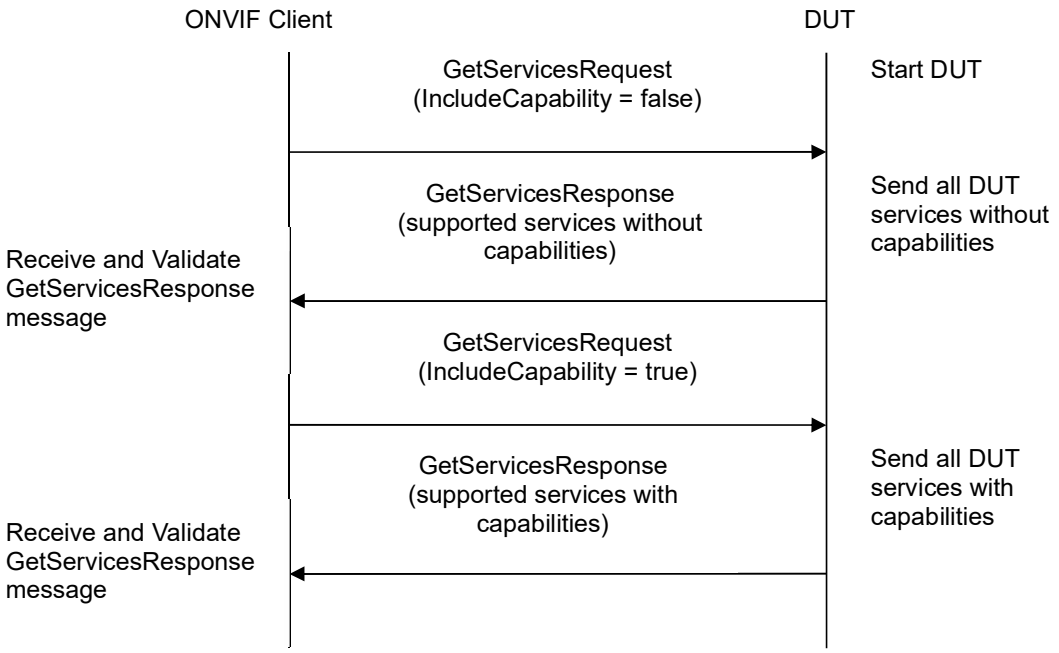
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify that Receiver Service is received using GetServicesRequest.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = false) to retrieve all DUT services without service capabilities.
4. Verify the GetServicesResponse message from the DUT.
5. Verify that there are the following services in GetServicesResponse message: Receiver service.
6. Verify that no Capabilities are returned for Receiver service.
7. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = true) to retrieve all DUT services with service capabilities.
8. Verify the GetServicesResponse message from the DUT.
9. Verify that there are the following services in GetServicesResponse message: Receiver service.
10. Verify that Capabilities element returned for Receiver service and it contains a corresponding element with Receiver service Capabilities. Check that this element is valid according to the corresponding WSDL.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetServicesResponse message at step 4 and step 8.

The DUT did not return Receiver services.

The DUT sent Capabilities at step 4.

The DUT did not send Capabilities at step 8.

The DUT sent Capabilities with wrong element (element does not correspond the service or an element is invalid according to WSDL).

Note: Service will be defined as Receiver service if it has Namespace element that is equal to "http://www.onvif.org/ver10/receiver/wsdl".

6.1.21 GET SERVICES – ACCESS CONTROL SERVICE

Test Label: Device Management Get Services Verification for Access Control Service.

Test Case ID: DEVICE-1-1-24

ONVIF Core Specification Coverage: Capability exchange (ONVIF Core Specification)

Command under test: GetServices

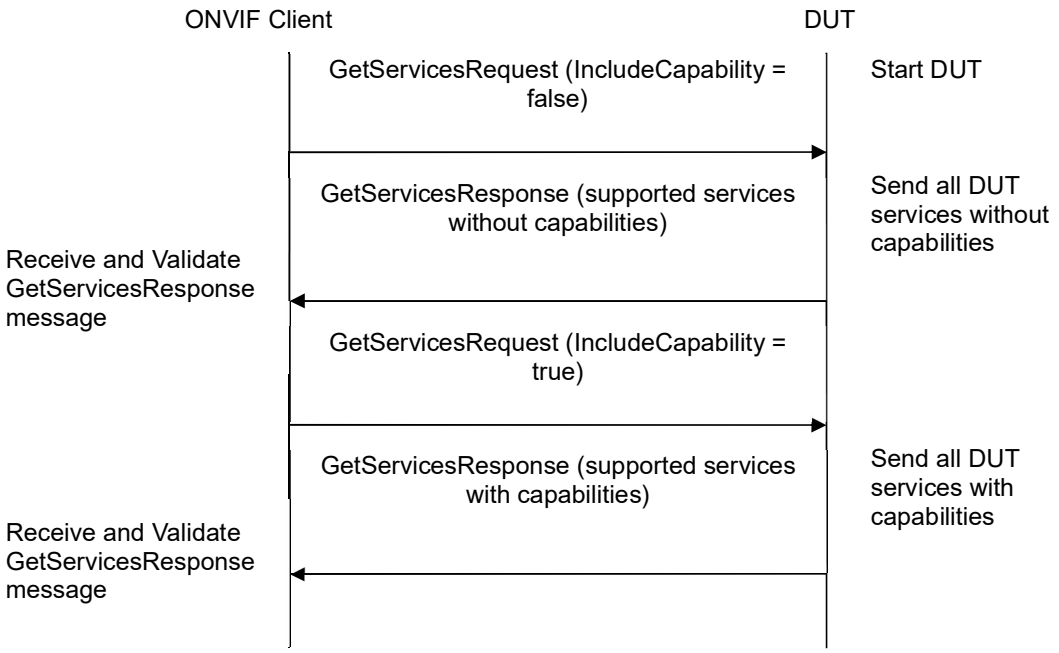
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify that Access Control Service is received using GetServicesRequest.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = false) to retrieve all DUT services without service capabilities.
4. Verify the GetServicesResponse message from the DUT.
5. Verify that there are the following services in GetServicesResponse message: Access Control service.
6. Verify that no Capabilities are returned for Access Control service.
7. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = true) to retrieve all services of the DUT with service capabilities.
8. Verify the GetServicesResponse message from the DUT.
9. Verify that there are the following services in GetServicesResponse message: Access Control service.
10. Verify that Capabilities element returned for Access Control service and it contains a corresponding element with Access Control service Capabilities. Check that this element is valid according to a corresponding WSDL.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetServicesResponse message at step 4 and step 8.

The DUT did not return Access Control services.

The DUT sent Capabilities at step 4.

The DUT did not send Capabilities at step 8.

The DUT sent Capabilities with wrong element (element does not correspond the service or element is invalid according to WSDL).

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

6.1.22 GET SERVICES – DOOR CONTROL SERVICE

Test Label: Device Management Get Services Verification for Door Control Service.

Test Case ID: DEVICE-1-1-25

ONVIF Core Specification Coverage: Capability exchange (ONVIF Core Specification)

Command under test: GetServices

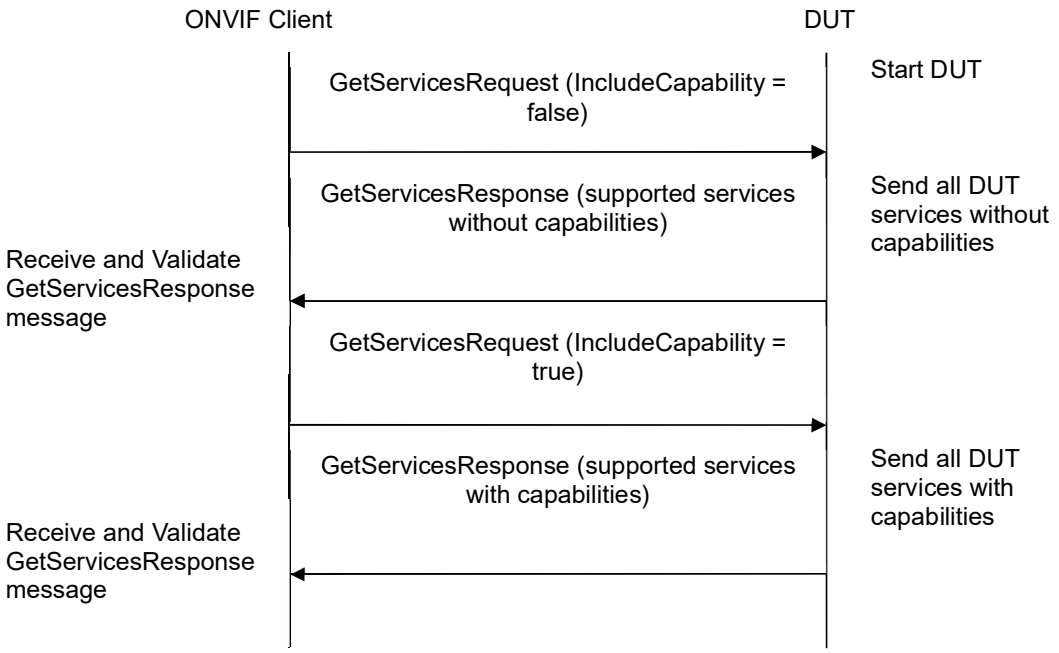
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify getting Door Control Service using GetServicesRequest.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = false) to retrieve all DUT services without service capabilities.
4. Verify the GetServicesResponse message from the DUT.
5. Verify that there are the following services in GetServicesResponse message: Door Control service.
6. Verify that no Capabilities are returned for Door Control service.
7. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = true) to retrieve all DUT services with service capabilities.
8. Verify the GetServicesResponse message from the DUT.
9. Verify that there are the following services in GetServicesResponse message: Door Control service.
10. Verify that Capabilities element returned for Door Control service and it contains a corresponding element with Door Control service Capabilities. Check that this element is valid according to a corresponding WSDL.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetServicesResponse message at step 4 and step 8.

The DUT did not return Door Control services.

The DUT sent Capabilities at step 4.

The DUT did not send Capabilities at step 8.

The DUT sent Capabilities with wrong element (element does not correspond the service or an element is invalid according to WSDL).

Note: Service will be defined as Door Control service if it has Namespace element that is equal to "http://www.onvif.org/ver10/doorcontrol/wsdl".

6.1.23 GET SERVICES – ADVANCED SECURITY SERVICE

Test Label: Device Management Get Services Verification for Advanced Security Service.

Test Case ID: DEVICE-1-1-26

ONVIF Core Specification Coverage: Capability exchange (ONVIF Core Specification)

Command under test: GetServices

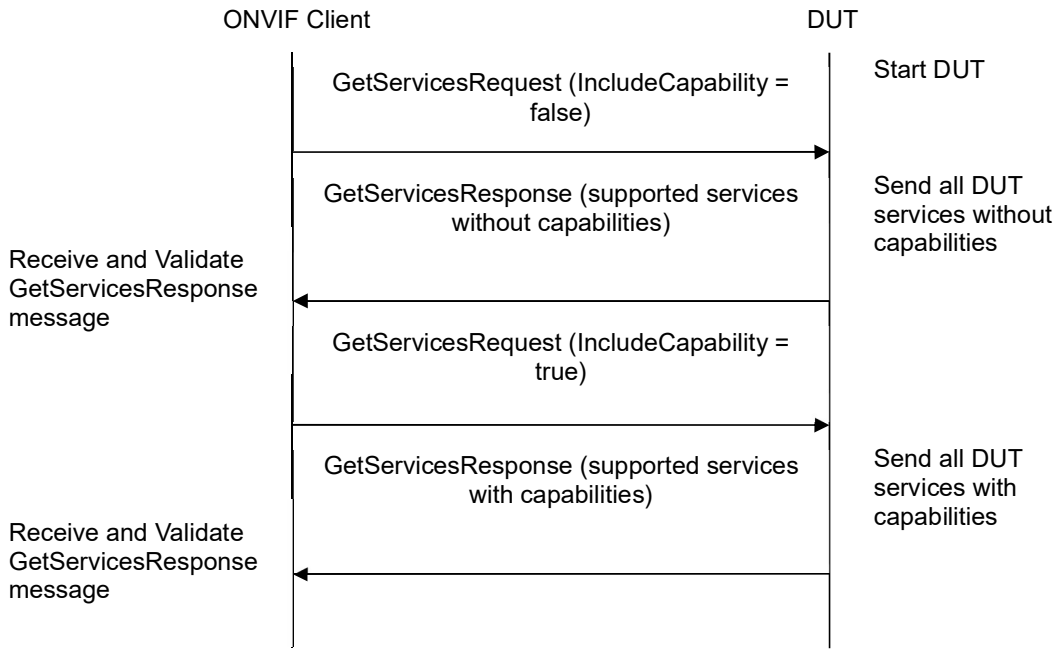
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify getting Advanced Security Service using GetServicesRequest.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = false) to retrieve all DUT services without service capabilities.
4. Verify the GetServicesResponse message from the DUT.
5. Verify that there are the following services in GetServicesResponse message: Advanced Security service.
6. Verify that no Capabilities are returned for Advanced Security service.
7. ONVIF Client will invoke GetServicesRequest message (IncludeCapability = true) to retrieve all DUT services with service capabilities.
8. Verify the GetServicesResponse message from the DUT.
9. Verify that there are the following services in GetServicesResponse message: Advanced Security service.
10. Verify that Capabilities element returned for Advanced Security service and it contains a corresponding element with Advanced Security service Capabilities. Check that this element is valid according to a corresponding WSDL.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetServicesResponse message at step 4 and step 8.

The DUT did not return Advanced Security services.

The DUT sent Capabilities at step 4.

The DUT did not send Capabilities at step 8.

The DUT sent Capabilities with wrong element (element does not correspond the service or an element is invalid according to WSDL).

Note: Service will be defined as Advanced Security service if it has Namespace element that is equal to “http://www.onvif.org/ver10/advancedsecurity/wsdl”.

6.1.24 GET SERVICES – ACCESS RULES SERVICE

Test Label: Device Management Get Services Verification for Access Rules Service.

Test Case ID: DEVICE-1-1-27

ONVIF Core Specification Coverage: Capability exchange (ONVIF Core Specification)

Command under test: GetServices

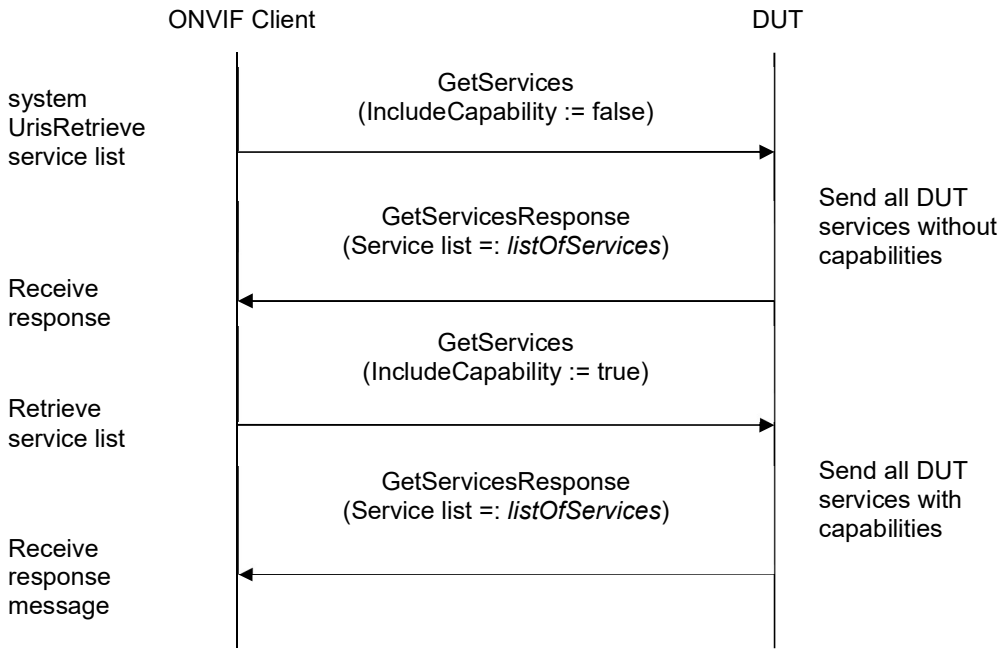
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify getting Access Rules Service using GetServicesRequest.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
    - IncludeCapability := false
4. The DUT responds with a **GetServicesResponse** message with parameters:
    - Service list =: *listOfServices*
5. If *listOfServices* does not contain Access Rules service, FAIL the test and skip other steps.
6. If *listOfServices* contains Service.Capabilities element for Access Rules service, FAIL the test and skip other steps.
7. ONVIF Client invokes **GetServices** message with parameters:
    - IncludeCapability := true
8. The DUT responds with a **GetServicesResponse** message with parameters:
    - Service list =: *listOfServices*
9. If *listOfServices* does not contain Access Rules service, FAIL the test and skip other steps.
10. If *listOfServices* does not contain Service.Capabilities element for Access Rules service, FAIL the test and skip other steps.
11. If Service.Capabilities element for Access Rules service does not valid according to a corresponding WSDL, FAIL the test and skip other steps.

12. If Service.Capabilities element for Access Rules service does not correspond the Credential service, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

Note: Service will be defined as Access Rules service if it has Namespace element that is equal to "http://www.onvif.org/ver10/accessrules/wsdl".

6.1.25 GET SERVICES – CREDENTIAL SERVICE

Test Label: Device Management Get Services Verification for Credential Service.

Test Case ID: DEVICE-1-1-28

ONVIF Core Specification Coverage: Capability exchange (ONVIF Core Specification)

Command under test: GetServices

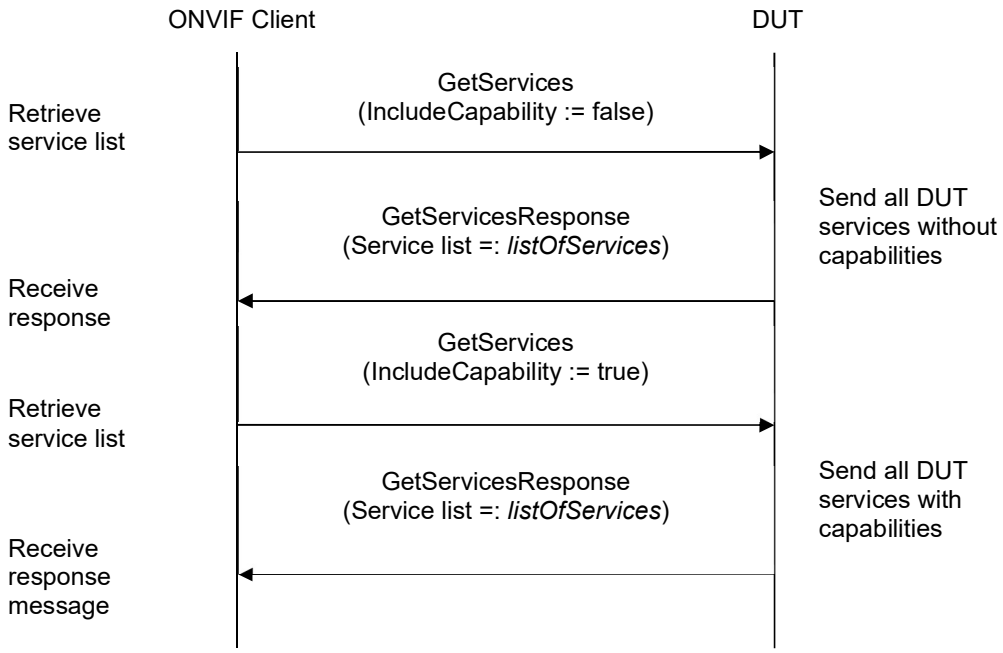
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify getting Credential Service using GetServicesRequest.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
    - IncludeCapability := false
4. The DUT responds with a **GetServicesResponse** message with parameters:
    - Service list =: *listOfServices*
5. If *listOfServices* does not contain Credential service, FAIL the test and skip other steps.
6. If *listOfServices* contains Service.Capabilities element for Credential service, FAIL the test and skip other steps.
7. ONVIF Client invokes **GetServices** message with parameters:
    - IncludeCapability := true
8. The DUT responds with a **GetServicesResponse** message with parameters:
    - Service list =: *listOfServices*
9. If *listOfServices* does not contain Credential service, FAIL the test and skip other steps.
10. If *listOfServices* does not contain Service.Capabilities element for Credential service, FAIL the test and skip other steps.
11. If Service.Capabilities element for Credential service does not valid according to a corresponding WSDL, FAIL the test and skip other steps.

12. If Service.Capabilities element for Credential service does not correspond the Credential service, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

Note: Service will be defined as Credential service if it has Namespace element that is equal to "http://www.onvif.org/ver10/credential/wsdl".

6.1.26 GET SERVICES – SCHEDULE SERVICE

Test Label: Device Management Get Services Verification for Schedule Service.

Test Case ID: DEVICE-1-1-29

ONVIF Core Specification Coverage: Capability exchange (ONVIF Core Specification)

Command under test: GetServices

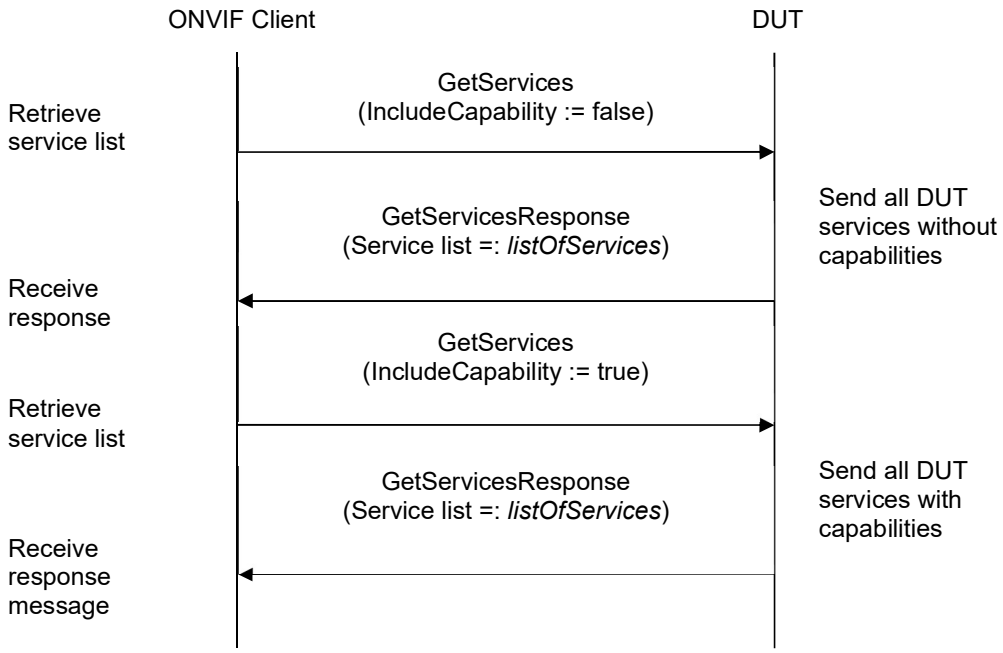
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify getting Schedule Service using GetServicesRequest.

Pre-Requisite: Get Services is supported by the DUT. Schedule Service is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
   - IncludeCapability := false
4. The DUT responds with a **GetServicesResponse** message with parameters:
   - Service list =: *listOfServices*
5. If *listOfServices* does not contain Schedule service, FAIL the test and skip other steps.
6. If *listOfServices* contains Service.Capabilities element for Schedule service, FAIL the test and skip other steps.
7. ONVIF Client invokes **GetServices** message with parameters:
   - IncludeCapability := true
8. The DUT responds with a **GetServicesResponse** message with parameters:
   - Service list =: *listOfServices*
9. If *listOfServices* does not contain Schedule service, FAIL the test and skip other steps.
10. If *listOfServices* does not contain Service.Capabilities element for Schedule service, FAIL the test and skip other steps.
11. If Service.Capabilities element for Schedule service does not valid according to a corresponding WSDL, FAIL the test and skip other steps.

12. If Service.Capabilities element for Schedule service does not correspond the Schedule service, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

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

6.2 Network

6.2.1 NETWORK COMMAND HOSTNAME CONFIGURATION

Test Label: Device Management Network Command GetHostname Test.

Test Case ID: DEVICE-2-1-1

ONVIF Core Specification Coverage: Get hostname

Command Under Test: GetHostname

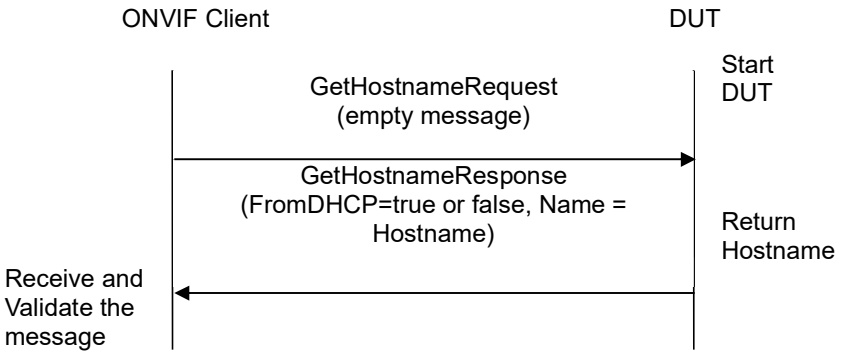
WSDL Reference: devicemgmt.wsdl

Test Purpose: To retrieve hostname of the DUT using GetHostname command.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetHostnameRequest (empty message) message to retrieve Hostname of the DUT.
4. Verify the GetHostnameResponse from the DUT (FromDHCP = true or false, Name = Hostname).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetHostnameResponse message.

6.2.2 NETWORK COMMAND SETHOSTNAME TEST ERROR CASE

Test Label: Device Management Network Command SetHostname Test for invalid hostname.

Test Case ID: DEVICE-2-1-3

ONVIF Core Specification Coverage: Set hostname

Command Under Test: SetHostname.

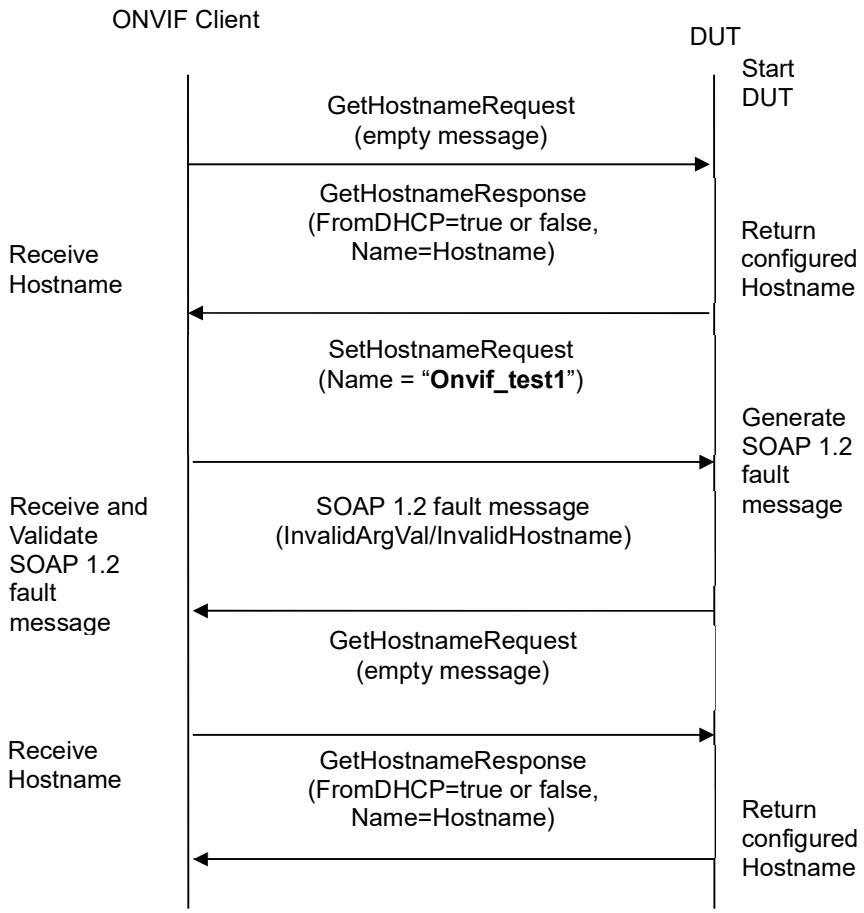
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify behavior of DUT for invalid hostname configuration.

Pre-Requisite: Testing environment (DHCP server) should not change the IP address of DUT during this test case execution.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetHostnameRequest to get the Hostname of the DUT.
4. The DUT returns configured currently Hostname settings.
5. ONVIF Client will invoke SetHostnameRequest message (Name = "Onvif_test1") to configure the hostname.
6. Verify that the DUT generates SOAP 1.2 fault message (InvalidArgVal/InvalidHostname).
7. Verify hostname from DUT through GetHostnameRequest.
8. The DUT sends valid hostname in GetHostnameResponse message (FromDHCP=true or false, Name=Hostname).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SOAP 1.2 fault message.

The DUT did not send a correct fault code in the SOAP 1.2 fault message (InvalidArgVal/InvalidHostname).

The DUT did not send GetHostnameResponse message.

The DUT returned “Onvif_test1” as its Hostname.

Note: Hostname “Onvif_test1” is just an example. See Annex A.2 for Invalid Hostname and SOAP 1.2 fault message definitions.

6.2.3 GET DNS CONFIGURATION

Test Label: Device Management Network Command GetDNS Test.

Test Case ID: DEVICE-2-1-4

ONVIF Core Specification Coverage: Get DNS settings

Command Under Test: GetDNS

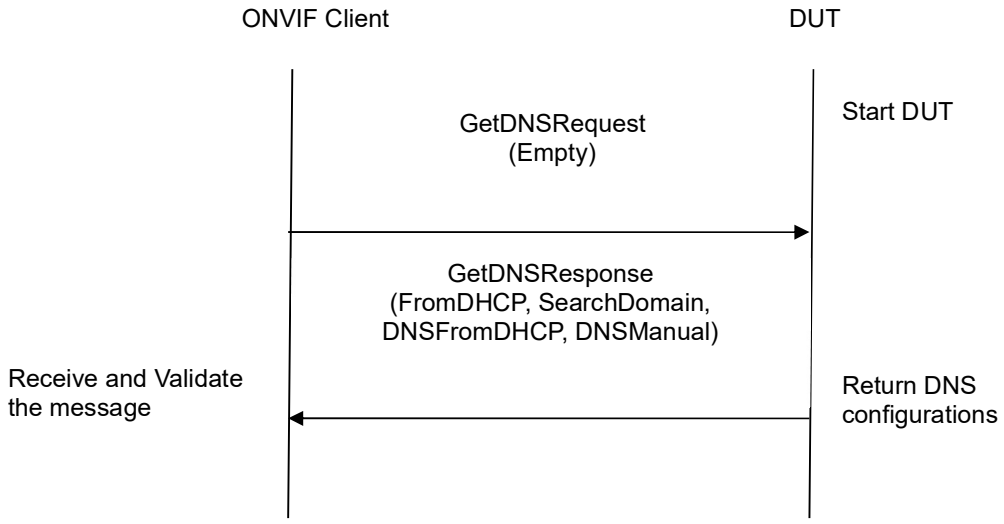
WSDL Reference: devicemgmt.wsdl

Test Purpose: To retrieve DNS configurations of DUT through GetDNS command.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetDNSRequest message to retrieve DNS configurations of the DUT.
4. Verify the GetDNSResponse from the DUT (DNSInformation [FromDHCP = true or false, SearchDomain = domain to search if hostname is not fully qualified, DNSFromDHCP = list of DNS Servers obtained from DHCP, DNSManual = list of DNS Servers manually configured]).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetDNSResponse message.

The DUT did not send correct information (i.e. DNSInformation [FromDHCP = true or false, SearchDomain = domain to search if hostname is not fully qualified, DNSFromDHCP = list of DNS Servers obtained from DHCP, DNSManual = list of DNS Servers manually configured]) in the GetDNSResponse message.

Note: See Annex A.10 for valid expression in terms of an empty IP address.

6.2.4 SET DNS CONFIGURATION - SEARCHDOMAIN

Test Label: Device Management Network Command SetDNS SearchDomain Test.

Test Case ID: DEVICE-2-1-5

ONVIF Core Specification Coverage: Set DNS settings

Command Under Test: SetDNS

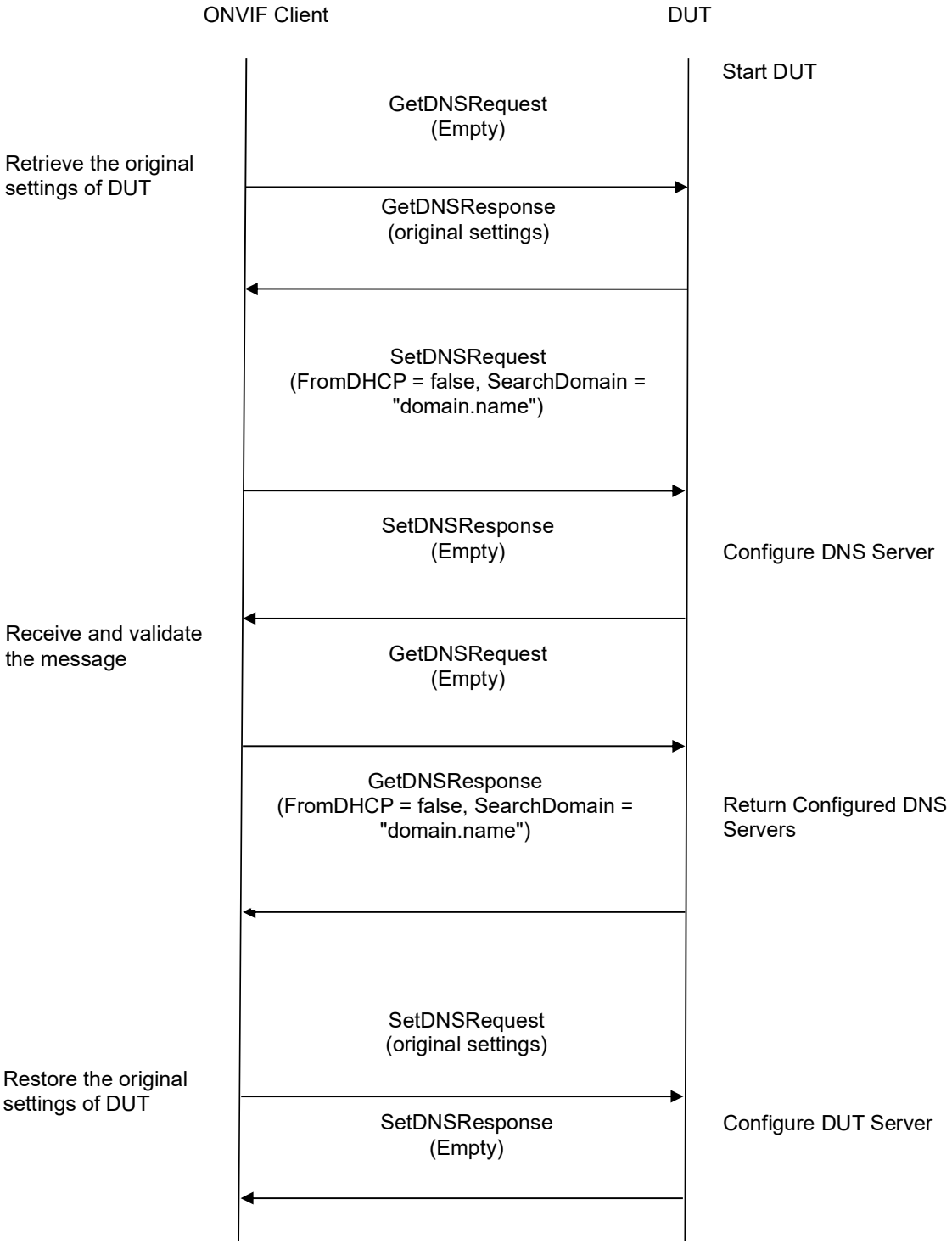
WSDL Reference: devicemgmt.wsdl

Test Purpose: To configure DNS Search Domain setting at the DUT using SetDNS command.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetDNSRequest message to retrieve the original settings of the DUT.
4. ONVIF Client will invoke SetDNSRequest message (FromDHCP = false, SearchDomain = "domain.name").
5. Verify that the DUT sends SetDNSResponse (empty message).
6. Verify the DNS configurations at the DUT through GetDNSRequest.
7. The DUT sends its DNS configurations in the GetDNSResponse message (DNSInformation [FromDHCP = false, SearchDomain = "domain.name"]).
8. ONVIF Client will invoke SetDNSRequest message to restore the original settings of the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetDNSResponse message.

The DUT did not send GetDNSResponse message.

The DUT did not send correct information (i.e. DNSInformation [FromDHCP = false, SearchDomain = "domain.name"]) in the GetDNSResponse message.

6.2.5 SET DNS CONFIGURATION - DNSMANUAL IPV4

Test Label: Device Management Network Command SetDNS Test (DNSManual = IPv4 address).

Test Case ID: DEVICE-2-1-6

ONVIF Core Specification Coverage: Set DNS settings

Command Under Test: SetDNS

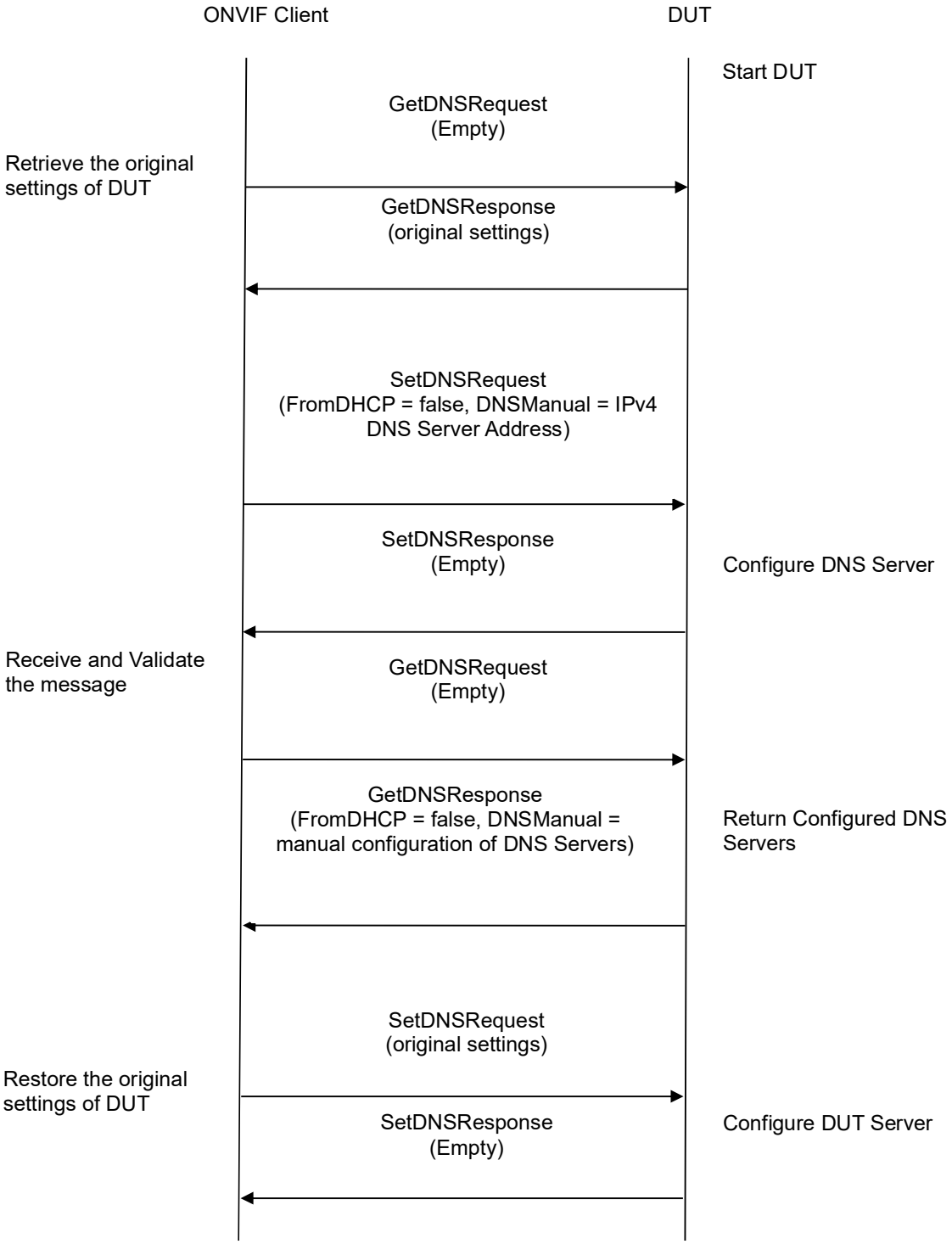
WSDL Reference: devicemgmt.wsdl

Test Purpose: To configure IPv4 DNS server address setting at the DUT using SetDNS command.

Pre-Requisite: DHCP is disabled (see Annex A.13).

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetDNSRequest message to retrieve the original settings of the DUT.
4. ONVIF Client will invoke SetDNSRequest message (FromDHCP = false, DNSManual = "IPv4", "10.1.1.1").
5. Verify that the DUT sends SetDNSResponse (empty message).
6. Verify the DNS configurations in DUT through GetDNSRequest.
7. The DUT sends its DNS configurations in the GetDNSResponse message (DNSInformation [FromDHCP = false, DNSManual = "IPv4", "10.1.1.1"]).
8. ONVIF Client will invoke SetDNSRequest message to restore the original settings of DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetDNSResponse message.

The DUT did not send GetDNSResponse message.

The DUT did not send correct information (i.e. DNSInformation [FromDHCP = false, DNSManual = "IPv4", "10.1.1.1"]) in the GetDNSResponse message.

Note: See Annex A.6 for Valid IPv4 Address definition. See Annex A.10 for valid expression in terms of empty IP address.

6.2.6 SET DNS CONFIGURATION - DNSMANUAL IPV6

Test Label: Device Management Network Command SetDNS Test (DNSManual = IPv6 address).

Test Case ID: DEVICE-2-1-7

ONVIF Core Specification Coverage: Set DNS settings

Command Under Test: SetDNS

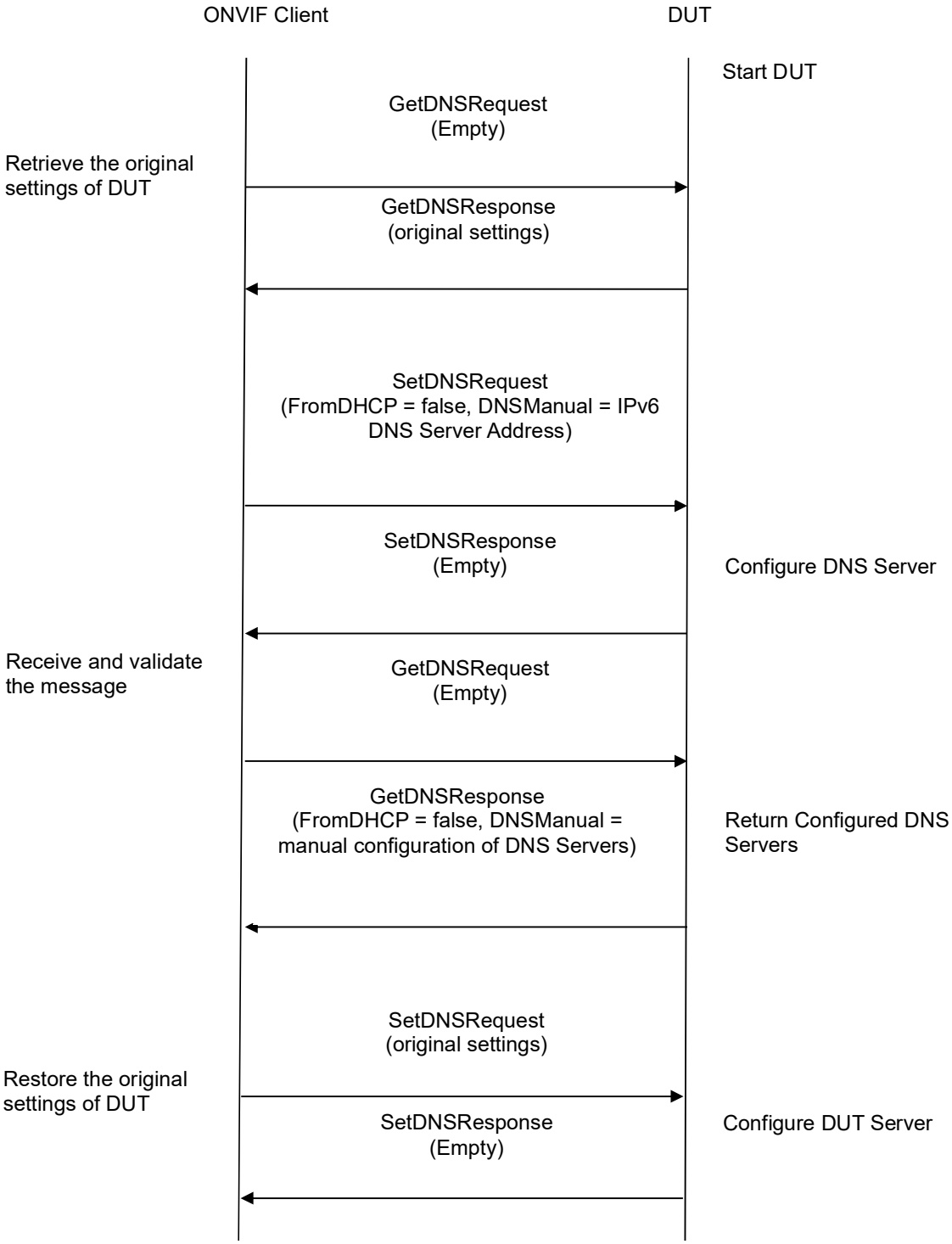
WSDL Reference: devicemgmt.wsdl

Test Purpose: To configure IPv6 DNS server address setting at the DUT using SetDNS command.

Pre-Requisite: DHCP is disabled (see Annex A.15). IPv6 is implemented by the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetDNSRequest message to retrieve the original settings of the DUT.
4. ONVIF Client will invoke SetDNSRequest message (FromDHCP = false, DNSManual = "IPv6", "2001:1:1:1:1:1:1:1").
5. Verify that the DUT sends SetDNSResponse (empty message).
6. Verify the DNS configurations at the DUT through GetDNSRequest.
7. The DUT sends its DNS configurations in the GetDNSResponse message (DNSInformation [FromDHCP = false, DNSManual = "IPv6", "2001:1:1:1:1:1:1:1"]).
8. ONVIF Client will invoke SetDNSRequest message to restore the original settings of DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetDNSResponse message.

The DUT did not send GetDNSResponse message.

The DUT did not send correct information (i.e. DNSInformation [FromDHCP = false, DNSManual = "IPv6", "2001:1:1:1:1:1:1:1"]) in the GetDNSResponse message.

6.2.7 SET DNS CONFIGURATION - FROMDHCP

Test Label: Device Management Network Command SetDNS FromDHCP Test.

Test Case ID: DEVICE-2-1-8

ONVIF Core Specification Coverage: Set DNS settings

Command Under Test: SetDNS

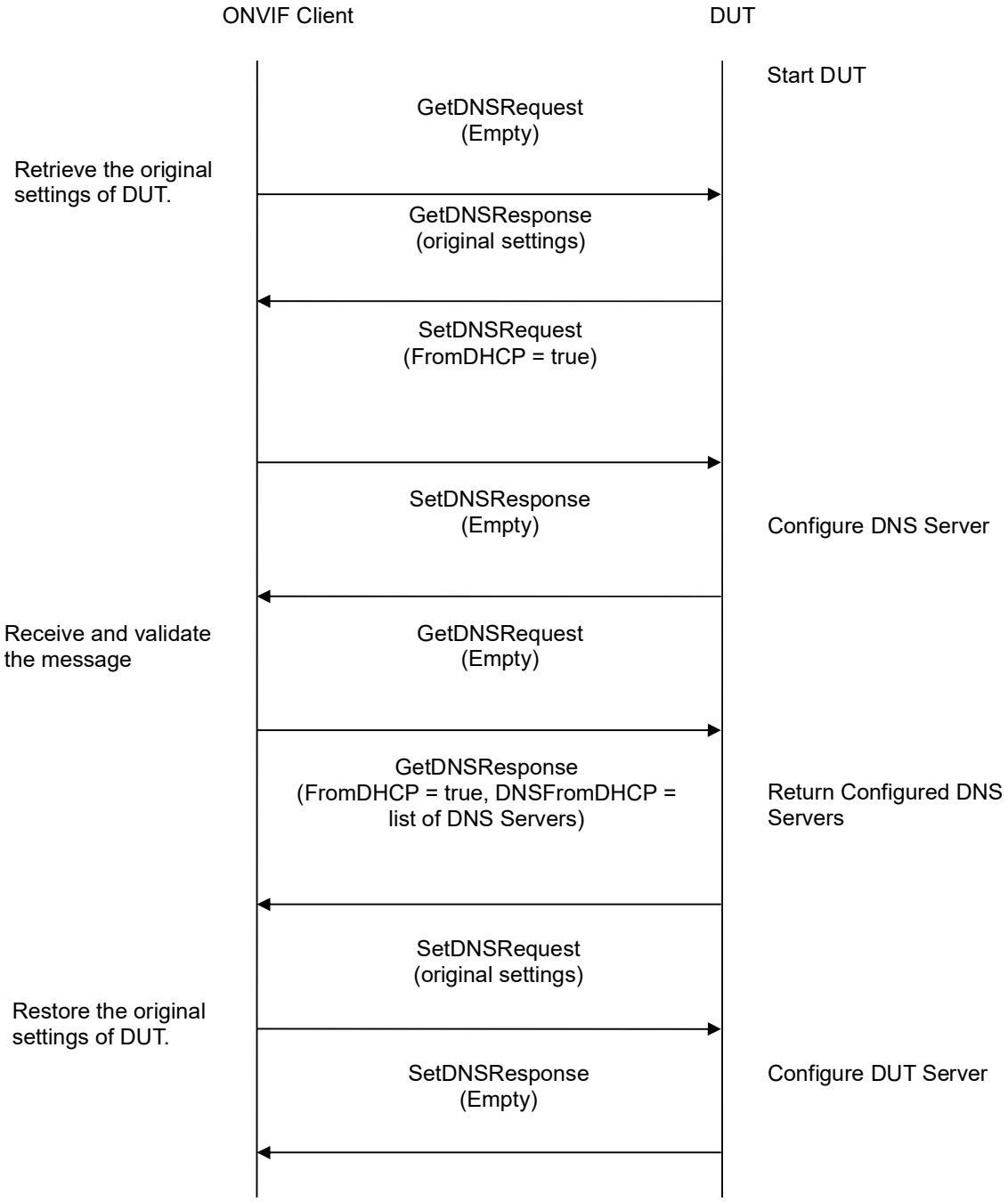
WSDL Reference: devicemgmt.wsdl

Test Purpose: To configure DNS FromDHCP setting in DUT through SetDNS command.

Pre-Requisite: DHCP is enabled (see Annex A.12 and A.14).

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

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

3. ONVIF Client will invoke GetDNSRequest message to retrieve the original settings of the DUT.
4. ONVIF Client will invoke SetDNSRequest message (FromDHCP = true).
5. Verify that the DUT sends SetDNSResponse (empty message)
6. Verify the DNS configurations in the DUT through GetDNSRequest.
7. The DUT sends its DNS configurations in the GetDNSResponse message (DNSInformation [FromDHCP = true, DNSFromDHCP = list of DNS Servers obtained from DHCP]).
8. ONVIF Client will invoke SetDNSRequest message to restore the original settings of DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetDNSResponse message.

The DUT did not send GetDNSResponse message.

The DUT did not send correct information (i.e. DNSInformation [FromDHCP = true, DNSFromDHCP = list of DNS Servers obtained from DHCP, the list can be empty if the DHCP server does not deliver DNS Addresses]) in the GetDNSResponse message.

6.2.8 SET DNS CONFIGURATION - DNSMANUAL INVALID IPV4

Test Label: Device Management Network Command SetDNS Test (DNSManual = invalid IPv4 address).

Test Case ID: DEVICE-2-1-9

ONVIF Core Specification Coverage: Set DNS settings

Command Under Test: SetDNS

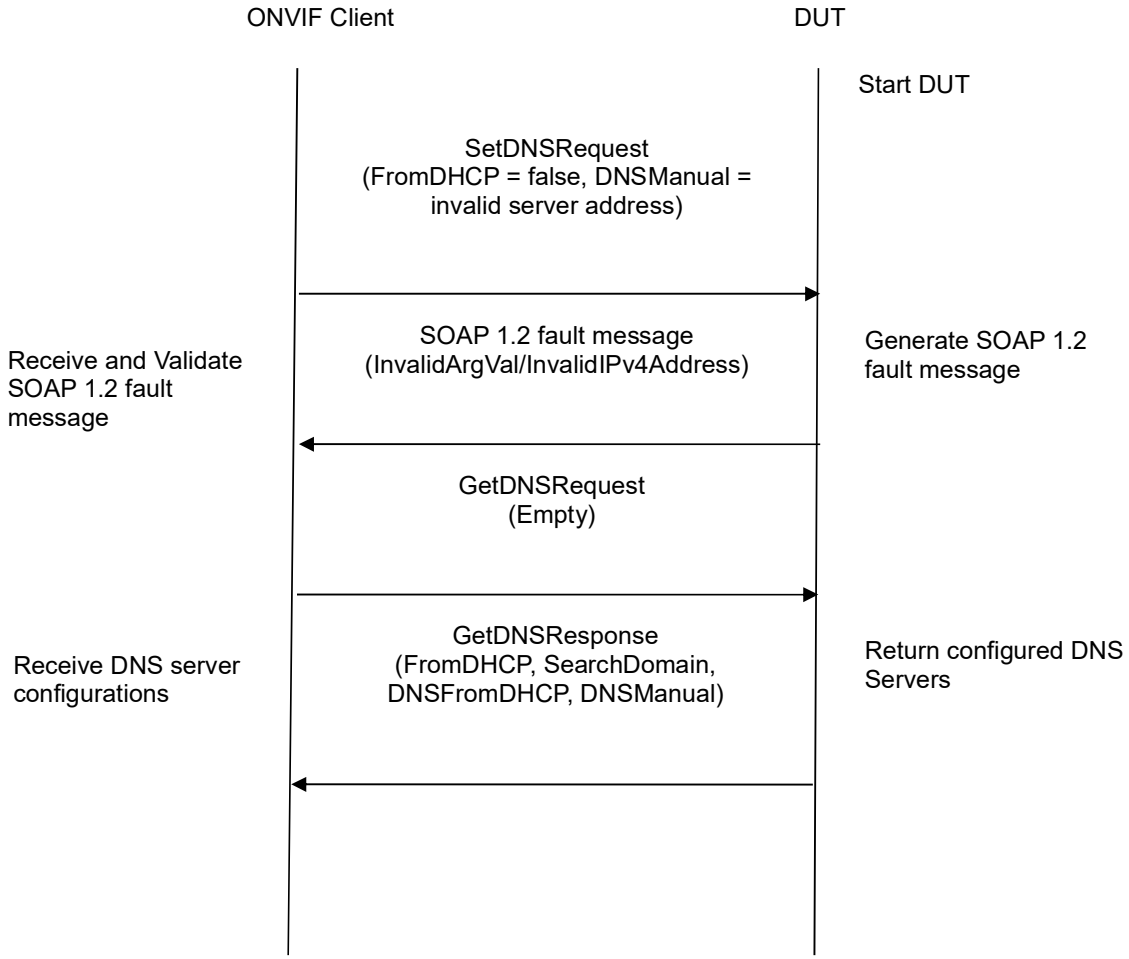
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify behavior of DUT for invalid DNS IPv4 address configuration.

Pre-Requisite: DHCP is disabled (see Annex A.13).

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke SetDNSRequest message (FromDHCP = false, DNSManual = “IPv4”, “10.1.1”).
4. Verify that the DUT generates SOAP 1.2 fault message (InvalidArgVal/InvalidIPv4Address).
5. Retrieve DNS configurations from DUT through GetDNSRequest.
6. DUT sends valid DNS configurations in the GetDNSResponse message (DNSInformation [FromDHCP=true or false, SearchDomain = domain to search if hostname is not fully qualified, DNSFromDHCP = list of DNS Servers obtained from DHCP, DNSManual = list of manual DNS Servers]).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SOAP 1.2 fault message.

The DUT did not send correct fault code in the SOAP fault message (InvalidArgVal/InvalidIPv4Address).

The DUT did not GetDNSResponse message.

The DUT returned “10.1.1” as DNS Server address.

The DUT did not send correct information (i.e. DNSInformation [FromDHCP=true or false, SearchDomain = domain to search if hostname is not fully qualified, DNSFromDHCP = list of DNS Servers obtained from DHCP, DNSManual = list of manual DNS Servers]) in the GetDNSResponse message.

Note: See Annex A.6 for Invalid IPv4 Address and SOAP 1.2 fault message definitions. See Annex A.10 for valid expression in terms of empty IP address.

6.2.9 SET DNS CONFIGURATION - DNSMANUAL INVALID IPV6

Test Label: Device Management Network Command SetDNS Test. (DNSManual = invalid IPv6 address)

Test Case ID: DEVICE-2-1-10

ONVIF Core Specification Coverage: Set DNS settings

Command Under Test: SetDNS

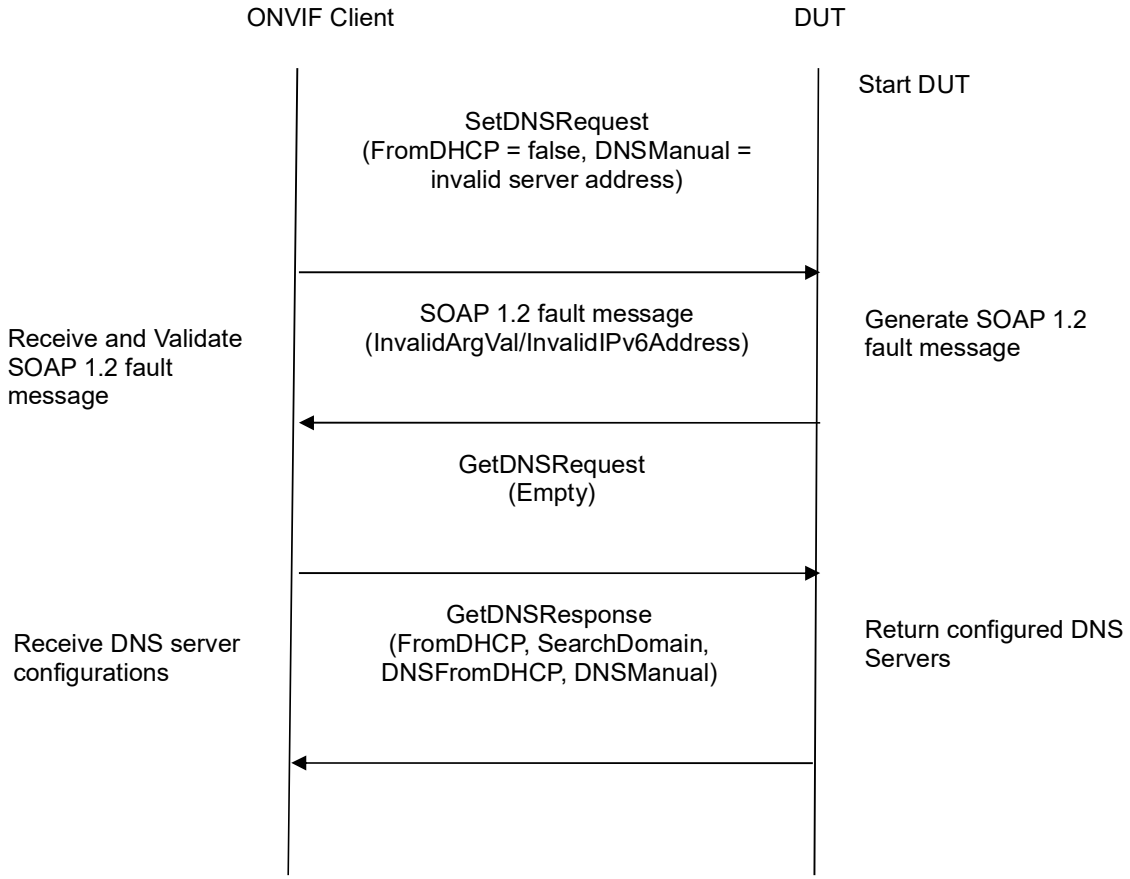
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify behavior of the DUT for invalid DNS IPv6 address configuration.

Pre-Requisite: DHCP is disabled (see Annex A.15). IPv6 is implemented by the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke SetDNSRequest message (FromDHCP = false, DNSManual = “IPv6”, “FF02:1”).
4. Verify that the DUT generates SOAP 1.2 fault message (InvalidArgVal/InvalidIPv6Address).
5. Retrieve DNS configurations from DUT through GetDNSRequest.
6. The DUT sends valid DNS configurations in the GetDNSResponse message (DNSInformation [FromDHCP=true or false, SearchDomain = domain to search if hostname is not fully qualified, DNSFromDHCP = list of DNS Servers obtained from DHCP, DNSManual = list of manual DNS Servers]).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SOAP 1.2 fault message.

The DUT did not send correct fault code in the SOAP fault message (InvalidArgVal/InvalidIPv6Address).

The DUT did not GetDNSResponse message.

The DUT returned “FF02:1” as DNS Server address.

The DUT did not send correct information (i.e. DNSInformation [FromDHCP=true or false, SearchDomain = domain to search if hostname is not fully qualified, DNSFromDHCP = list of DNS Servers obtained from DHCP, DNSManual = list of manual DNS Servers]) in the GetDNSResponse message.

6.2.10 GET NTP CONFIGURATION

Test Label: Device Management Network Command GetNTP Test

Test Case ID: DEVICE-2-1-11

ONVIF Core Specification Coverage: Get NTP settings

Command Under Test: GetNTP

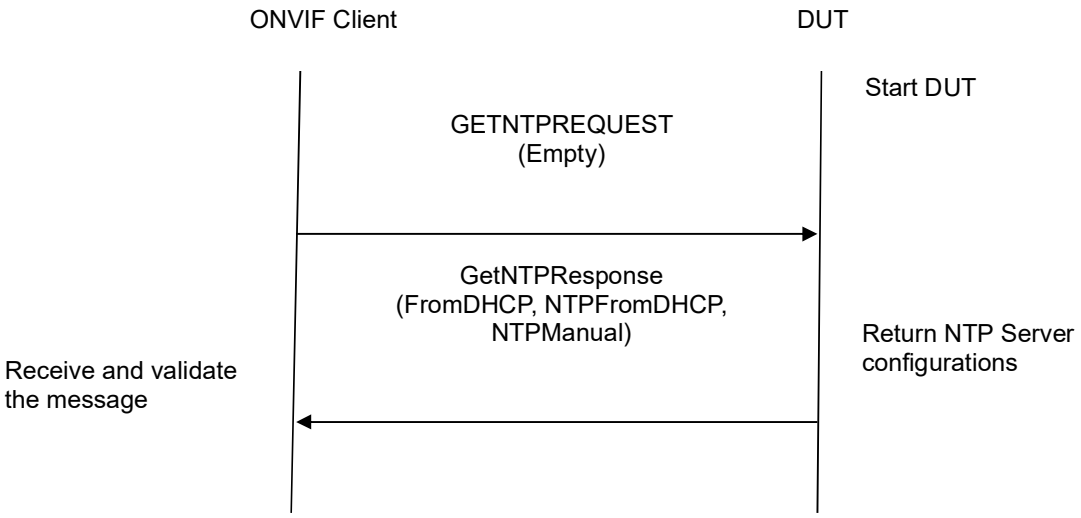
WSDL Reference: devicemgmt.wsdl

Test Purpose: To retrieve NTP Server settings of the DUT through GetNTP command.

Pre-Requisite: NTP 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 will invoke GetNTPRequest message to retrieve NTP Server settings of the DUT.
4. Verify the GetNTPResponse from the DUT (NTPInformation [FromDHCP = true or false, NTPFromDHCP = list of NTP Servers obtained from DHCP, NTPManual = list of NTP Servers manually configured]).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetNTPResponse message.

The DUT did not send correct information (i.e. NTPInformation [FromDHCP = true or false, NTPFromDHCP = list of NTP Servers obtained from DHCP, NTPManual = list of NTP Servers manually configured]) in the GetNTPResponse message.

Note: See Annex A.10 for valid expression in terms of empty IP address

6.2.11 SET NTP CONFIGURATION - NTPMANUAL IPV4

Test Label: Device Management Network Command SetNTP Test (NTPManual = IPv4 address).

Test Case ID: DEVICE-2-1-12

ONVIF Core Specification Coverage: Set NTP settings

Command Under Test: SetNTP

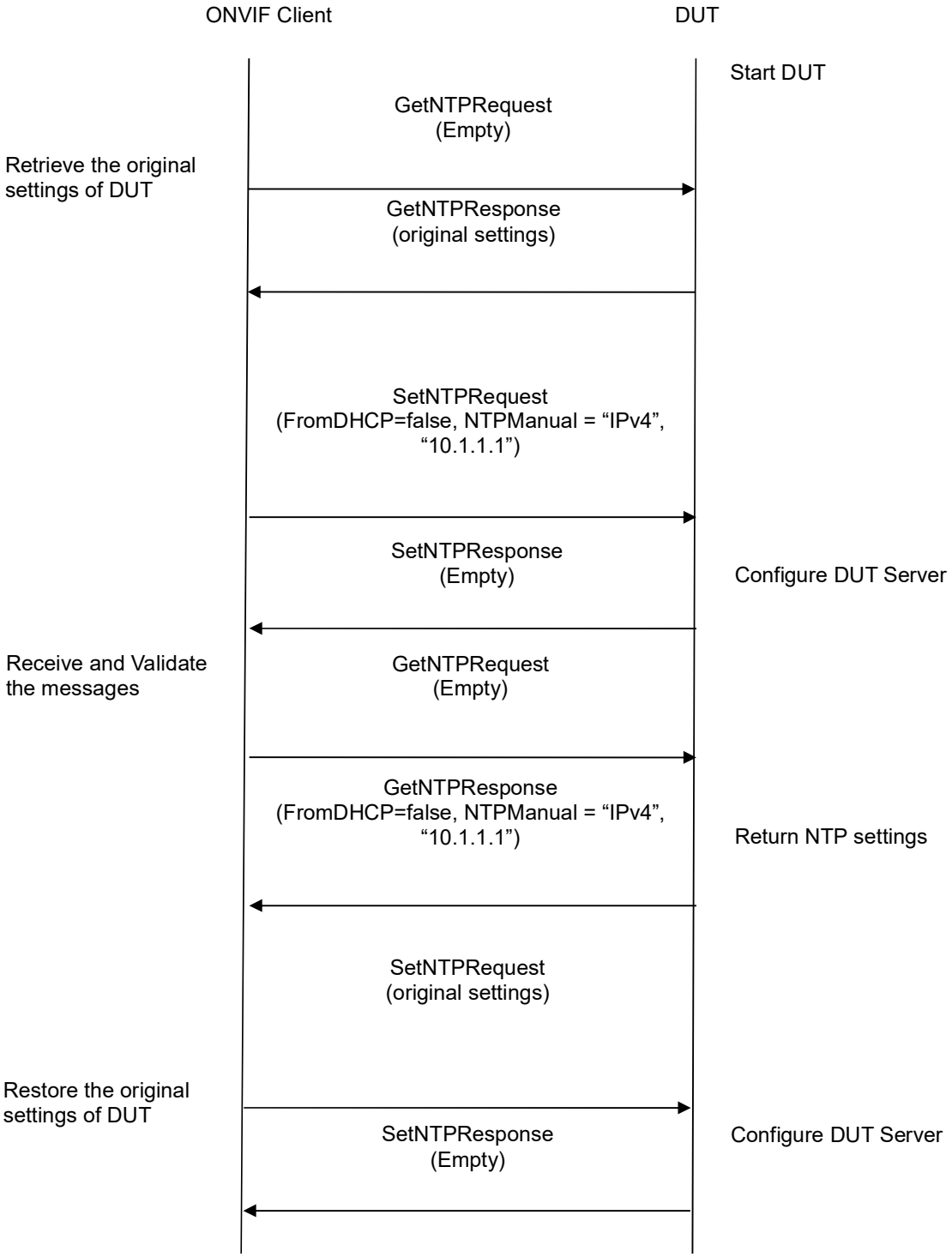
WSDL Reference: devicemgmt.wsdl

Test Purpose: To configure NTP IPv4 address settings on a DUT using SetNTP command.

Pre-Requisite: DHCP is disabled (see Annex A.13).NTP 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 will invoke GetNTPRequest message to retrieve the original settings of the DUT.
4. ONVIF Client will invoke SetNTPRequest message (FromDHCP = false, NTPManual [Type = “IPv4”, IPv4Address = “10.1.1.1”]).
5. Verify that the DUT sends SetNTPResponse (empty message).
6. Verify the NTP Server settings in DUT through GetNTPRequest message.
7. DUT sends its NTP Server settings in the GetNTPResponse message (NTPInformation [FromDHCP= false, NTPManual [Type = “IPv4”, IPv4Address = “10.1.1.1”]]).
8. ONVIF Client will invoke SetNTPRequest message to restore the original settings of DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetNTPResponse message at step-5.

The DUT did not send GetNTPResponse message at step-7.

The DUT did not send correct NTP Server information (i.e. NTPInformation [FromDHCP=false, NTPManual [Type = “IPv4”, IPv4Address = “10.1.1.1”]]) in GetNTPResponse message at step-6.

Note: See Annex A.6 for Valid IPv4 Address definition. See Annex A.10 for valid expression in terms of an empty IP address.

6.2.12 SET NTP CONFIGURATION - NTPMANUAL IPV6

Test Label: Device Management Network Command SetNTP Test (NTPManual = IPv6 address).

Test Case ID: DEVICE-2-1-13

ONVIF Core Specification Coverage: Set NTP settings

Command Under Test: SetNTP

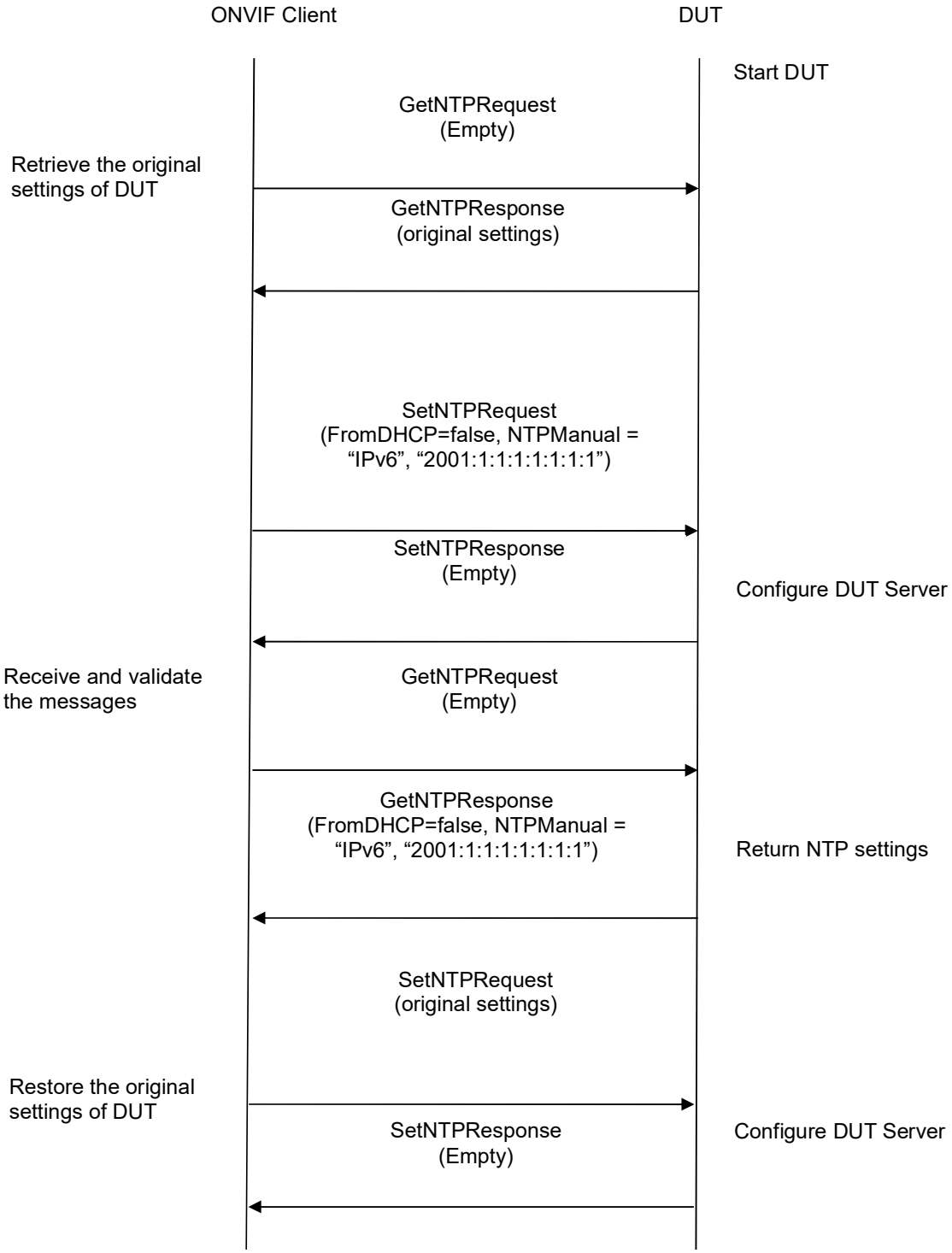
WSDL Reference: devicemgmt.wsdl

Test Purpose: To configure NTP IPv6 address settings on a DUT using SetNTP command.

Pre-Requisite: DHCP is disabled (see Annex A.15).NTP is supported by the DUT. IPv6 is implemented by the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client will invoke GetNTPRequest message to retrieve the original settings of the DUT.
4. ONVIF Client will invoke SetNTPRequest message (FromDHCP = false, NTPManual [Type = “IPv6”, IPv6Address = “2001:1:1:1:1:1:1:1”]).
5. Verify that the DUT sends SetNTPResponse (empty message).
6. Verify the NTP Server settings in DUT through GetNTPRequest message.
7. The DUT sends its NTP Server settings in the GetNTPResponse message (NTPInformation [FromDHCP= false, NTPManual [Type = “IPv6”, IPv6Address = “2001:1:1:1:1:1:1:1”]]).
8. ONVIF Client will invoke SetNTPRequest message to restore the original settings of the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetNTPResponse message at step-5.

The DUT did not send GetNTPResponse message at step-7.

The DUT did not send correct NTP Server information (i.e. NTPInformation [FromDHCP=false, NTPManual [Type = “IPv6”, IPv6Address = “2001:1:1:1:1:1:1:1”]]) in GetNTPResponse message at step 7.

6.2.13 SET NTP CONFIGURATION - FROMDHCP

Test Label: Device Management Network Command SetNTP FromDHCP Test.

Test Case ID: DEVICE-2-1-14

ONVIF Core Specification Coverage: Set NTP settings

Command Under Test: SetNTP

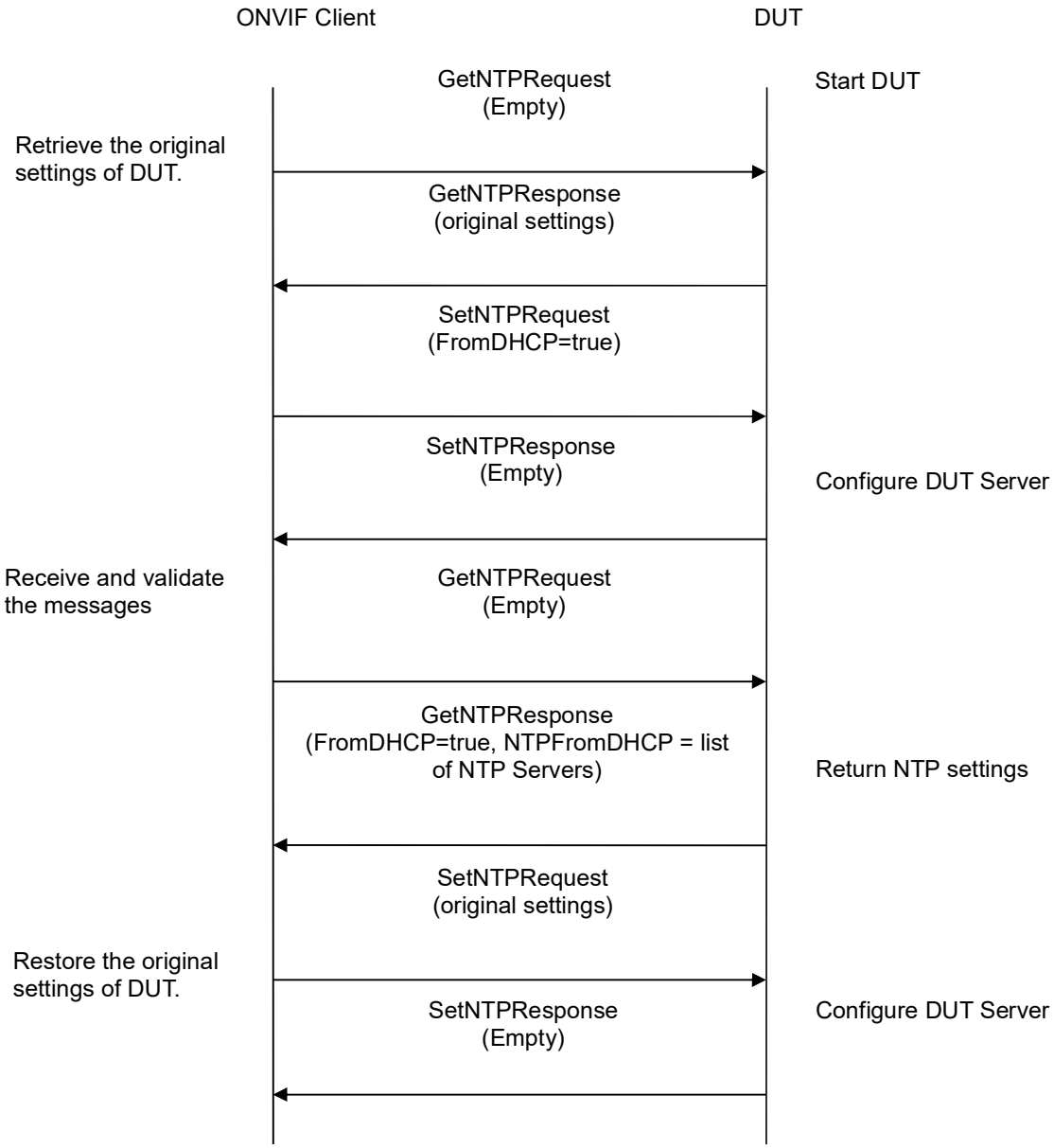
WSDL Reference: devicemgmt.wsdl

Test Purpose: To configure DUT's NTP settings via DHCP server using SetNTP command.

Pre-Requisite: DHCP is enabled (see Annex A.12 and A.14). NTP is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetNTPRequest message to retrieve the original settings of the DUT.
4. ONVIF Client will invoke SetNTPRequest message (FromDHCP = true).
5. Verify that the DUT sends SetNTPResponse (empty message).

6. Verify the NTP Server settings in DUT through GetNTPRequest message.
7. The DUT sends its NTP Server settings in the GetNTPResponse message (NTPInformation [FromDHCP= true, NTPFromDHCP = list of NTP Servers obtained from DHCP]).
8. ONVIF Client will invoke SetNTPRequest message to restore the original settings of DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetNTPResponse message at step-5.

The DUT did not send GetNTPResponse message at step-7.

The DUT did not send correct information (i.e. NTPInformation [FromDHCP = true, NTPFromDHCP = list of NTP Servers obtained from DHCP, the list can be empty if the DHCP server does not deliver NTP Addresses]) in the GetNTPResponse message at step-7.

6.2.14 SET NTP CONFIGURATION - NTPMANUAL INVALID IPV4

Test Label: Device Management Network Command SetNTP Test (NTPManual = invalid IPv4 address).

Test Case ID: DEVICE-2-1-15

ONVIF Core Specification Coverage: Set NTP settings

Command Under Test: SetNTP

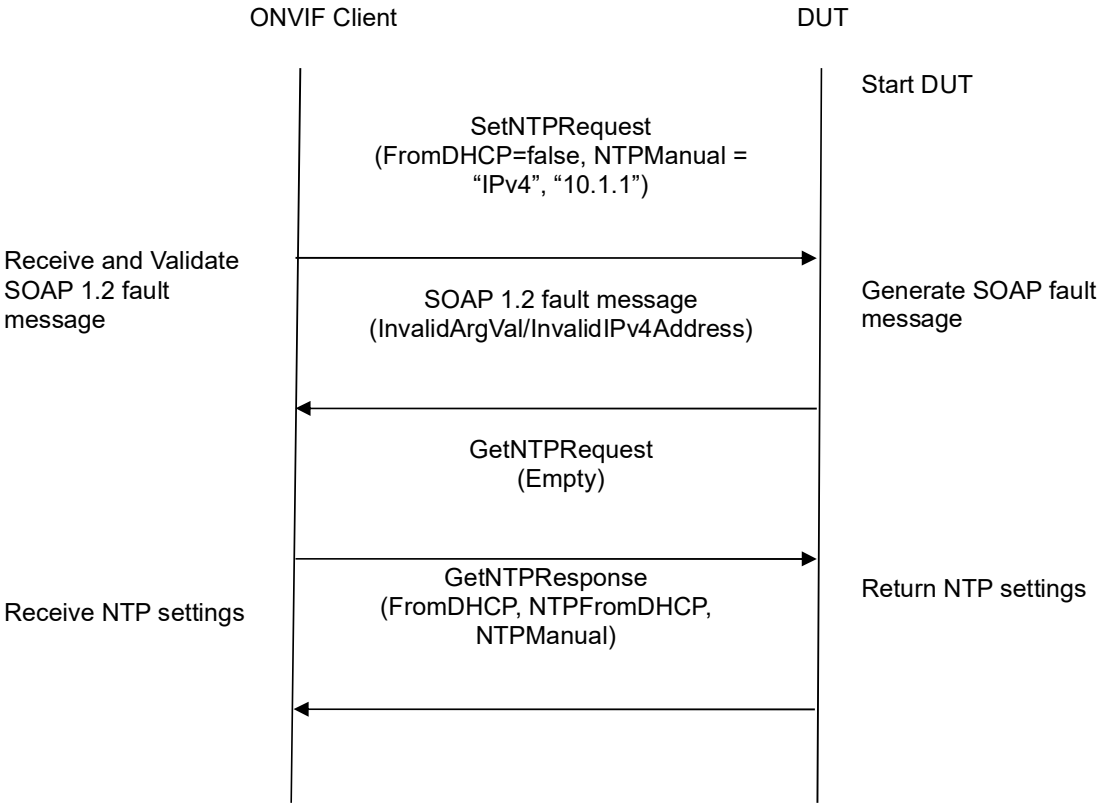
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify behavior of the DUT for invalid NTP IPv4 address configuration.

Pre-Requisite: DHCP is disabled (see Annex A.13). NTP 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 will invoke SetNTPRequest message (FromDHCP = false, NTPManual [Type = “IPv4”, IPv4Address = “10.1.1”]).
4. Verify that the DUT generates SOAP 1.2 fault message (InvalidArgVal/InvalidIPv4Address).
5. Retrieve NTP Server configurations from DUT through GetNTPRequest message.
6. The DUT sends valid NTP Server configurations in the GetNTPResponse message (NTPInformation [FromDHCP = true or false, NTPFromDHCP = list of NTP Servers obtained from DHCP, NTPManual = list of NTP Servers manually configured]).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SOAP 1.2 fault message.

The DUT did not send correct fault code in the SOAP fault message (InvalidArgVal/InvalidIPv4Address).

The DUT did not GetNTPResponse message.

The DUT returned “10.1.1” as NTP Server address.

The DUT did not send correct NTP Server information (i.e. NTPInformation [FromDHCP = true or false, NTPFromDHCP = list of NTP Servers obtained from DHCP, NTPManual = list of NTP Servers manually configured]) in GetNTPResponse message.

Note: See Annex A.6 for Invalid IPv4 Address and SOAP 1.2 fault message definitions.

6.2.15 SET NTP CONFIGURATION - NTPMANUAL INVALID IPV6

Test Label: Device Management Network Command SetNTP Test (NTPManual = invalid IPv6 address).

Test Case ID: DEVICE-2-1-16

ONVIF Core Specification Coverage: Set NTP settings

Command Under Test: SetNTP

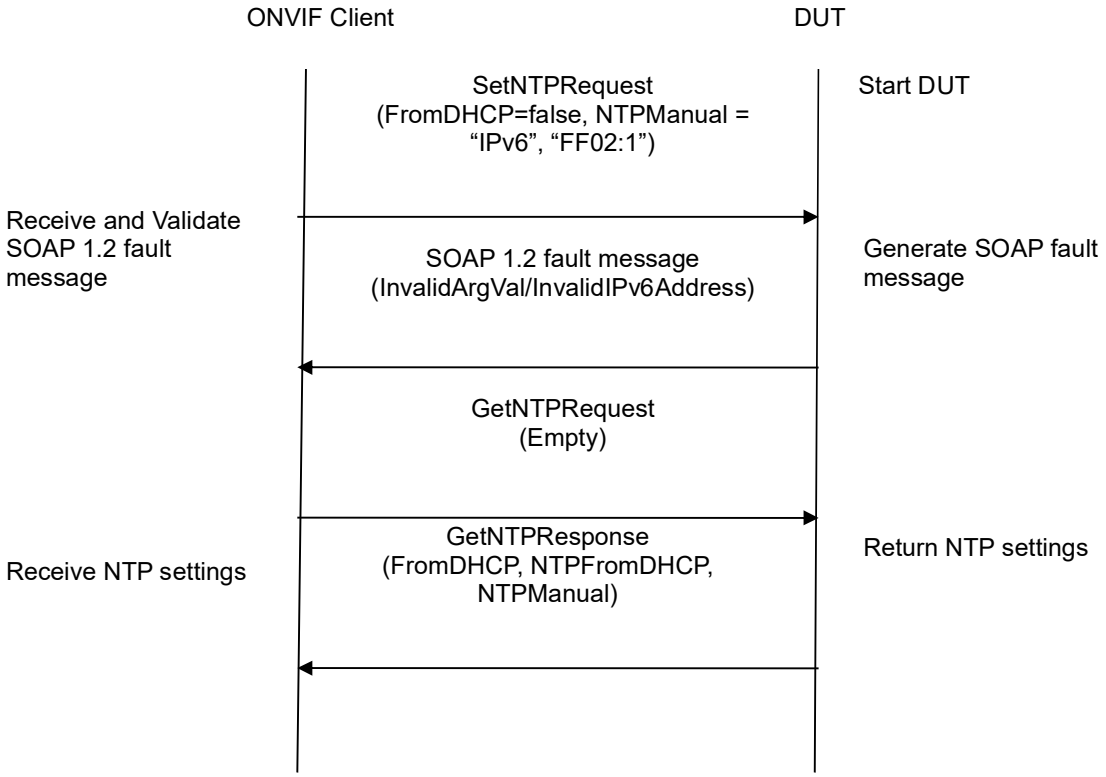
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify behavior of the DUT for invalid NTP IPv6 address configuration.

Pre-Requisite: DHCP is disabled (see Annex A.15). NTP is supported by DUT. IPv6 is implemented by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke SetNTPRequest message (FromDHCP = false, NTPManual [Type = “IPv6”, IPv6Address = “FF02:1”]).
4. Verify that the DUT generates SOAP 1.2 fault message (InvalidArgVal/InvalidIPv6Address).
5. Retrieve NTP Server configurations from DUT through GetNTPRequest message.
6. The DUT sends valid NTP Server configurations in the GetNTPResponse message (NTPInformation [FromDHCP = true or false, NTPFromDHCP = list of NTP Servers obtained from DHCP, NTPManual = list of NTP Servers manually configured]).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SOAP 1.2 fault message.

The DUT did not send correct fault code in the SOAP fault message (InvalidArgVal/InvalidIPv6Address).

The DUT did not GetNTPResponse message.

The DUT returned “FF02:1” as NTP Server address.

The DUT did not send correct NTP Server information (i.e. NTPInformation [FromDHCP = true or false, NTPFromDHCP = list of NTP Servers obtained from DHCP, NTPManual = list of NTP Servers manually configured]) in GetNTPResponse message.

6.2.16 GET NETWORK INTERFACE CONFIGURATION

Test Label: Device Management Network Command GetNetworkInterfaces Test.

Test Case ID: DEVICE-2-1-17

ONVIF Core Specification Coverage: Get network interface configuration

Command Under Test: GetNetworkInterfaces

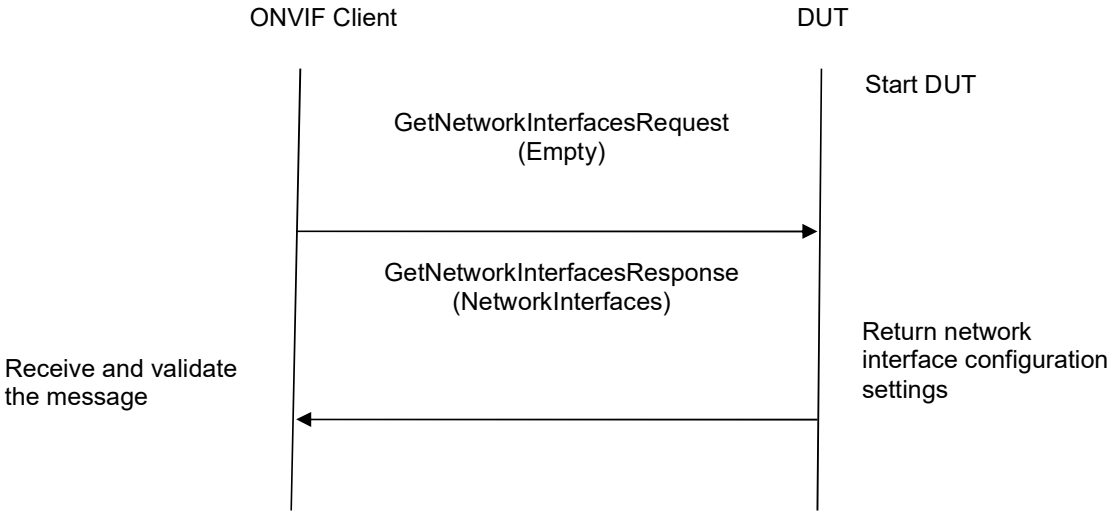
WSDL Reference: devicemgmt.wsdl

Test Purpose: To retrieve network interface configurations of DUT through GetNetworkInterfaces command.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetNetworkInterfacesRequest message to retrieve network interface configuration settings of the DUT.

4. Verify the GetNetworkInterfacesResponse from DUT (NetworkInterfaces = list of network interfaces).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetNetworkInterfacesResponse message.

The DUT did not send correct network interface information (i.e. NetworkInterfaces = list of network interfaces) in GetNetworkInterfacesResponse message.

6.2.17 SET NETWORK INTERFACE CONFIGURATION - IPV4

Test Label: Device Management Network Command SetNetworkInterfaces Test. (for IPv4 address)

Test Case ID: DEVICE-2-1-18

ONVIF Core Specification Coverage: Set network interface configuration

Command Under Test: SetNetworkInterfaces

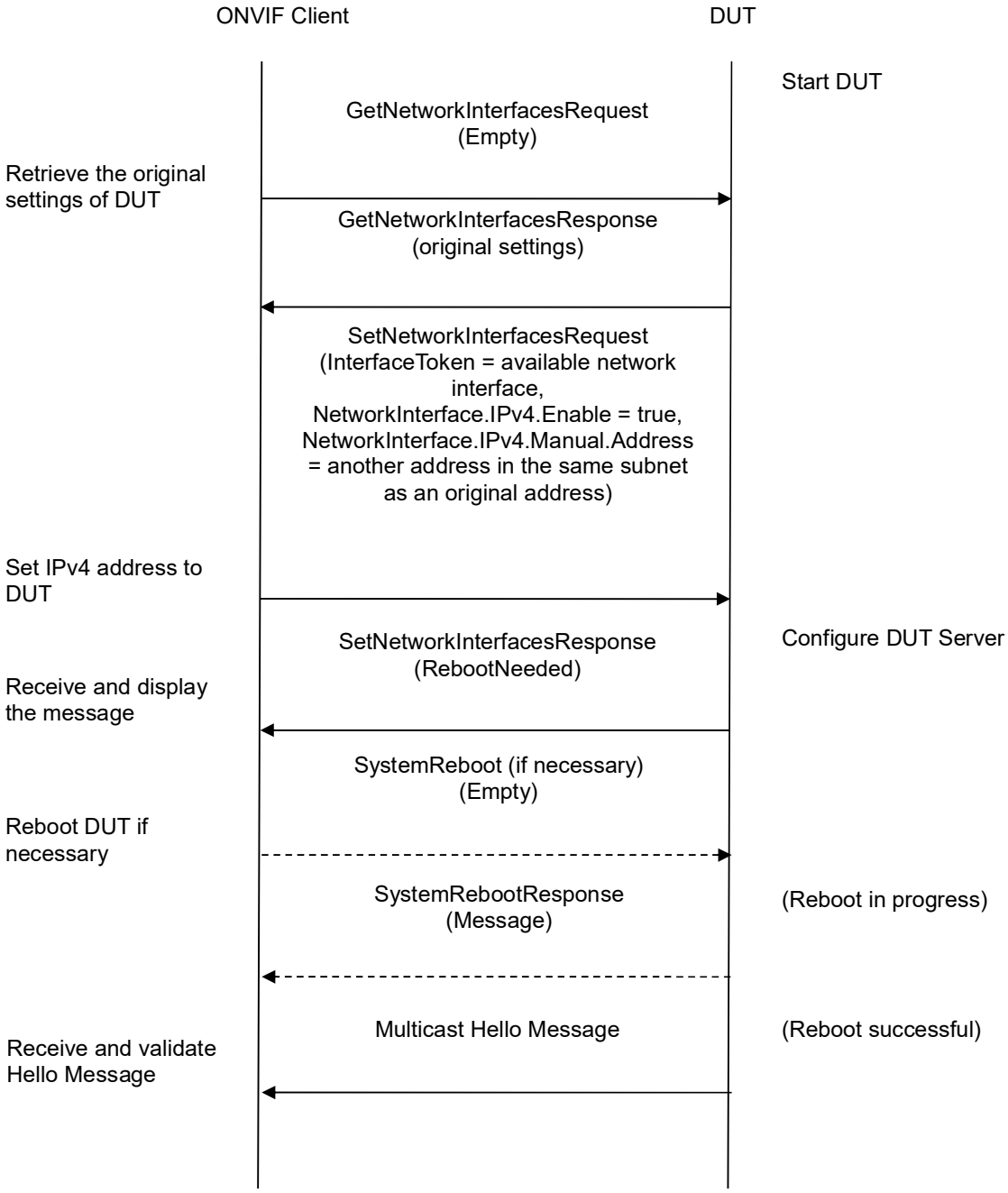
WSDL Reference: devicemgmt.wsdl

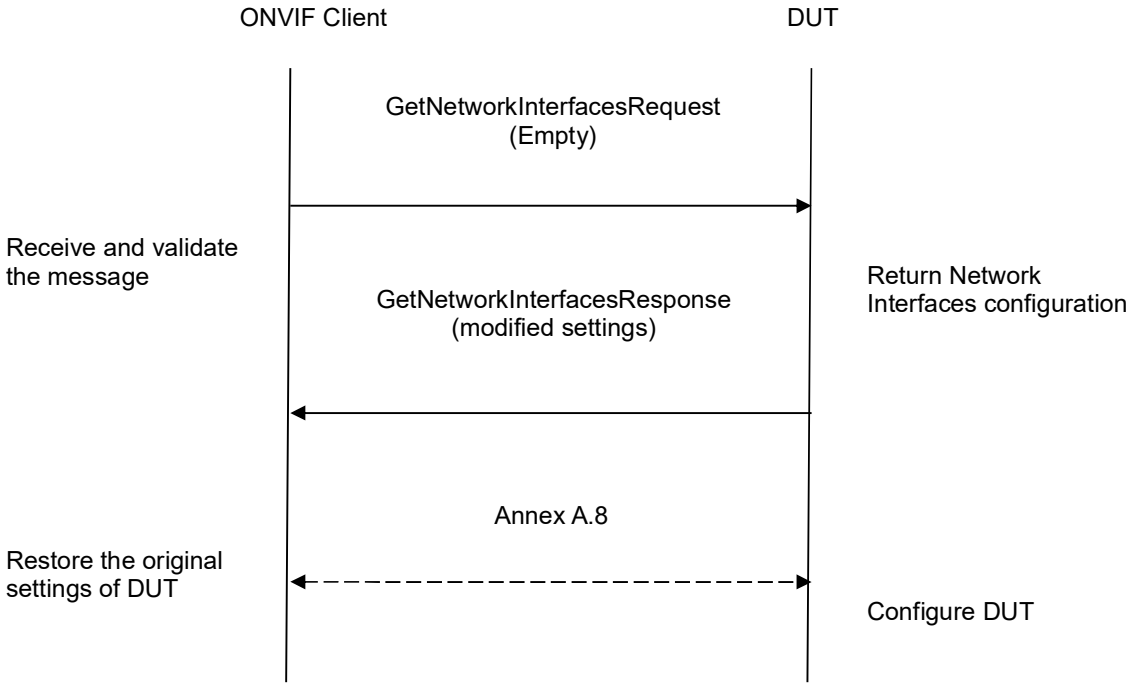
Test Purpose: To configure IPv4 address setting on a DUT using SetNetworkInterfaces command.

Pre-Requisite: ONVIF Client knows the available network interface token of DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
GetNetworkInterfacesRequest
(Empty)
Retrieve the original settings of DUT
GetNetworkInterfacesResponse
(original settings)
SetNetworkInterfacesRequest
(InterfaceToken = available network interface,
NetworkInterface.IPv4.Enable = true,
NetworkInterface.IPv4.Manual.Address = another address in the same subnet as an original address)
Set IPv4 address to DUT
SetNetworkInterfacesResponse
(RebootNeeded)
Configure DUT Server
Receive and display the message
SystemReboot (if necessary)
(Empty)
Reboot DUT if necessary
SystemRebootResponse
(Message)
(Reboot in progress)
Multicast Hello Message
(Reboot successful)
Receive and validate Hello Message



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetNetworkInterfacesRequest message to retrieve the original settings of DUT.
4. ONVIF Client will invoke SetNetworkInterfacesRequest message to set static IPv4 address to DUT (InterfaceToken = available network interface, NetworkInterface.IPv4.Enable = true, NetworkInterface.IPv4.Manual.Address = another address in the same subnet as an original address).
5. The DUT will return SetNetworkInterfacesResponse.
6. If necessary, ONVIF Client will invoke SystemReboot message to restart DUT. Otherwise, go to step-8.
7. The DUT will return SystemRebootResponse message.
8. The DUT will send Multicast Hello message from newly configured address.
9. ONVIF Client will receive and validate Hello message sent from newly configured address by the DUT.
10. Verify the network interfaces settings in DUT through GetNetworkInterfacesRequest message.
11. The DUT sends its network interfaces settings in the GetNetworkInterfacesResponse message (i.e. NetworkInterface [token = available network interface, IPv4.Enable = true, IPv4.Config.Manual = newly configured address]) from newly configured address.

12. ONVIF Client will restore the original settings by following the procedure mentioned in Annex A.8.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetNetworkInterfacesResponse message.

The DUT did not send SystemRebootResponse message.

The DUT did not send Multicast Hello message after IP configuration change.

The DUT did not send GetNetworkInterfacesResponse message.

The DUT did not send correct network interface information (i.e. NetworkInterface [token = available network interface, IPv4.Enable = true, IPv4.Config.Manual = newly configured address]) in GetNetworkInterfacesResponse message.

6.2.18 SET NETWORK INTERFACE CONFIGURATION - IPV6

Test Label: Device Management Network Command SetNetworkInterfaces Test (for IPv6 address).

Test Case ID: DEVICE-2-1-19

ONVIF Core Specification Coverage: Set network interface configuration

Command Under Test: SetNetworkInterfaces

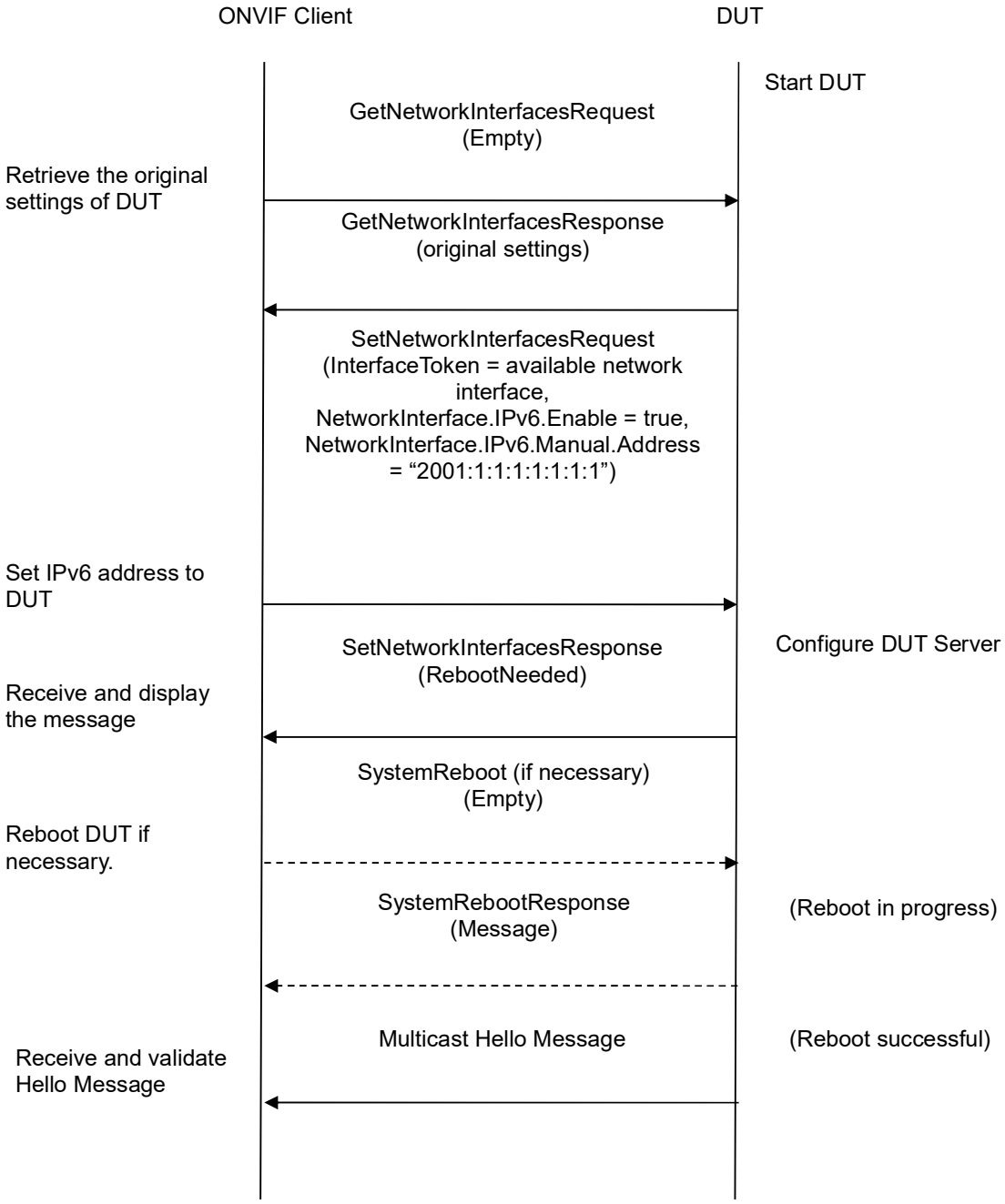
WSDL Reference: devicemgmt.wsdl

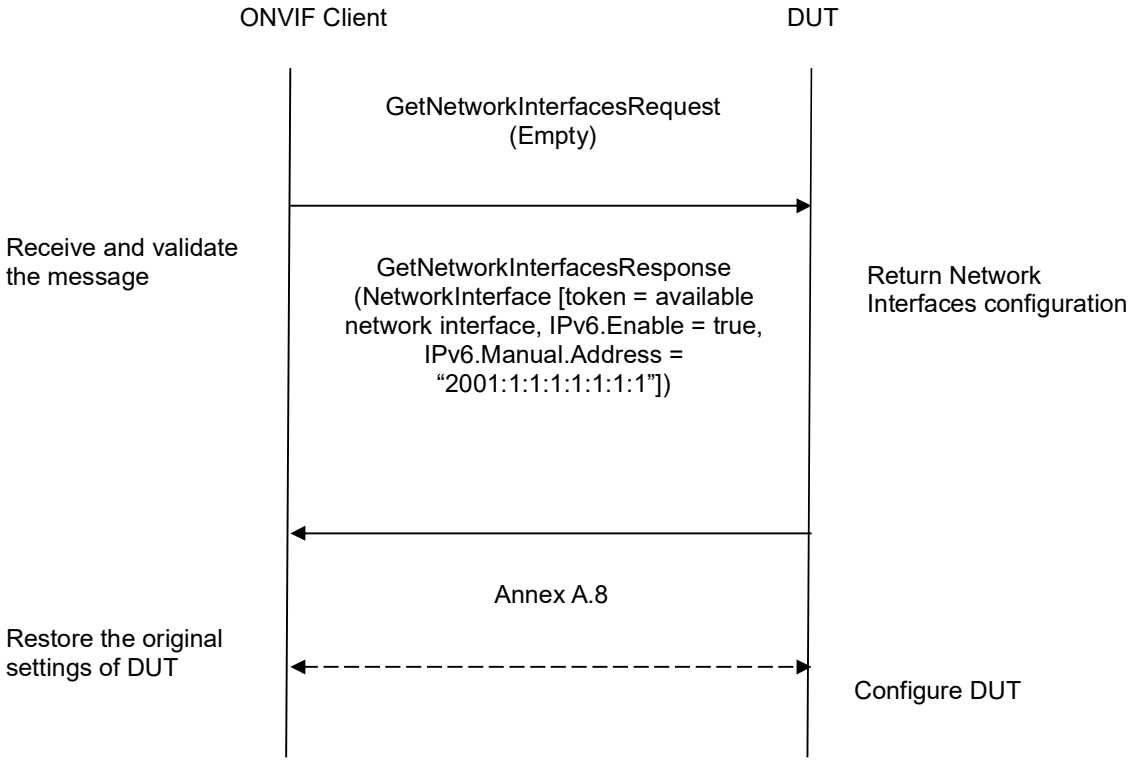
Test Purpose: To configure IPv6 address setting on a DUT through SetNetworkInterfaces command.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
GetNetworkInterfacesRequest
(Empty)
Retrieve the original settings of DUT
GetNetworkInterfacesResponse
(original settings)
SetNetworkInterfacesRequest
(InterfaceToken = available network interface,
NetworkInterface.IPv6.Enable = true,
NetworkInterface.IPv6.Manual.Address = “2001:1:1:1:1:1:1:1”)
Set IPv6 address to DUT
SetNetworkInterfacesResponse
(RebootNeeded)
Configure DUT Server
Receive and display the message
SystemReboot (if necessary)
(Empty)
Reboot DUT if necessary.
SystemRebootResponse
(Message)
(Reboot in progress)
Multicast Hello Message
(Reboot successful)
Receive and validate Hello Message



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetNetworkInterfacesRequest message to retrieve the original settings of the DUT.
4. ONVIF Client will invoke SetNetworkInterfacesRequest message to set static IPv6 address to DUT (InterfaceToken = available network interface, NetworkInterface.IPv6.Enable = true, NetworkInterface.IPv6.Manual.Address = "2001:1:1:1:1:1:1:1").
5. The DUT will return SetNetworkInterfacesResponse.
6. If necessary, ONVIF Client will invoke SystemReboot message to restart DUT. Otherwise, go to step-8.
7. The DUT will return SystemRebootResponse message.
8. The DUT will send Multicast Hello message.
9. ONVIF Client will receive and validate Hello message sent by the DUT.
10. Verify the network interfaces settings in DUT through GetNetworkInterfacesRequest message.

11. The DUT sends its network interfaces settings in the GetNetworkInterfacesResponse message (i.e. NetworkInterface [token = available network interface, IPv6.Enable = true, IPv6.Config.Manual = “2001:1:1:1:1:1:1:1”]).
12. ONVIF Client will restore the original settings by following the procedure mentioned in Annex A.8.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetNetworkInterfacesResponse message.

The DUT did not send SystemRebootResponse message.

The DUT did not send Multicast Hello message after IP configuration change.

The DUT did not send GetNetworkInterfacesResponse message.

The DUT did not send correct network interface information (i.e. NetworkInterface [token = available network interface, IPv6.Enable = true, IPv6.Config.Manual = “2001:1:1:1:1:1:1:1”]) in GetNetworkInterfacesResponse message.

6.2.19 SET NETWORK INTERFACE CONFIGURATION - INVALID IPV4

Test Label: Device Management Network Command SetNetworkInterfaces Test. (for invalid IPv4 address)

Test Case ID: DEVICE-2-1-20

ONVIF Core Specification Coverage: Set network interface configuration

Command Under Test: SetNetworkInterfaces

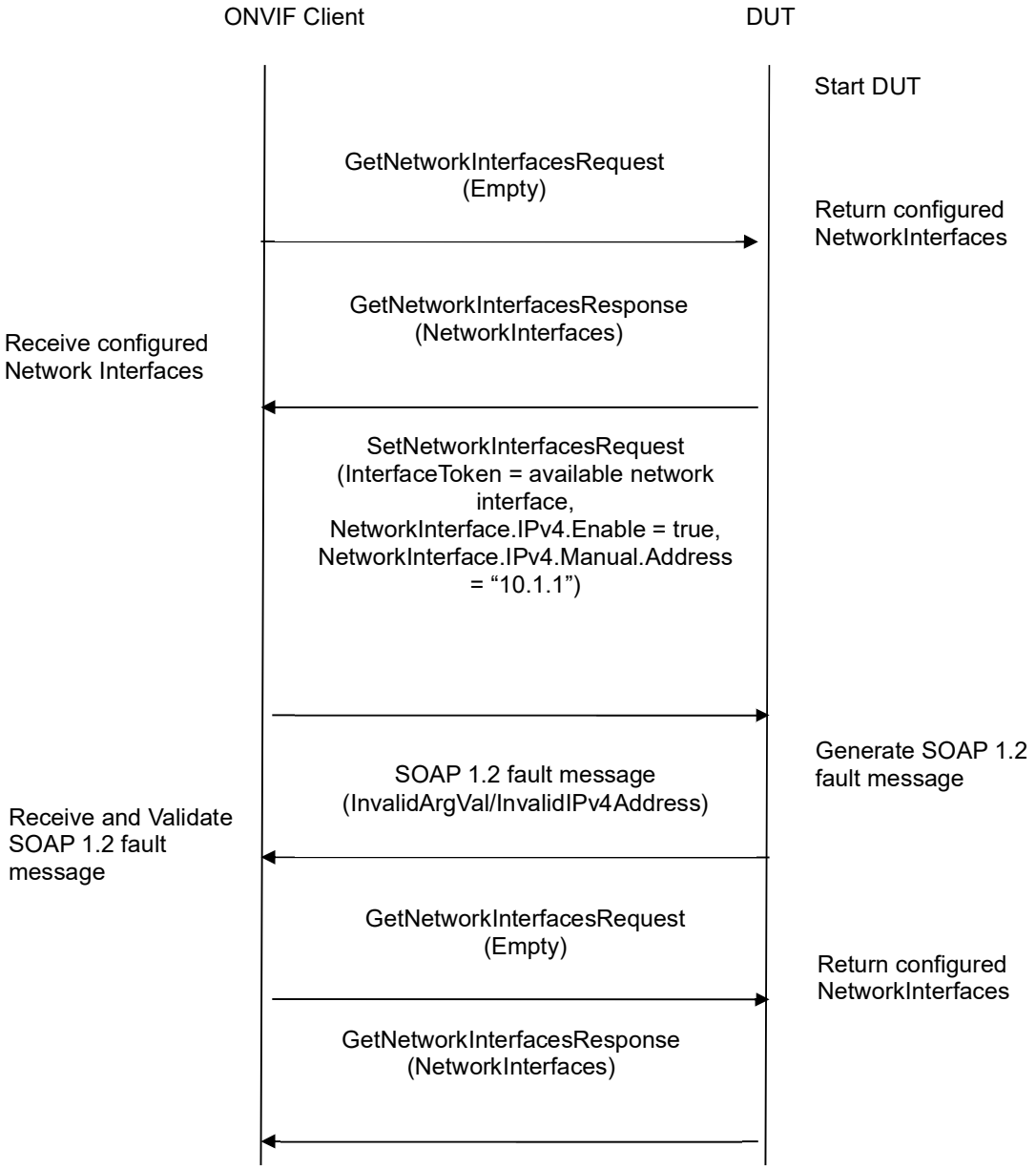
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify behavior of the DUT for invalid IPv4 address configuration.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetNetworkInterfacesRequest to retrieve the configured network interfaces of the device.
4. The DUT returns its network interface settings.

5. ONVIF Client will invoke SetNetworkInterfacesRequest message (InterfaceToken = available network interface, NetworkInterface.IPv4.Enable = true, NetworkInterface.IPv4.Manual.Address = “10.1.1”).
6. Verify that the DUT generates SOAP 1.2 fault message (InvalidArgVal/InvalidIPv4Address).
7. Retrieve network interface configurations from DUT through GetNetworkInterfacesRequest message.
8. DUT sends valid network interface configurations in the GetNetworkInterfacesResponse message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SOAP 1.2 fault message.

The DUT did not send correct fault code in the SOAP fault message (InvalidArgVal/InvalidIPv4Address).

The DUT did not send GetNetworkInterfacesResponse message.

The DUT returned “10.1.1” as DUT IPv4 address.

The DUT did not send correct network interface information (i.e. NetworkInterfaces = list of network interfaces) in GetNetworkInterfacesResponse message.

Note: See Annex A.6 for Invalid IPv4 Address and SOAP 1.2 fault message definitions.

6.2.20 SET NETWORK INTERFACE CONFIGURATION - INVALID IPV6

Test Label: Device Management Network Command SetNetworkInterfaces Test. (for invalid IPv6 address)

Test Case ID: DEVICE-2-1-21

ONVIF Core Specification Coverage: Set network interface configuration

Command Under Test: SetNetworkInterfaces

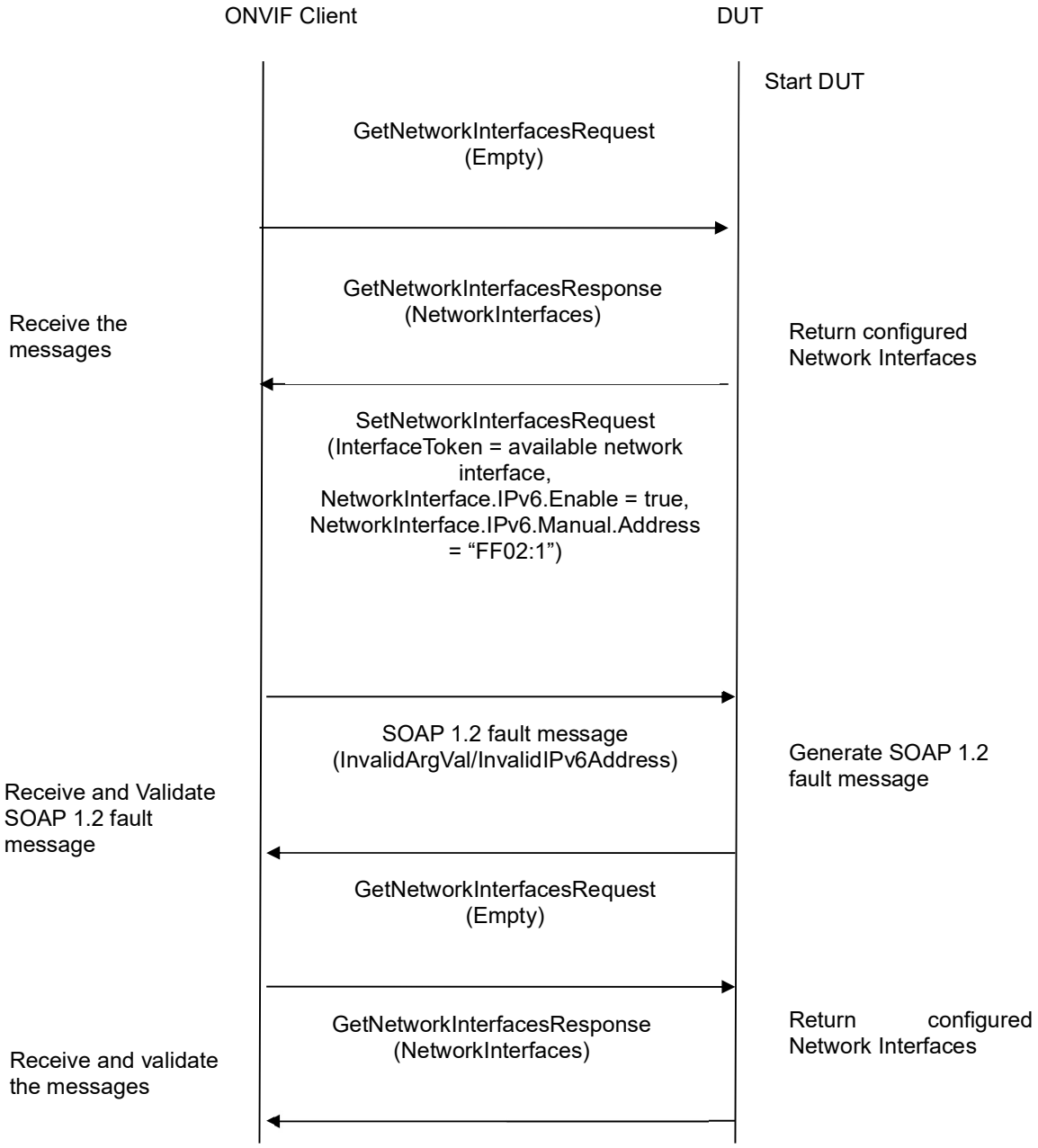
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify behavior of the DUT for invalid IPv6 address configuration.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetNetworkInterfacesRequest to get the network interface settings of the device.

4. The device returns its network interfaces.
5. ONVIF Client will invoke SetNetworkInterfacesRequest message (InterfaceToken = available network interface, NetworkInterface.IPv6.Enable = true, NetworkInterface.IPv6.Manual.Address = “FF02:1”).
6. Verify that the DUT generates SOAP 1.2 fault message (InvalidArgVal/InvalidIPv6Address).
7. Retrieve network interface configurations from DUT through GetNetworkInterfacesRequest message.
8. The DUT sends valid network interface configurations in the GetNetworkInterfacesResponse message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SOAP 1.2 fault message.

The DUT did not send correct fault code in the SOAP fault message (InvalidArgVal/InvalidIPv6Address).

The DUT did not send GetNetworkInterfacesResponse message.

The DUT returned “FF02:1” as DUT IPv6 address.

The DUT did not send correct network interface information (i.e. NetworkInterfaces = list of network interfaces) in GetNetworkInterfacesResponse message.

6.2.21 GET NETWORK PROTOCOLS CONFIGURATION

Test Label: Device Management Network Command GetNetworkProtocols Test.

Test Case ID: DEVICE-2-1-33

ONVIF Core Specification Coverage: Get network protocols

Command Under Test: GetNetworkProtocols

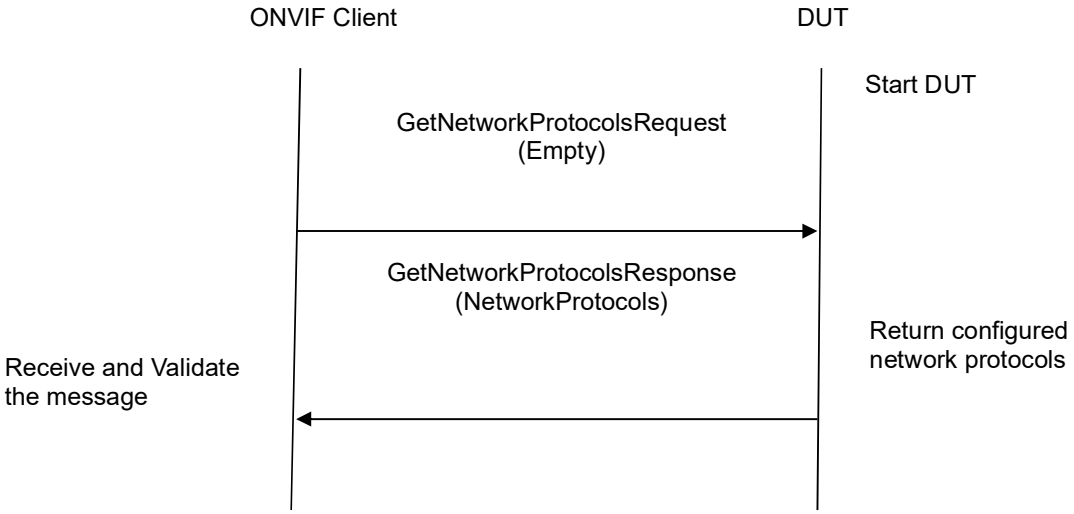
WSDL Reference: devicemgmt.wsdl

Test Purpose: To retrieve network protocols configurations of the DUT using GetNetworkProtocols command.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetNetworkProtocolsRequest message to retrieve configured network protocols of the DUT.
4. Verify the GetNetworkProtocolsResponse from DUT (NetworkProtocols = list of configured network protocols).
5. Check that the mandatory HTTP protocol is present on the list.
6. Check that the RTSP protocol is present on the list, if Real-time Streaming or Replay Service is supported by the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetNetworkProtocolsResponse message.

The DUT did not send correct information in the GetNetworkProtocolsResponse message (i.e. NetworkProtocols = list of configured network protocols, contains HTTP and contains RTSP Real-time Streaming or Replay Service is supported by the DUT).

6.2.22 SET NETWORK PROTOCOLS CONFIGURATION

Test Label: Device Management Network Command SetNetworkProtocols Test.

Test Case ID: DEVICE-2-1-34

ONVIF Core Specification Coverage: Set network protocols

Command Under Test: SetNetworkProtocols

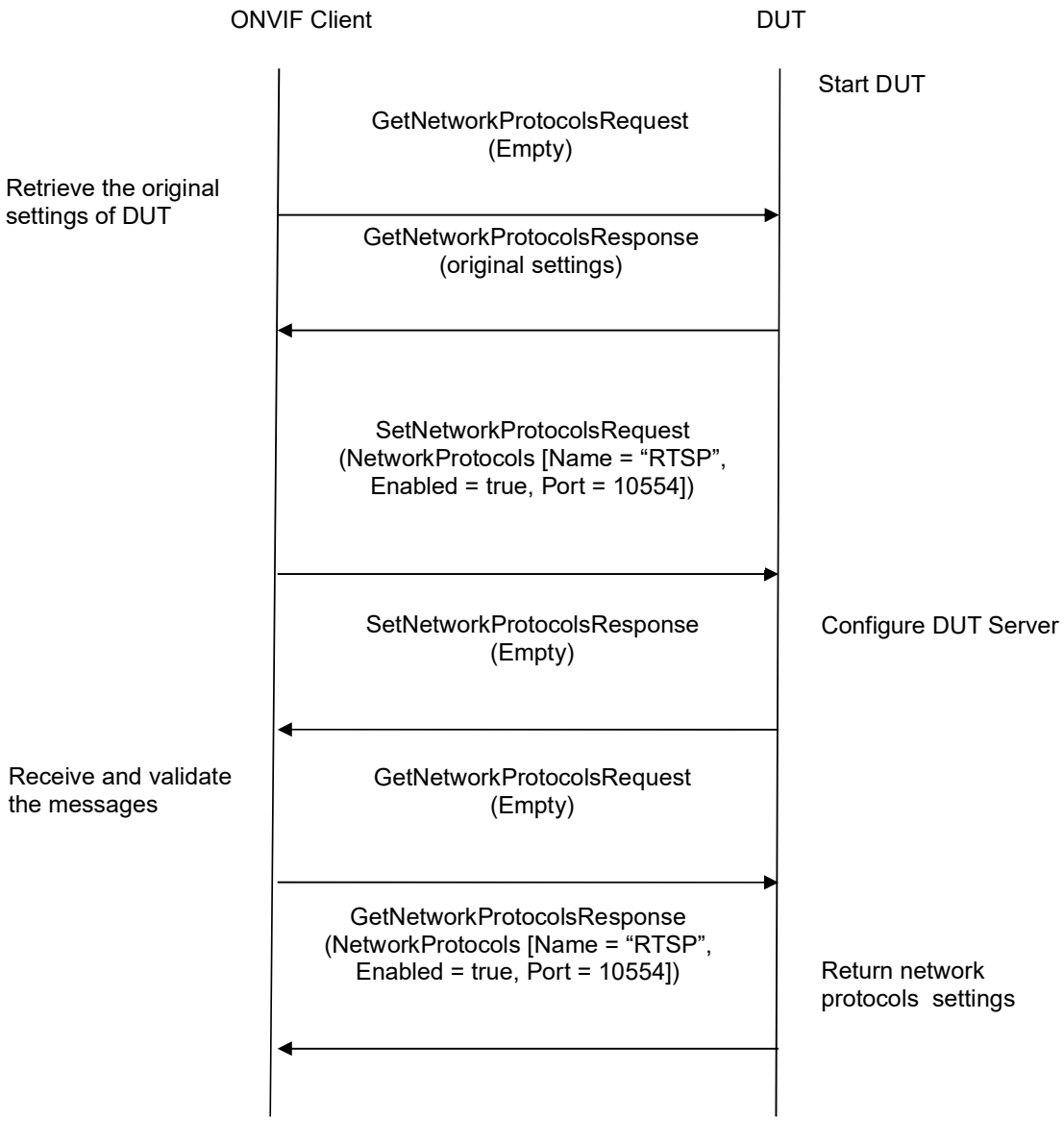
WSDL Reference: devicemgmt.wsdl

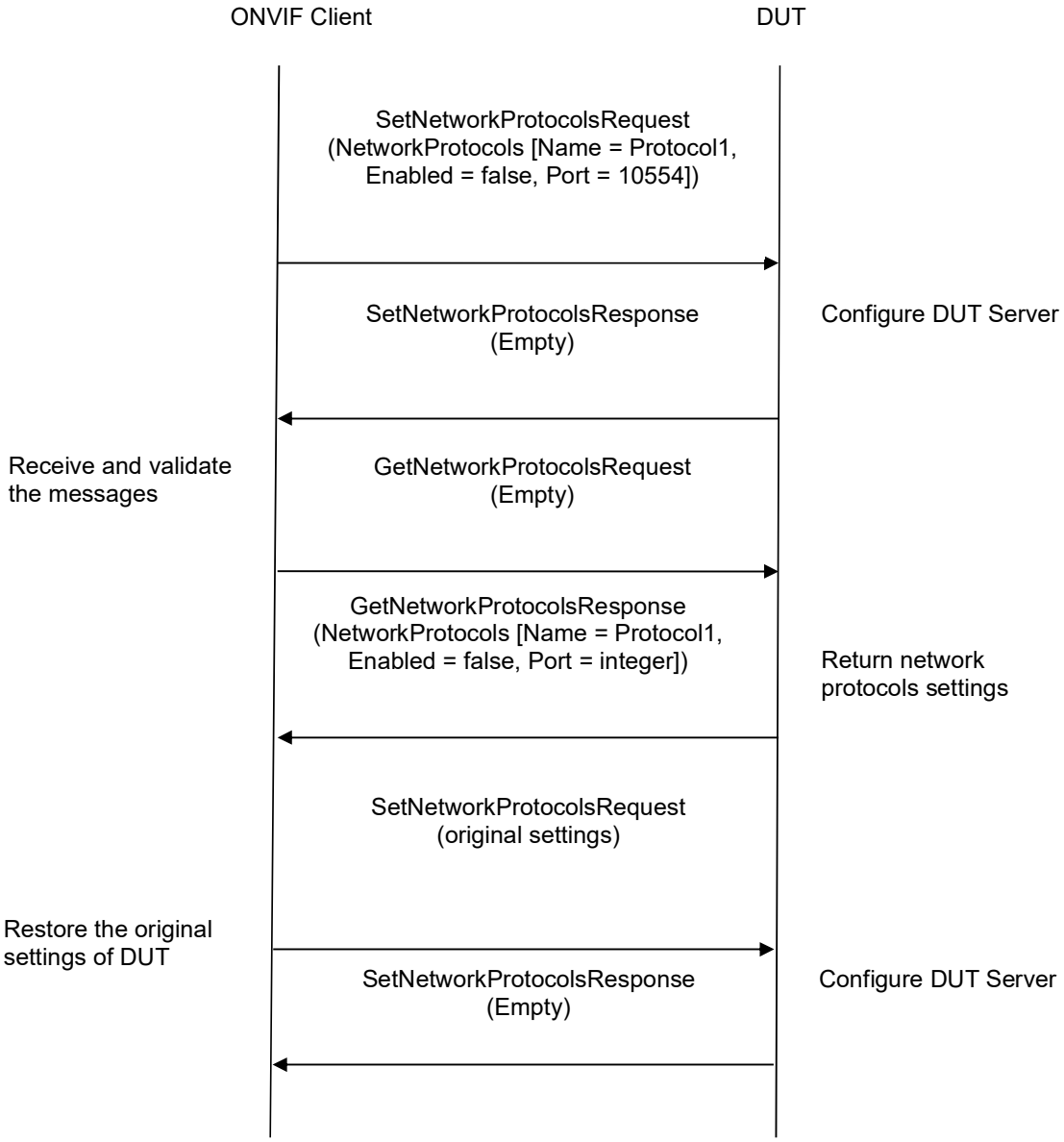
Test Purpose: To configure network protocols setting on a DUT using SetNetworkProtocols command.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:





Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetNetworkProtocolsRequest message to retrieve the original settings of the DUT.
4. If RTSP protocol is not supported, skip other steps and go to the next test.
5. ONVIF Client will invoke SetNetworkProtocolsRequest message (NetworkProtocols [Name = “RTSP”, Enabled = true, Port = 10554]).

6. Verify that the DUT sends SetNetworkProtocolsResponse (empty message).
7. Verify the network protocols settings in DUT through GetNetworkProtocolsRequest message.
8. The DUT sends its network protocols settings in the GetNetworkProtocolsResponse message (NetworkProtocols [Name = “RTSP”, Enabled = true, Port = 10554]).
9. ONVIF Client will invoke SetNetworkProtocolsRequest message (NetworkProtocols [Name = “RTSP”, Enabled = false, Port = 10554]).
10. Verify that the DUT sends SetNetworkProtocolsResponse (empty message).
11. Verify the network protocols settings in DUT through GetNetworkProtocolsRequest message.
12. DUT sends its network protocols settings in the GetNetworkProtocolsResponse message (NetworkProtocols [Name = “RTSP”, Enabled = false, Port = integer]).
13. ONVIF Client will invoke SetNetworkProtocolsRequest message to restore the original settings of DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetNetworkProtocolsResponse message.

The DUT did not send GetNetworkProtocolsResponse message.

The DUT did not send correct network protocols information (i.e. NetworkProtocols [Name =”RTSP”, Enabled = true, Port = 10554]) in GetNetworkProtocolsResponse message at step 8.

The DUT did not send correct network protocols information (i.e. NetworkProtocols [Name = “RTSP”, Enabled = false, Port = integer]) in GetNetworkProtocolsResponse message at step 12.

6.2.23 SET NETWORK PROTOCOLS CONFIGURATION - UNSUPPORTED PROTOCOLS

Test Label: Device Management Network Command SetNetworkProtocols Test. (for unsupported protocols)

Test Case ID: DEVICE-2-1-35

ONVIF Core Specification Coverage: Set network protocols

Command Under Test: SetNetworkProtocols

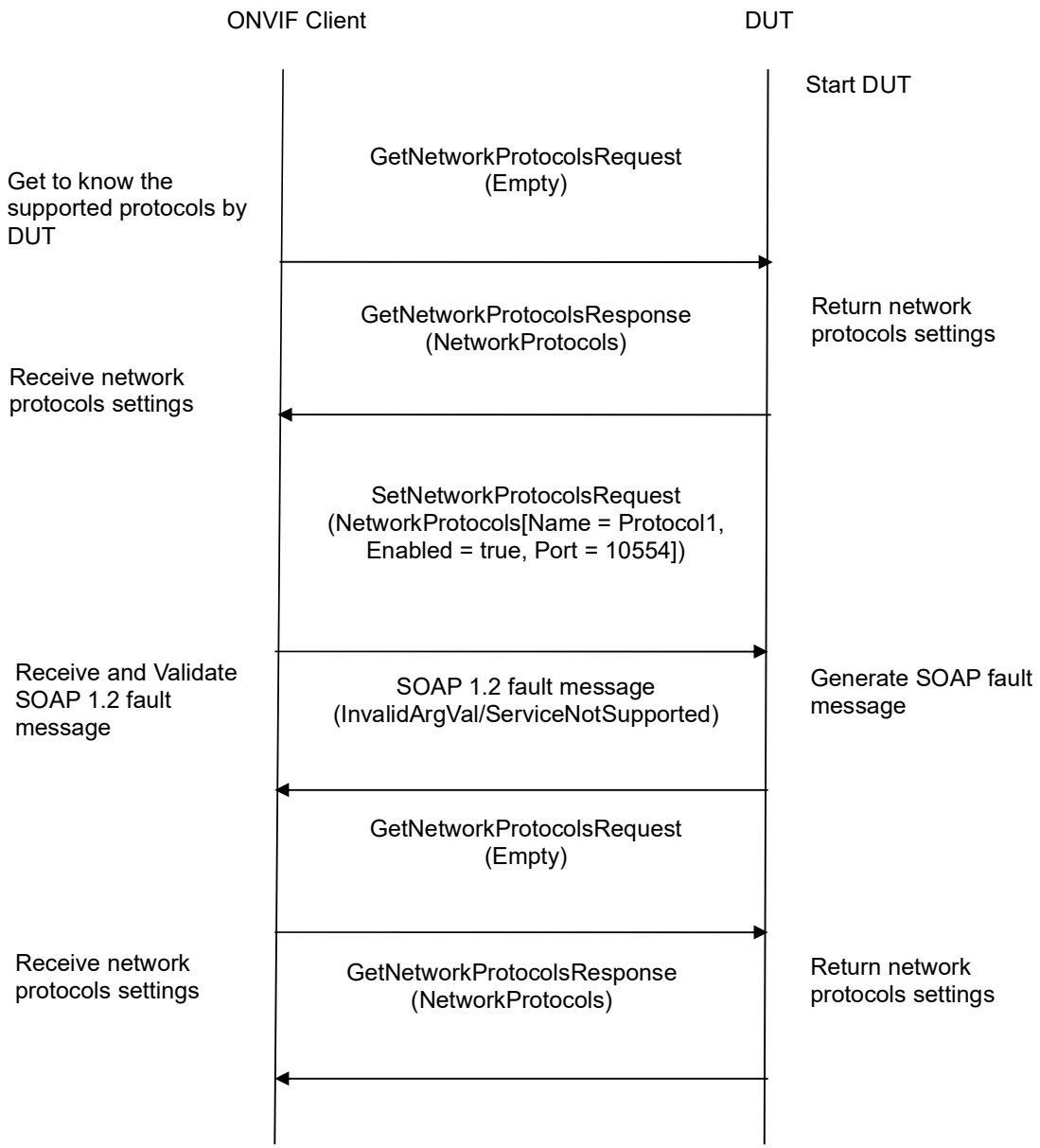
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify behavior of the DUT for unsupported protocols configuration.

Pre-Requisite: DUT does not support all protocols.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF client will retrieve network protocol configurations from the DUT through GetNetworkProtocolsRequest message.

4. The DUT sends network protocol configurations in the GetNetworkProtocolsResponse message (NetworkProtocols = supported protocols).
5. If HTTPS is not on the list of supported protocols, select HTTPS for the test as Protocol1. Otherwise, if RTSP is not on the list of supported protocols, select RTSP for the test as Protocol1. Otherwise, skip other steps and go to the next test.
6. ONVIF Client will invoke SetNetworkProtocolsRequest message (NetworkProtocols [Name = Protocol1, Enabled = true, Port = 10554]).
7. Verify that the DUT generates SOAP 1.2 fault message (InvalidArgVal/ServiceNotSupported).
8. Retrieve network protocol configurations from DUT through GetNetworkProtocolsRequest message.
9. The DUT sends valid network protocol configurations in the GetNetworkProtocolsResponse message (NetworkProtocols = supported protocols).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SOAP 1.2 fault message.

The DUT did not send correct fault code in the SOAP fault message (InvalidArgVal/ ServiceNotSupported).

The DUT did not send GetNetworkProtocolsResponse message.

The DUT returned unsupported protocols in GetNetworkProtocolsResponse message.

The DUT did not send correct network protocols information (i.e. NetworkProtocols = supported protocols) in GetNetworkProtocolsResponse message.

6.2.24 GET NETWORK DEFAULT GATEWAY CONFIGURATION

Test Label: Device Management Network Command GetNetworkDefaultGateway Test.

Test Case ID: DEVICE-2-1-25

ONVIF Core Specification Coverage: Get default gateway

Command Under Test: GetNetworkDefaultGateway

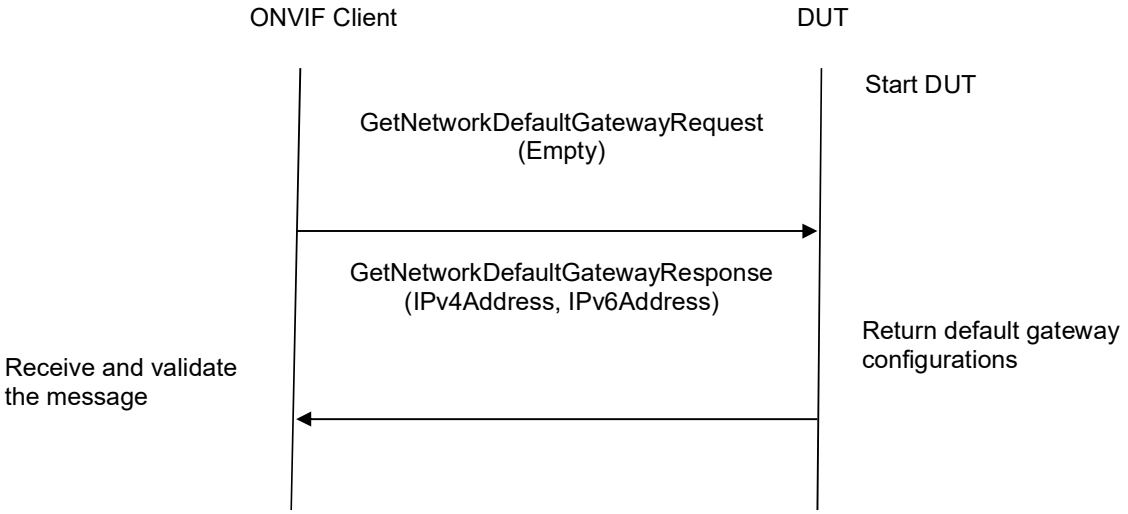
WSDL Reference: devicemgmt.wsdl

Test Purpose: To retrieve default gateway setting of DUT through GetNetworkDefaultGateway command.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetNetworkDefaultGatewayRequest message to retrieve default gateway settings of the DUT.
4. Verify the GetNetworkDefaultGatewayResponse from DUT (IPv4Address = list of IPv4 default gateway address, IPv6Address = list of IPv6 default gateway address).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetNetworkDefaultGatewayResponse message.

The DUT did not send correct default gateway information (i.e. IPv4Address = list of IPv4 default gateway address, IPv6Address = list of IPv6 default gateway address) in GetNetworkDefaultGatewayResponse message.

Note: See Annex A.10 for valid expression in terms of empty IP address.

6.2.25 SET NETWORK DEFAULT GATEWAY CONFIGURATION - INVALID IPV4

Test Label: Device Management Network Command SetNetworkDefaultGateway Test. (for invalid IPv4 address)

Test Case ID: DEVICE-2-1-28

ONVIF Core Specification Coverage: Set default gateway

Command Under Test: SetNetworkDefaultGateway

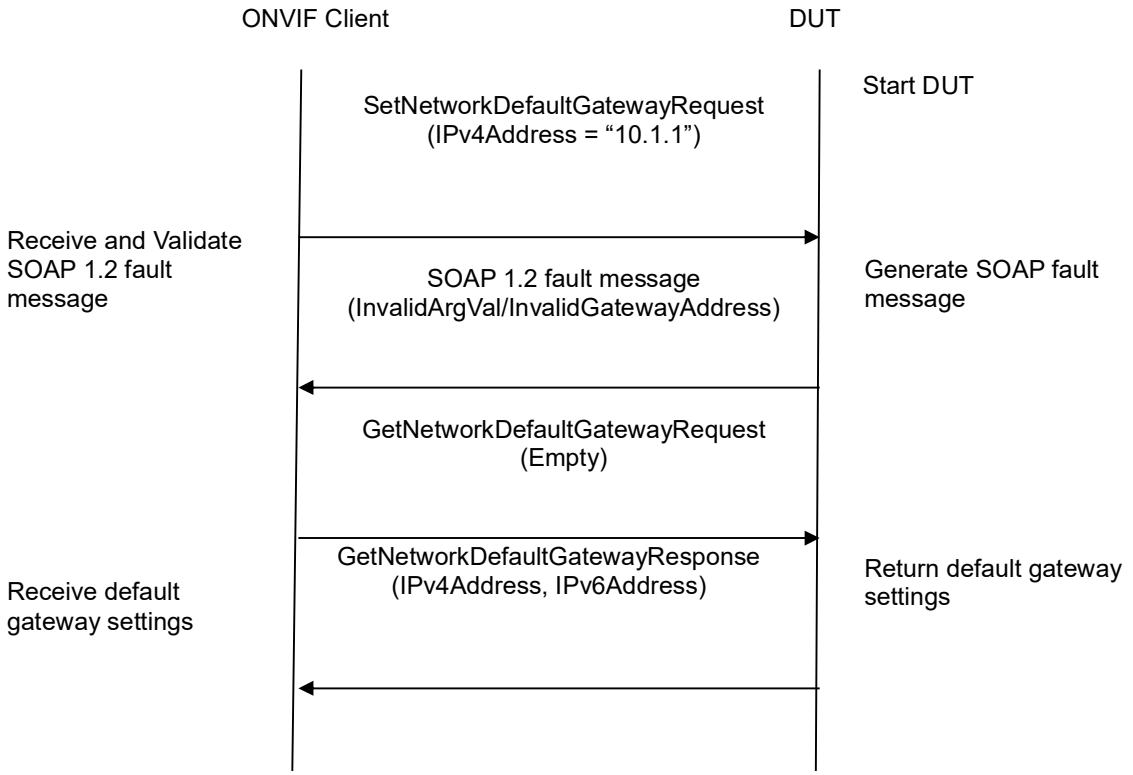
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify behavior of DUT for invalid default gateway IPv4 address configuration.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke SetNetworkDefaultGatewayRequest message (IPv4Address = “10.1.1”).
4. Verify that the DUT generates SOAP 1.2 fault message (InvalidArgVal/InvalidGatewayAddress).
5. Retrieve default gateway configurations from DUT through GetNetworkDefaultGatewayRequest message.
6. The DUT sends valid default gateway configurations in the GetNetworkDefaultGatewayResponse message (IPv4Address = list of IPv4 default gateway address, IPv6Address = list of IPv6 default gateway addresses).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SOAP 1.2 fault message.

The DUT did not send a correct fault code in the SOAP fault message (InvalidArgVal/InvalidGatewayAddress).

The DUT did not GetNetworkDefaultGatewayResponse message.

The DUT returned “10.1.1” as IPv4 default gateway address.

The DUT did not send correct default gateway information (i.e. IPv4Address = list of IPv4 default gateway address, IPv6Address = list of IPv6 default gateway addresses) in GetNetworkDefaultGatewayResponse message.

Note: See Annex A.6 for Invalid IPv4 Address and SOAP 1.2 fault message definitions.

6.2.26 SET NETWORK DEFAULT GATEWAY CONFIGURATION - INVALID IPV6

Test Label: Device Management Network Command SetNetworkDefaultGateway Test (for invalid IPv6 address).

Test Case ID: DEVICE-2-1-29

ONVIF Core Specification Coverage: Set default gateway

Command Under Test: SetNetworkDefaultGateway

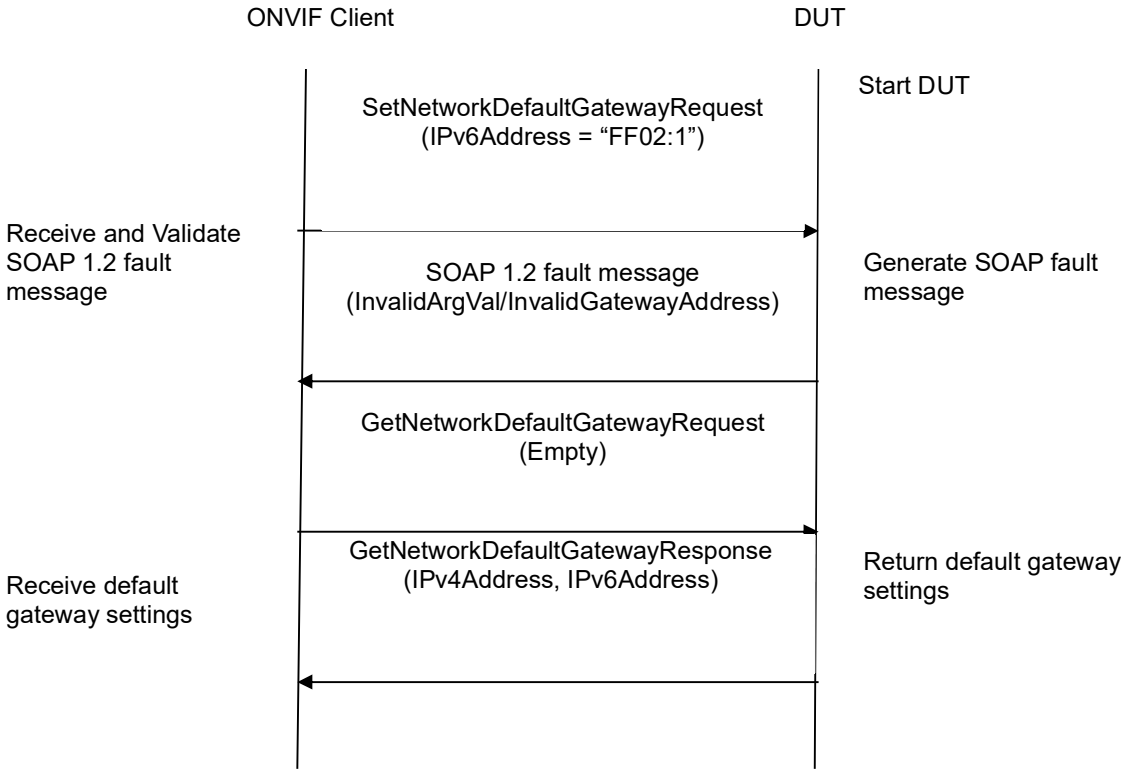
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify behavior of DUT for invalid default gateway IPv6 address configuration.

Pre-Requisite: IPv6 is implemented by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke SetNetworkDefaultGatewayRequest message (IPv6Address = “FF02:1”).
4. Verify that the DUT generates SOAP 1.2 fault message (InvalidArgVal/InvalidGatewayAddress).
5. Retrieve default gateway configurations from DUT through GetNetworkDefaultGatewayRequest message.
6. The DUT sends valid default gateway configurations in the GetNetworkDefaultGatewayResponse message (IPv4Address = list of IPv4 default gateway address, IPv6Address = list of IPv6 default gateway addresses).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SOAP 1.2 fault message.

The DUT did not send correct fault code in the SOAP fault message (InvalidArgVal/InvalidGatewayAddress).

The DUT did not GetNetworkDefaultGatewayResponse message.

The DUT returned “FF02:1” as IPv6 default gateway address.

The DUT did not send correct default gateway information (i.e. IPv4Address = list of IPv4 default gateway address, IPv6Address = list of IPv6 default gateway addresses) in GetNetworkDefaultGatewayResponse message.

6.2.27 SET NETWORK DEFAULT GATEWAY CONFIGURATION - IPV4

Test Label: Device Management Network Command SetNetworkDefaultGateway Test (for IPv4 address).

Test Case ID: DEVICE-2-1-30

ONVIF Core Specification Coverage: Set default gateway

Command Under Test: SetNetworkDefaultGateway

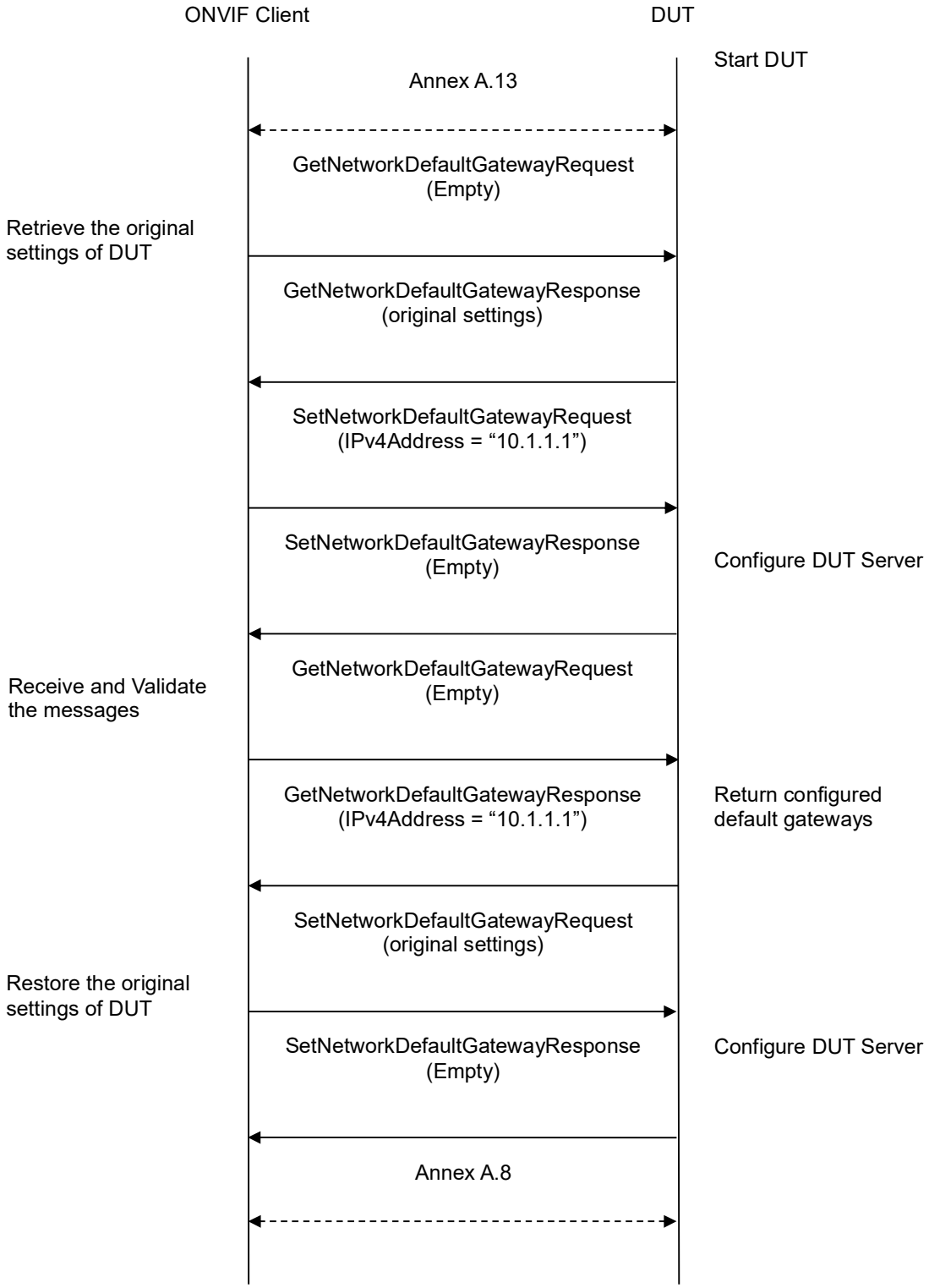
WSDL Reference: devicemgmt.wsdl

Test Purpose: To configure default gateway IPv4 address setting on a DUT using SetNetworkDefaultGateway command.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will follow the procedure described in Annex A.13 to turn off IPv4 DHCP and set manual IP configuration.
4. ONVIF Client will invoke GetNetworkDefaultGatewayRequest message to retrieve the original settings of DUT.
5. Verify that the DUT sends GetNetworkDefaultGatewayResponse (original settings).
6. ONVIF Client will invoke SetNetworkDefaultGatewayRequest message (IPv4Address = “10.1.1.1”).
7. Verify that the DUT sends SetNetworkDefaultGatewayResponse (empty message).
8. Verify the configured default gateways settings in DUT through GetNetworkDefaultGatewayRequest message.
9. DUT sends its configured default gateways settings in the GetNetworkDefaultGatewayResponse message (IPv4Address = “10.1.1.1”).
10. ONVIF Client will invoke SetNetworkDefaultGatewayRequest message to restore the original network default gateway settings of DUT.
11. ONVIF Client will follow the procedure described in Annex A.8 to turn on IPv4 DHCP to restore IP configuration.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetNetworkDefaultGatewayResponse message at step 7.

The DUT did not send GetNetworkDefaultGatewayResponse message at step 9.

The DUT did not send correct default gateway information (i.e. IPv4Address = “10.1.1.1”) in GetNetworkDefaultGatewayResponse message at step 9.

Note: See Annex A.6 for Valid IPv4 Address definition. See Annex A.10 for valid expression in terms of an empty IP address.

6.2.28 SET NETWORK DEFAULT GATEWAY CONFIGURATION - IPV6

Test Label: Device Management Network Command SetNetworkDefaultGateway Test (for IPv6 address).

Test Case ID: DEVICE-2-1-31

ONVIF Core Specification Coverage: Set default gateway

Command Under Test: SetNetworkDefaultGateway

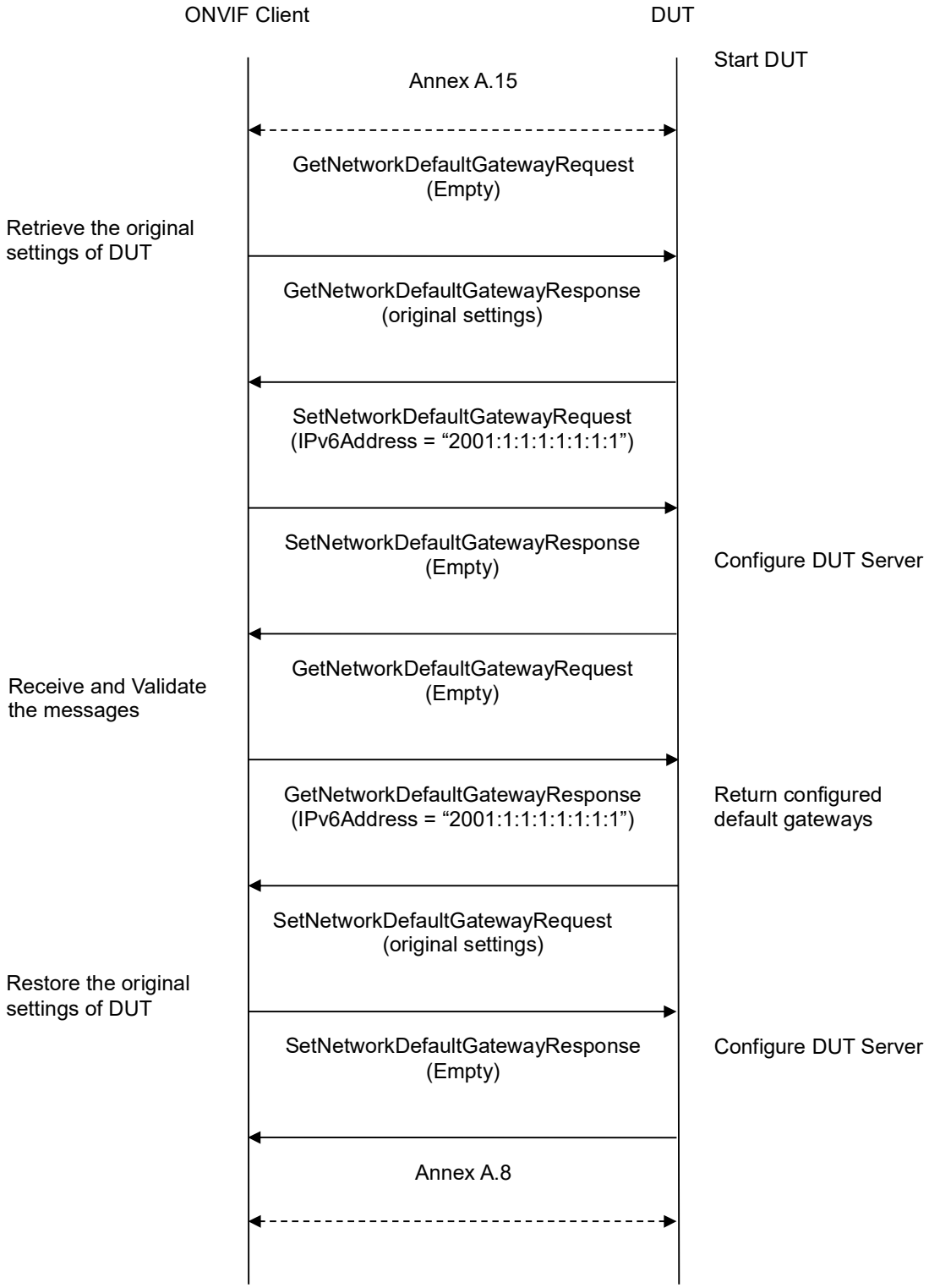
WSDL Reference: devicemgmt.wsdl

Test Purpose: To configure default gateway IPv6 address setting on a DUT using SetNetworkDefaultGateway command.

Pre-Requisite: IPv6 is implemented by the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start an ONVIF Client.
3. Start the DUT.
4. ONVIF Client will follow the procedure described in Annex A.15 to turn off IPv6 DHCP and set manual IP configuration.
5. ONVIF Client will invoke GetNetworkDefaultGatewayRequest message to retrieve the original settings of DUT.
6. Verify that the DUT sends GetNetworkDefaultGatewayResponse (original settings).
7. ONVIF Client will invoke SetNetworkDefaultGatewayRequest message (IPv6Address = “2001:1:1:1:1:1:1:1”).
8. Verify that the DUT sends SetNetworkDefaultGatewayResponse (empty message).
9. Verify the configured default gateways settings in DUT through GetNetworkDefaultGatewayRequest message.
10. The DUT sends its configured default gateways settings in the GetNetworkDefaultGatewayResponse message (IPv6Address = “2001:1:1:1:1:1:1:1”).
11. ONVIF Client will invoke SetNetworkDefaultGatewayRequest message to restore the original network default gateway settings of DUT.
12. ONVIF Client will follow the procedure described in Annex A.8 to turn on IPv6 DHCP to restore IP configuration.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetNetworkDefaultGatewayResponse message at step 7.

The DUT did not send GetNetworkDefaultGatewayResponse message at step 9.

The DUT did not send correct default gateway information (i.e. IPv6Address = “2001:1:1:1:1:1:1:1”) in GetNetworkDefaultGatewayResponse message at step 9.

6.2.29 NETWORK COMMAND SETHOSTNAME TEST

Test Label: Device Management Network Command SetHostname Test.

Test Case ID: DEVICE-2-1-32

ONVIF Core Specification Coverage: Set hostname

Command Under Test: SetHostname

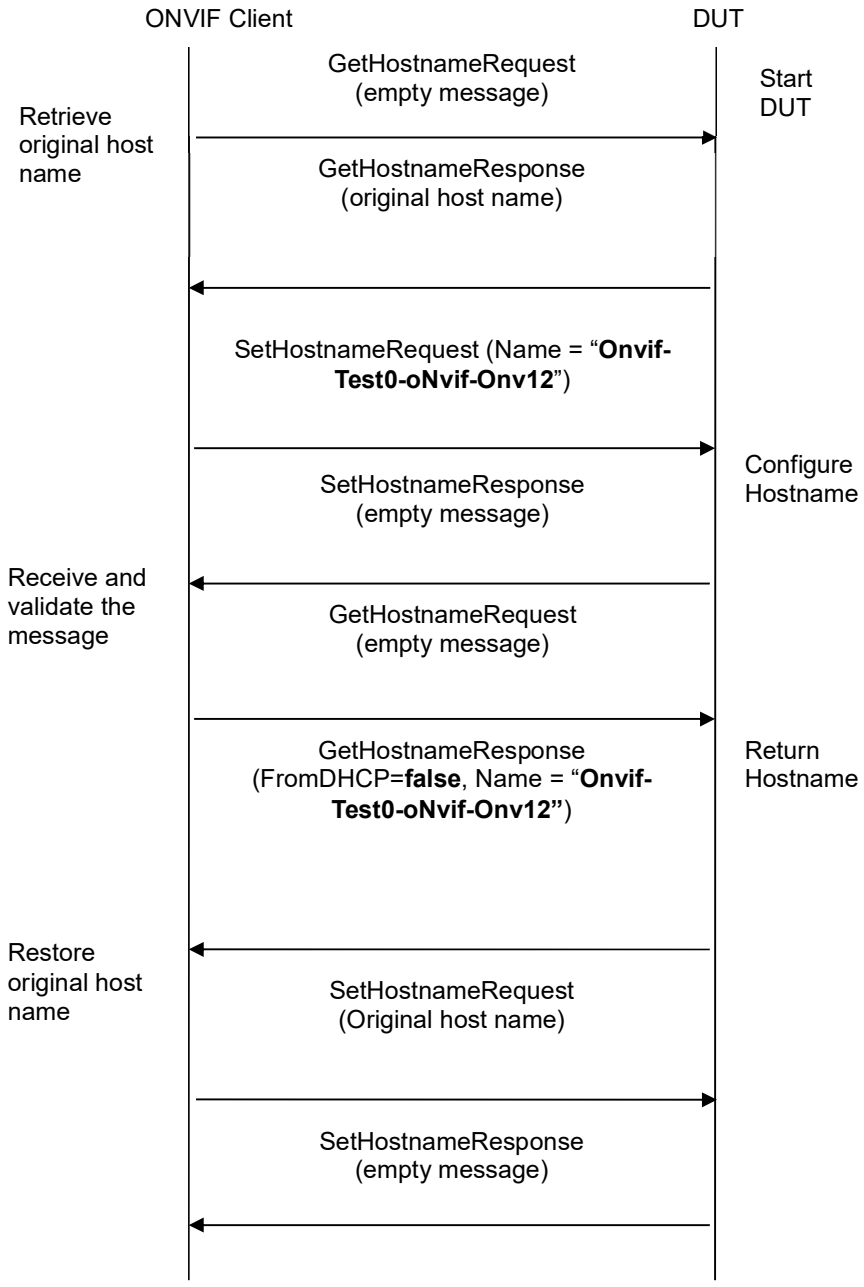
WSDL Reference: devicemgmt.wsdl

Test Purpose: To configure hostname on the DUT using SetHostname command.

Pre-Requisite: Testing environment (DHCP server) should not change the IP address of DUT during this test case execution.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetHostnameRequest to retrieve original settings of the DUT.
4. ONVIF Client will invoke SetHostnameRequest message (Name = “Onvif-Test0-oNvif-Onv12”, whose length is equal to 23 bytes) to configure the hostname.
5. Verify that the DUT sends SetHostnameResponse (empty message).
6. Verify the hostname configurations in DUT through GetHostnameRequest.
7. The DUT sends hostname configuration in the GetHostnameResponse message (FromDHCP = false, Name = “Onvif-Test0-oNvif-Onv12”).
8. ONVIF Client invokes SetHostnameRequest to restore original settings of the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetHostnameResponse message.

The DUT did not send GetHostnameResponse message.

The DUT did not send a correct hostname (i.e. “Onvif-Test0-oNvif-Onv123-Onvif12”) in the GetHostnameResponse message.

Note: See Annex A.2 for valid host names.

6.3 System

6.3.1 SYSTEM COMMAND GETSYSTEMDATEANDTIME

Test Label: Device Management System Command GetSystemDateAndTime Test.

Test Case ID: DEVICE-3-1-1

ONVIF Core Specification Coverage: Get system date and time

Command Under Test: GetSystemDateAndTime

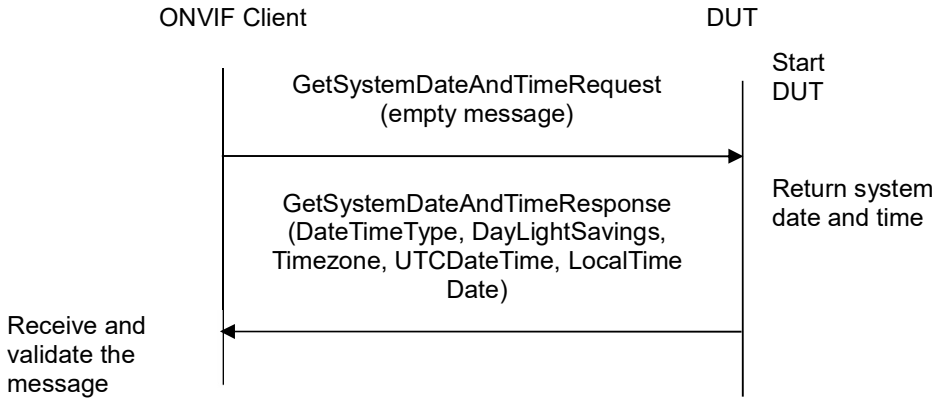
WSDL Reference: devicemgmt.wsdl

Test Purpose: To retrieve DUT system date and time using GetSystemDateAndTime command.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT.

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetSystemDateAndTimeRequest message to get DUT system date and time.
4. Verify system date and time configurations of DUT in GetSystemDateAndTimeResponse message (DateTimeType = Manual or NTP, DayLightSavings = true or false, TimeZone = POSIX 1003.1, UTC DateTime = Hour:Min:Sec, Year:Month:Day and LocalDateTime = Hour:Min:Sec, Year:Month:Day).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetSystemDateAndTimeResponse message.

The DUT did not send DateTimeType and DayLightSavings information in the GetSystemDateAndTimeResponse message.

Note: If system date and time are set manually, then DUT shall return UTCDateTime or LocalDateTime in the GetSystemDateAndTimeResponse message.

6.3.2 SYSTEM COMMAND SETSYSTEMDATEANDTIME

Test Label: Device Management Network Command SetSystemDateAndTime Test.

Test Case ID: DEVICE-3-1-2

ONVIF Core Specification Coverage: Set system date and time.

Command Under Test: SetSystemDateAndTime.

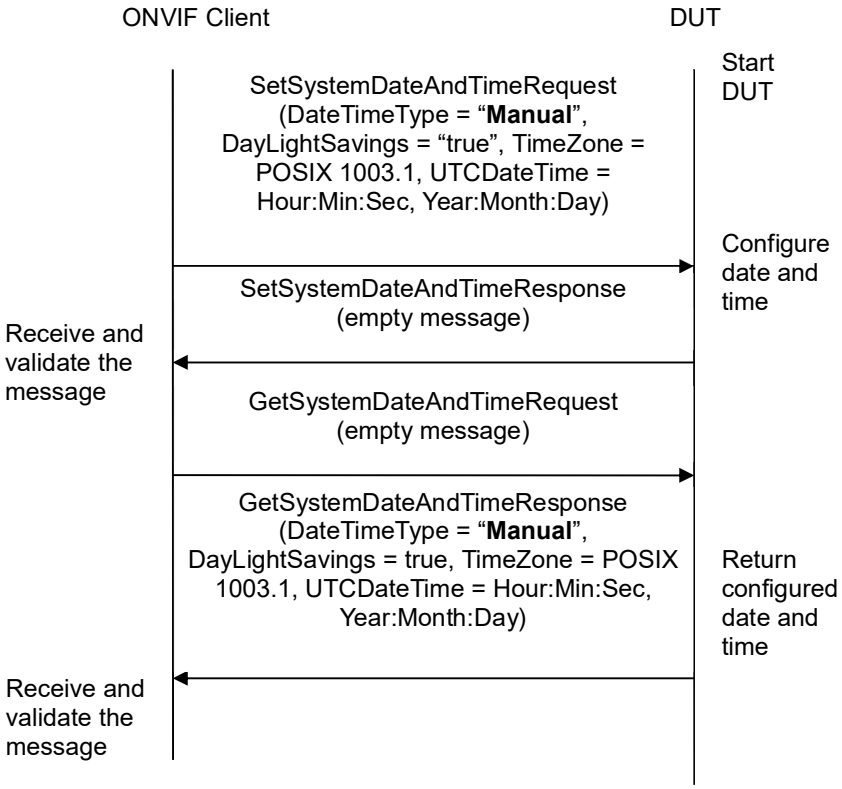
WSDL Reference: devicemgmt.wsdl

Test Purpose: To set the DUT system date and time using SetSystemDateAndTime command, date and time are entered manually.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke SetSystemDateAndTimeRequest message (DateTimeType = "Manual", DayLightSavings = true, TimeZone = POSIX 1003.1, UTCDateTime = Hour:Min:Sec, Year:Month:Day) to configure the date and time in the DUT.
4. Verify that DUT sends SetSystemDateAndTimeResponse message (empty message).
5. Verify the DUT date and time configurations through GetSystemDateAndTimeRequest message.

6. DUT sends system date and time configurations in the GetSystemDateAndTimeResponse message (DateTimeType = “Manual”, DayLightSavings = true, TimeZone = POSIX 1003.1, UTCDateTime = Hour:Min:Sec, Year:Month:Day).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetSystemDateAndTimeResponse message.

The DUT did not send GetSystemDateAndTimeResponse message.

The DUT did not send expected system date and time configuration (DateTimeType = “Manual”, DayLightSavings = true, TimeZone = POSIX 1003.1, UTCDateTime = Hour:Min:Sec, Year:Month:Day) in the GetSystemDateAndTimeResponse message.

6.3.3 SYSTEM COMMAND SETSYSTEMDATEANDTIME TEST FOR INVALID TIMEZONE

Test Label: Device Management Network Command SetSystemDateAndTime Test, for invalid time zone.

Test Case ID: DEVICE-3-1-4

ONVIF Core Specification Coverage: Set system date and time.

Command Under Test: SetSystemDateAndTime

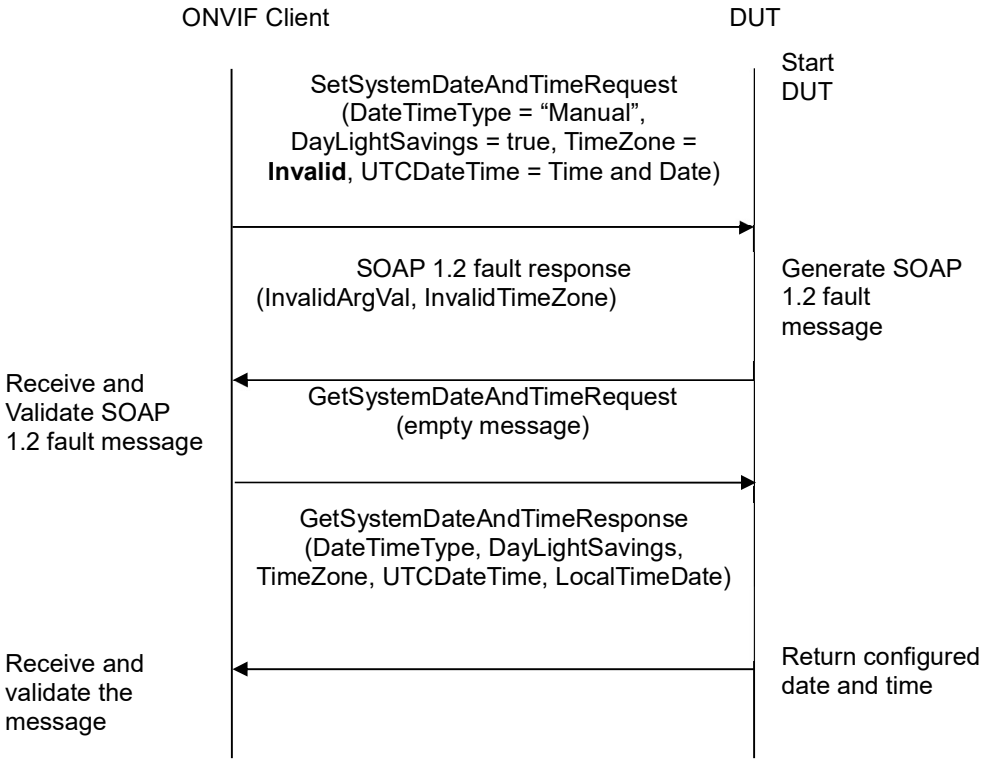
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify the behavior of the DUT for invalid TimeZone configuration.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT.

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke SetSystemDateAndTimeRequest message with invalid TimeZone (DateTimeType "Manual", DayLightSavings = "True", TimeZone = Invalid, UTCDateTime = Hour:Min:Sec, Year:Month:Day).
4. Verify that the DUT generates SOAP 1.2 fault response (InvalidArgVal, InvalidTimeZone).
5. Verify the DUT system date and time configurations through GetSystemDateAndTimeRequest message.
6. DUT sends system date and time configurations in the GetSystemDateAndTimeResponse message (DateTimeType = Manual or NTP, DayLightSavings = true or false, TimeZone = POSIX 1003.1, UTC DateTime = Hour:Min:Sec, Year:Month:Day and LocalDateTime = Hour:Min:Sec, Year:Month:Day).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SOAP 1.2 fault message.

The DUT did not send correct fault code in the SOAP fault message (InvalidArgVal, InvalidTimeZone).

The DUT did not send GetSystemDateAndTimeResponse message.

The DUT returned “Invalid TimeZone” in the GetSystemDateAndTimeResponse message.

Note: If system date and time are set manually, then DUT shall return UTCDateTime or LocalDateTime in the GetSystemDateAndTimeResponse message. See Annex A.3 for Invalid TimeZone and SOAP 1.2 fault message definitions.

6.3.4 SYSTEM COMMAND SETSYSTEMDATEANDTIME TEST FOR INVALID DATE

Test Label: Device Management Network Command SetSystemDateAndTime Test, for invalid date and time.

Test Case ID: DEVICE-3-1-5

ONVIF Core Specification Coverage: Set system date and time

Command Under Test: SetSystemDateAndTime

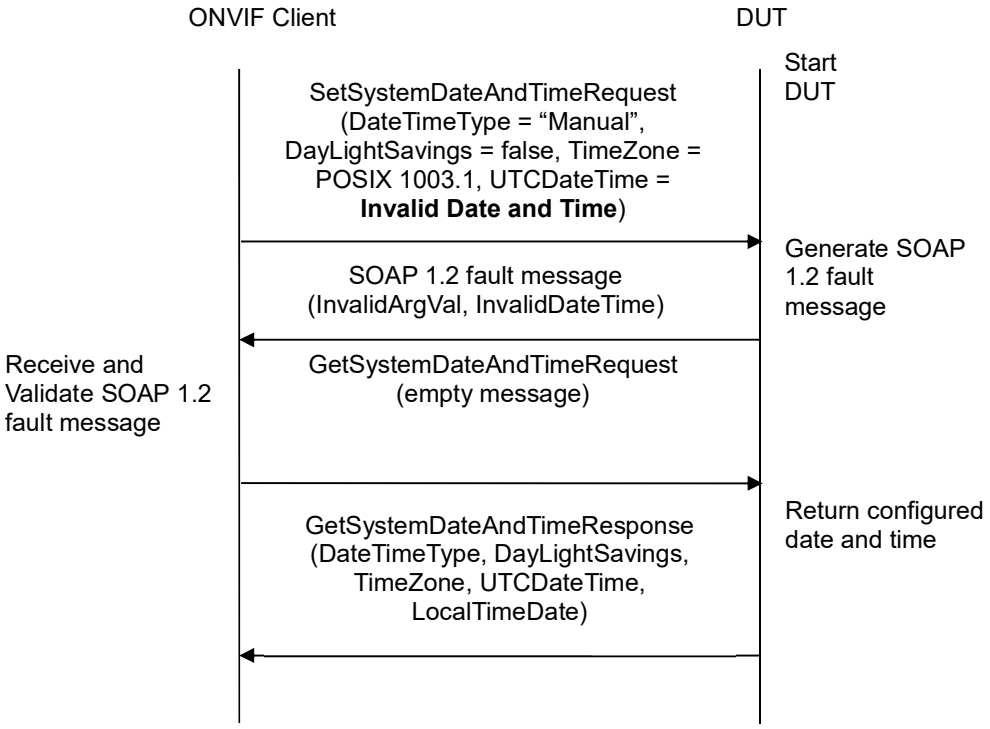
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify the behavior of the DUT for invalid system date and time configuration.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke SetSystemDateAndTimeRequest message with invalid Date and Time (DateTimeType = "Manual", DayLightSavings = false, TimeZone = POSIX 1003.1, UTCDateTime = Invalid Date and Time).
4. Verify that the DUT generates SOAP 1.2 fault message (InvalidArgVal, InvalidDateTime).
5. Verify the DUT system date and time configurations through GetSystemDateAndTimeRequest message.
6. The DUT sends system date and time configurations in the GetSystemDateAndTimeResponse message (DateTimeType = Manual or NTP, DayLightSavings = true or false, TimeZone = POSIX 1003.1, UTC DateTime = Hour:Min:Sec, Year:Month:Day and LocalDateTime = Hour:Min:Sec, Year:Month:Day).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SOAP 1.2 fault message.

The DUT did not send correct fault code in the SOAP fault message (InvalidArgVal, InvalidDateTime).

The DUT did not send GetSystemDateAndTimeResponse message.

The DUT did not send SetSystemDateAndTimeResponse message.

The DUT returned "Invalid Date and Time" in the GetSystemDateAndTimeResponse message.

Note: If system date and time are set manually, then the DUT shall return UTCDateTime or LocalDateTime in the GetSystemDateAndTimeResponse message.

See Annex A.4 for Invalid SOAP 1.2 fault message definition.

6.3.5 SYSTEM COMMAND FACTORY DEFAULT HARD

Test Label: Device Management System Command SetSystemFactoryDefault Test, Hard Reset.

Test Case ID: DEVICE-3-1-6

ONVIF Core Specification Coverage: Factory default

Command Under Test: SetSystemFactoryDefault

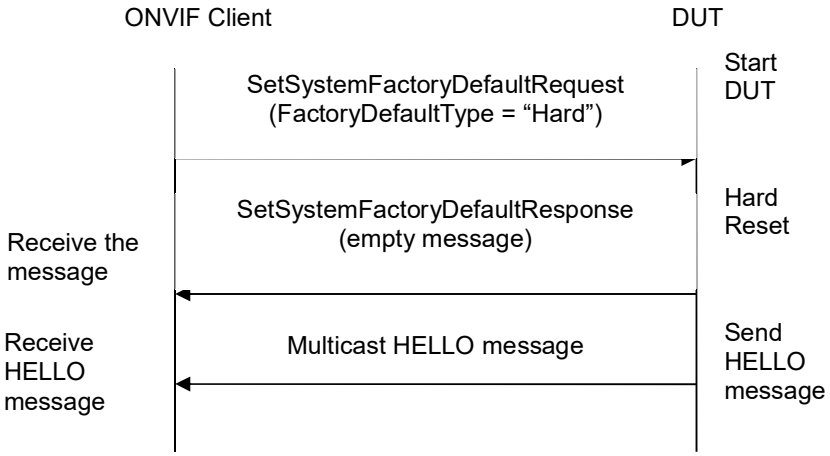
WSDL Reference: devicemgmt.wsdl

Test Purpose: To reload all parameters of the DUT to their default values using SetSystemFactoryDefault command. This test is for hard factory default.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT.

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client will invoke SetSystemFactoryDefaultRequest message (FactoryDefaultType = “Hard”).
4. Verify that DUT sends SetSystemFactoryDefaultResponse message.
5. Verify that DUT sends Multicast HELLO message after hard reset.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetSystemFactoryDefaultResponse message.

The DUT did not send HELLO message.

Note: After Hard Reset certain DUTs are not IP-reachable. In such situation DUT shall be configured with an IPv4 address, shall be IP reachable in the test network and other relevant configurations to be done for further tests.

6.3.6 SYSTEM COMMAND FACTORY DEFAULT SOFT

Test Label: Device Management System Command SetSystemFactoryDefault Test, Soft Reset.

Test Case ID: DEVICE-3-1-7

ONVIF Core Specification Coverage: Factory default

Command Under Test: SetSystemFactoryDefault

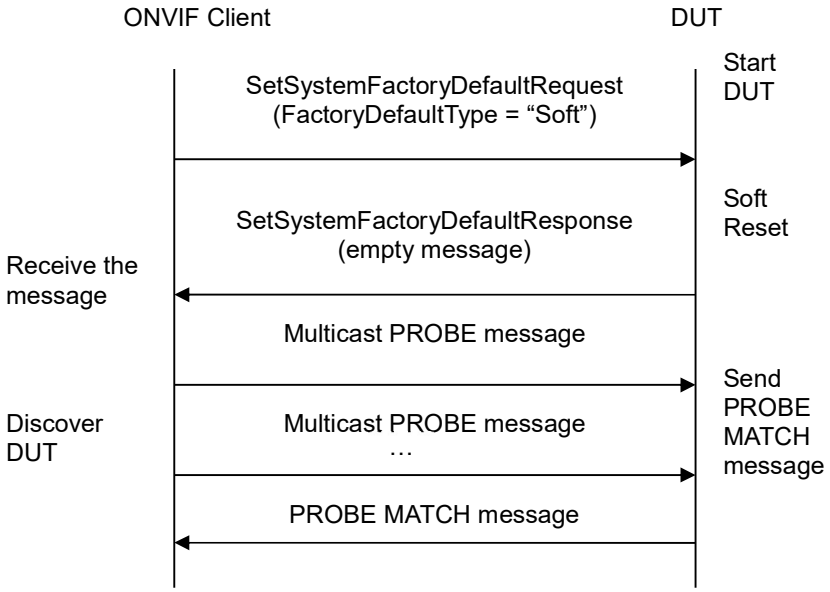
WSDL Reference: devicemgmt.wsdl

Test Purpose: To reload all parameters of the DUT to their default values using SetSystemFactoryDefault command. This test is for soft factory default.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT.

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke SetSystemFactoryDefaultRequest message (FactoryDefaultType = “Soft”).
4. Verify that the DUT sends SetSystemFactoryDefaultResponse message.
5. ONVIF Client will verify that DUT is accessible after soft reset. ONVIF Client will send Multicast PROBE message several times (i.e. 50 times at an interval of 5 seconds).
6. Verify that the DUT sends a PROBE MATCH message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetSystemFactoryDefaultResponse message.

The DUT did not send PROBE MATCH message (i.e. DUT cannot be discovered).

Note: After a soft reset some DUTs require some configurations to be done for further tests.

6.3.7 SYSTEM COMMAND REBOOT

Test Label: Device Management System Command SystemReboot Test.

Test Case ID: DEVICE-3-1-8

ONVIF Core Specification Coverage: Reboot

Command Under Test: SystemReboot

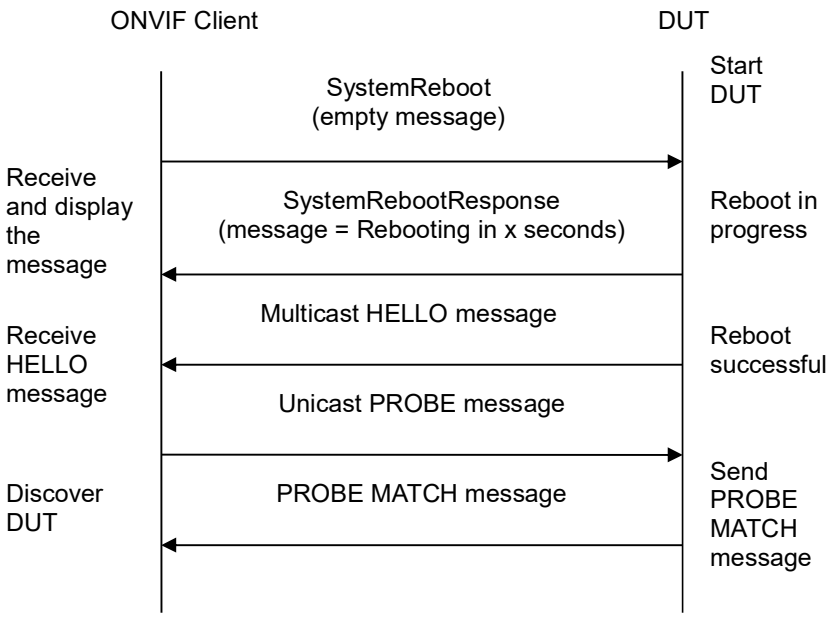
WSDL Reference: devicemgmt.wsdl

Test Purpose: To reboot the DUT through SystemReboot command.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke SystemReboot message to reset the DUT.
4. Verify that the DUT sends SystemRebootResponse message (example message string = “Rebooting in x seconds”).
5. The DUT will send Multicast HELLO message after it is successfully rebooted.
6. ONVIF Client will wait for Hello message sent from newly configured address by the DUT. Then ONVIF Client will verify the HELLO message and start using this newly configured address for further communications with DUT.
7. ONVIF Client will send Unicast PROBE message to discover the DUT.
8. The DUT will send a PROBE MATCH message.

9. ONVIF Client will verify the PROBE MATCH message sent by the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SystemRebootResponse message.

The DUT did not send HELLO message.

The DUT did not send PROBE MATCH message.

Note: If BYE message is supported by the DUT, then the DUT shall send multicast BYE message before the reboot.

6.3.8 SYSTEM COMMAND DEVICE INFORMATION

Test Label: Device Management System Command GetDeviceInformation Test.

Test Case ID: DEVICE-3-1-9

ONVIF Core Specification Coverage: Device Information

Command Under Test: GetDeviceInformation

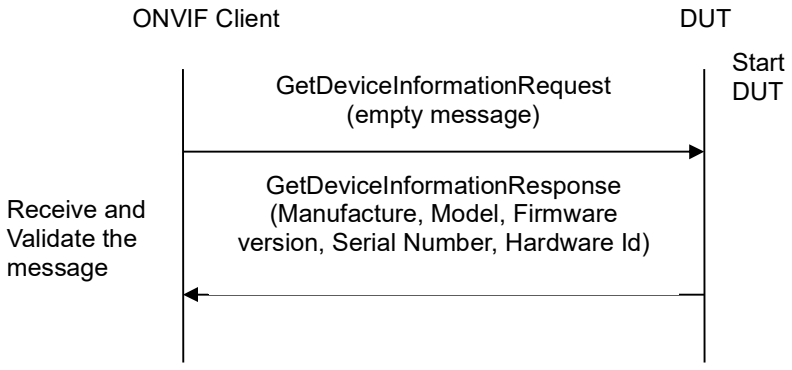
WSDL Reference: devicemgmt.wsdl

Test Purpose: To retrieve device information of the DUT using GetDeviceInformation command.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client will invoke GetDeviceInformationRequest message to retrieve device information such as manufacture, model and firmware version etc.
4. Verify the GetDeviceInformationResponse from the DUT (Manufacture, Model, Firmware version, Serial Number and Hardware Id).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetDeviceInformationResponse message.

The DUT did not send one or more mandatory information items in the GetDeviceInformationResponse message (mandatory information - Manufacturer, Model, Firmware Version, Serial Number and Hardware Id).

6.3.9 SYSTEM COMMAND GETSYSTEMLOG

Test Label: Device Management DUT System Command GetSystemLog Test.

Test Case ID: DEVICE-3-1-10

ONVIF Core Specification Coverage: Get system logs

Command under Test: GetSystemLog

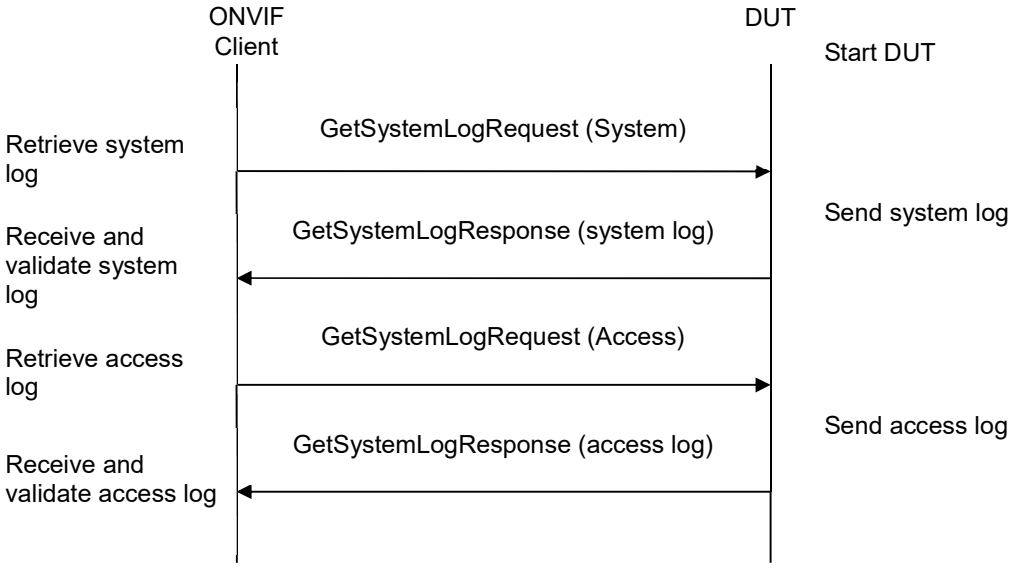
WSDL Reference: devicemgmt.wsdl

Test Propose: To retrieve the DUT system and access logs using GetSystemLog command.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetSystemLogRequest message (System) to retrieve system log from the DUT.
4. Verify the GetSystemLogResponse message from the DUT or SOAP 1.2 fault message (InvalidArgs/SystemlogUnavailable), if system log is unavailable.
5. ONVIF Client invokes GetSystemLogRequest message (Access) to retrieve system log from the DUT.
6. Verify the GetSystemLogResponse message from the DUT or SOAP 1.2 fault message (InvalidArgs/AccesslogUnavailable), if access log is unavailable.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetSystemLogResponse message or SOAP 1.2 fault message.

The DUT did not send valid GetSystemLogResponse message.

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

6.3.10 SYSTEM COMMAND SETSYSTEMDATEANDTIME

Test Label: Device Management Network Command SetSystemDateAndTime Test.

Test Case ID: DEVICE-3-1-11

ONVIF Core Specification Coverage: Set system date and time

Command Under Test: SetSystemDateAndTime

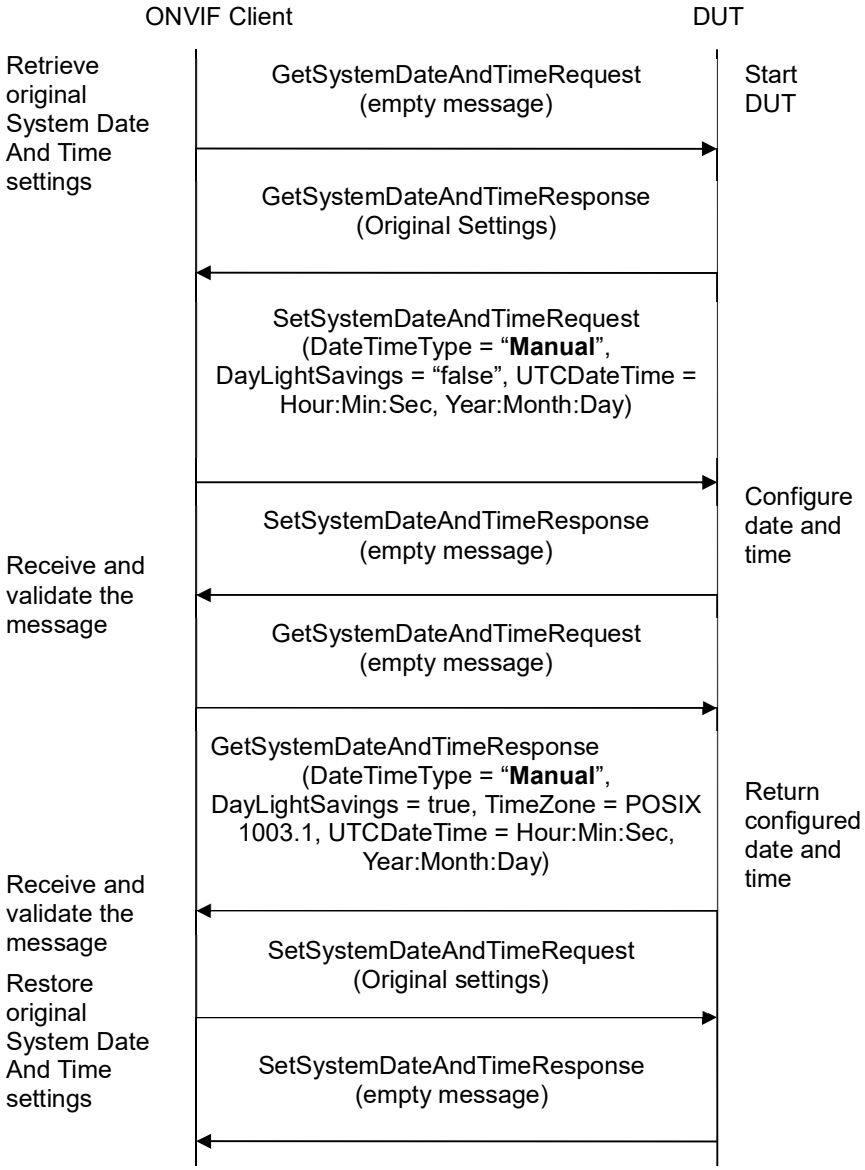
WSDL Reference: devicemgmt.wsdl

Test Purpose: To set the DUT system date and time using SetSystemDateAndTime command, date and time are entered manually.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetSystemDateAndTimeRequest to retrieve original system date and type settings of DUT.
4. ONVIF Client will invoke SetSystemDateAndTimeRequest message (DateTimeType = "Manual", DayLightSavings = false, UTCDateTime = Hour:Min:Sec, Year:Month:Day) to configure the date and time in the DUT.

5. Verify that DUT sends SetSystemDateAndTimeResponse message (empty message).
6. Verify the DUT date and time configurations through GetSystemDateAndTimeRequest message.
7. DUT sends system date and time configurations in the GetSystemDateAndTimeResponse message (DateTimeType = “Manual”, DayLightSavings = true, TimeZone = POSIX 1003.1, UTCDateTime = Hour:Min:Sec, Year:Month:Day).
8. ONVIF Client invokes SetSystemDateAndTimeRequest (original DateTimeType Settings, original DayLightSavings, original TimeZone, current UTCDateTime for synchronize time if DateTimeType=Manual) to restore original system date and type settings of DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetSystemDateAndTimeResponse message.

The DUT did not send GetSystemDateAndTimeResponse message.

The DUT did not send expected system date and time configuration (DateTimeType = “Manual”, DayLightSavings = true, TimeZone = POSIX 1003.1, UTCDateTime = Hour:Min:Sec, Year:Month:Day) in the GetSystemDateAndTimeResponse message.

6.3.11 SYSTEM COMMAND SETSYSTEMDATEANDTIME USING NTP

Test Label: Device Management Network Command SetSystemDateAndTime Test using NTP.

Test Case ID: DEVICE-3-1-12

ONVIF Core Specification Coverage: Set system date and time

Command Under Test: SetSystemDateAndTime

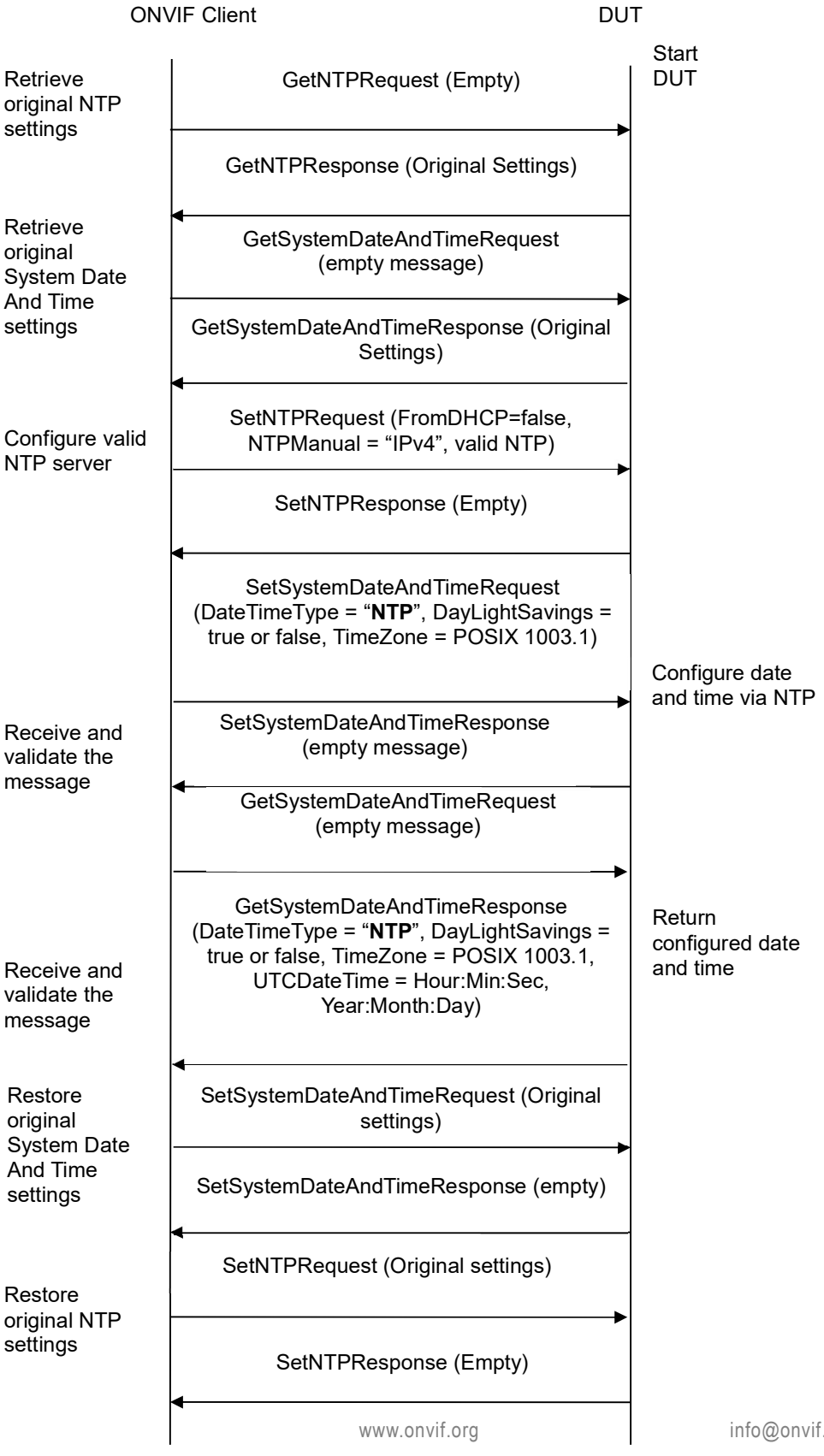
WSDL Reference: devicemgmt.wsdl

Test Purpose: To set the DUT system date and time using SetSystemDateAndTime command via NTP.

Pre-Requisite: A valid NTP server address should be configured in the DUT.

Test Configuration: ONVIF Client, DUT and NTP.

Test Sequence:


ONVIF Client
DUT
Start DUT
Retrieve original NTP settings
GetNTPRequest (Empty)
GetNTPResponse (Original Settings)
Retrieve original System Date And Time settings
GetSystemDateAndTimeRequest (empty message)
GetSystemDateAndTimeResponse (Original Settings)
Configure valid NTP server
SetNTPRequest (FromDHCP=false, NTPManual = "IPv4", valid NTP)
SetNTPResponse (Empty)
SetSystemDateAndTimeRequest (DateTimeType = "NTP", DayLightSavings = true or false, TimeZone = POSIX 1003.1)
Configure date and time via NTP
Receive and validate the message
SetSystemDateAndTimeResponse (empty message)
GetSystemDateAndTimeRequest (empty message)
GetSystemDateAndTimeResponse (DateTimeType = "NTP", DayLightSavings = true or false, TimeZone = POSIX 1003.1, UTCDateTime = Hour:Min:Sec, Year:Month:Day)
Return configured date and time
Receive and validate the message
Restore original System Date And Time settings
SetSystemDateAndTimeRequest (Original settings)
SetSystemDateAndTimeResponse (empty)
Restore original NTP settings
SetNTPRequest (Original settings)
SetNTPResponse (Empty)

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetNTPRequest to retrieve original NTP settings of the DUT.
4. ONVIF Client invokes GetSystemDateAndTimeRequest to retrieve original system date and type settings of DUT.
5. ONVIF Client invokes SetNTPRequest (FromDHCP=false, NTPManual = “IPv4”, valid NTP server address) to configure DUT with proper NTP server.
6. ONVIF Client will invoke SetSystemDateAndTimeRequest message (DateTimeType = “NTP”, DayLightSavings = true or false, TimeZone = POSIX 1003.1) to configure the time in the DUT.
7. The DUT shall obtain and configure time via NTP.
8. Verify that the DUT sends SetSystemDateAndTimeResponse (empty message).
9. Verify the DUT date and time configurations through GetSystemDateAndTimeRequest message.
10. The DUT sends system date and time configurations in the GetSystemDateAndTimeResponse message (DateTimeType = “NTP”, DayLightSavings = true or false, TimeZone = POSIX 1003.1, UTCDateTime = Hour:Min:Sec, Year:Month:Day).
11. ONVIF Client invokes SetSystemDateAndTimeRequest (original DateTimeType Settings, original DayLightSavings, original TimeZone, current UTCDateTime for synchronize time if DateTimeType=Manual) to restore original system date and type settings of DUT.
12. ONVIF Client invokes SetNTPRequest (original NTP Settings) to restore NTP settings of the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetSystemDateAndTimeResponse message.

The DUT did not send GetSystemDateAndTimeResponse message.

The DUT did not send expected system date and time configuration (DateTimeType = “NTP”, DayLightSavings = true or false, TimeZone = POSIX 1003.1, UTCDateTime = Hour:Min:Sec, Year:Month:Day) in the GetSystemDateAndTimeResponse message.

6.3.12 GET SYSTEM URIS

Test Label: Get System Uris Verification

Test Case ID: DEVICE-3-1-13

ONVIF Core Specification Coverage: Get System URIs (ONVIF Core Specification)

Command Under Test: GetSystemUris

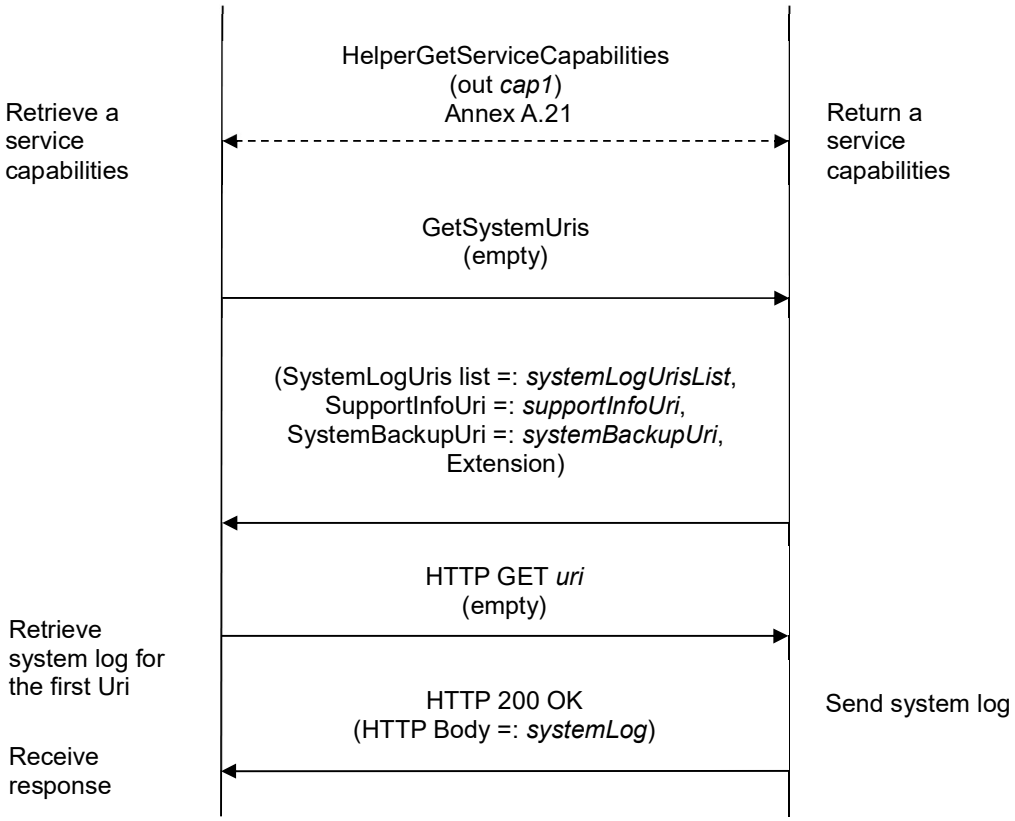
WSDL Reference: devicemgmt.wsdl

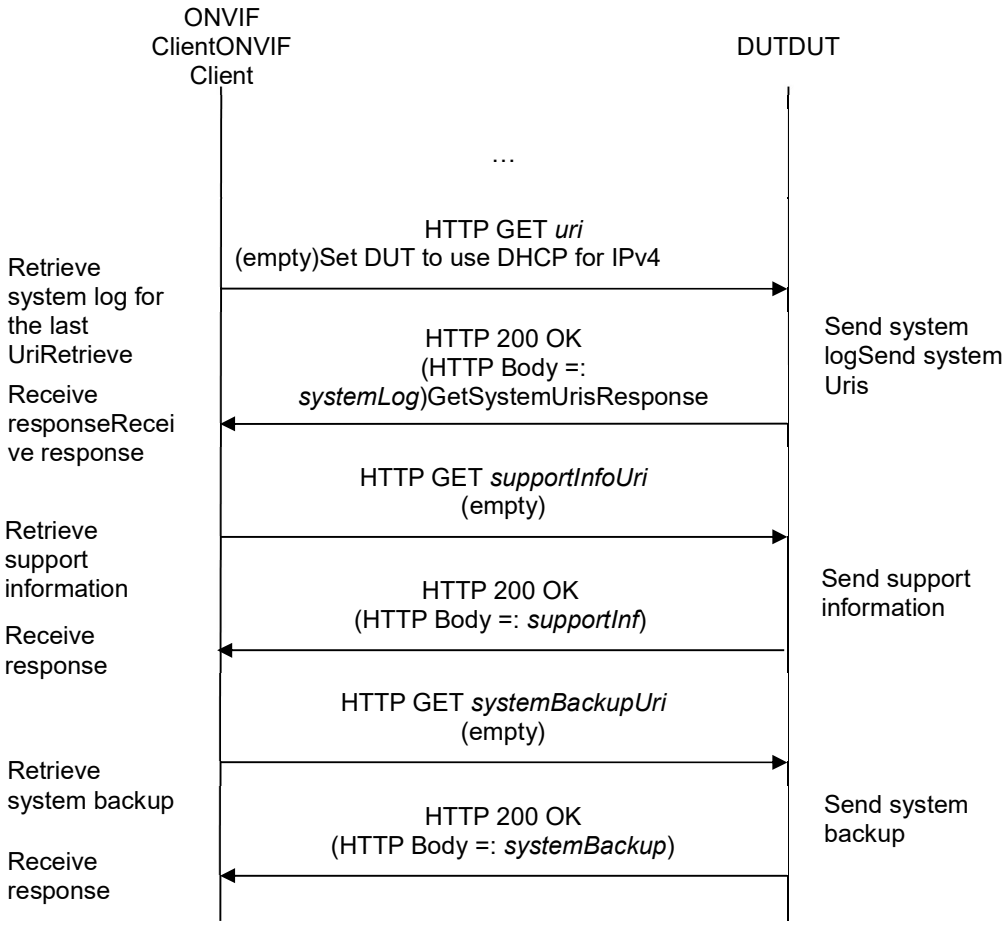
Test Purpose: To verify Get System Uris.

Pre-requisite: System backup and restore using HTTP GET and POST is supported by the DUT as indicated by the System.HttpSystemBackup capability or retrieval of system log using HTTP GET is supported by the DUT as indicated by the System.HttpSystemLogging capability or retrieval of support information using HTTP GET is supported by the DUT as indicated by the System.HttpSupportInformation capability.

Test Configuration: ONVIF Client and DUT

Test Sequence:





Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. If the DUT supports **GetServices** command:
   3.1. ONVIF Client gets the service capabilities (out *cap1*) by following the procedure mentioned in Annex A.21.
   3.2. Go to the step 5.
4. If the DUT does not support **GetServices** command:
   4.1. ONVIF Client gets the device service capabilities (out *cap2*) by following the procedure mentioned in Annex A.22.
5. ONVIF client invokes **GetSystemUris**.
6. The DUT responds with **GetSystemUrisResponse** message with parameters
   - SystemLogUris list =: *systemLogUrisList*

- SupportInfoUri =: *supportInfoUri*
- SystemBackupUri =: *systemBackupUri*
- Extension

7. If ((DUT supports **GetServices** command) and (*cap1*.System.HttpSystemLogging = true)) or ((DUT does not support GetServices command) and (*cap2*.System.Extension.HttpSystemLogging = true)):

   7.1. If *systemLogUrisList* is empty,FAIL the test and skip other steps.

   7.2. For each SystemLog.Uri *uri* from *systemLogUrisList* repeat the following steps:

      7.2.1. ONVIF client invokes **HTTP GET** to *uri*.

      7.2.2. The DUT responds with **HTTP 200 OK** response:

         - HTTP Body =: *systemLog*

      7.2.3. If *systemLog* is empty, FAIL the test and skip other steps.

8. If ((DUT supports GetServices command) and (*cap1*.System.HttpSupportInformation = true)) or ((DUT does not support GetServices command) and (*cap2*.System.Extension.HttpSupportInformation = true)):

   8.1. If *supportInfoUri* is empty,FAIL the test and skip other steps.

   8.2. ONVIF client invokes **HTTP GET** to *supportInfoUri*.

   8.3. The DUT responds with **HTTP 200 OK** response:

      - HTTP Body =: *supportInf*

   8.4. If *supportInf* is empty, FAIL the test and skip other steps.

9. If ((DUT supports GetServices command) and (*cap1*.System.HttpSystemBackup = true)) or ((DUT does not support GetServices command) and (*cap2*.System.Extension.HttpSystemBackup = true)):

   9.1. If *systemBackupUri* is empty,FAIL the test and skip other steps.

   9.2. ONVIF client invokes **HTTP GET** to *systemBackupUri*.

   9.3. The DUT responds with **HTTP 200 OK** response:

      - HTTP Body =: *systemBackup*

   9.4. If *systemBackup* is empty, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send **GetSystemUrisResponse** message.

The DUT did not send **HTTP 200 OK** response.

Note: HTTP GET requests could require authentication.

6.3.13 START SYSTEM RESTORE

Test Label: Start System Restore Verification

Test Case ID: DEVICE-3-1-14

ONVIF Core Specification Coverage: Start system restore (ONVIF Core Specification)

Command Under Test: StartSystemRestore

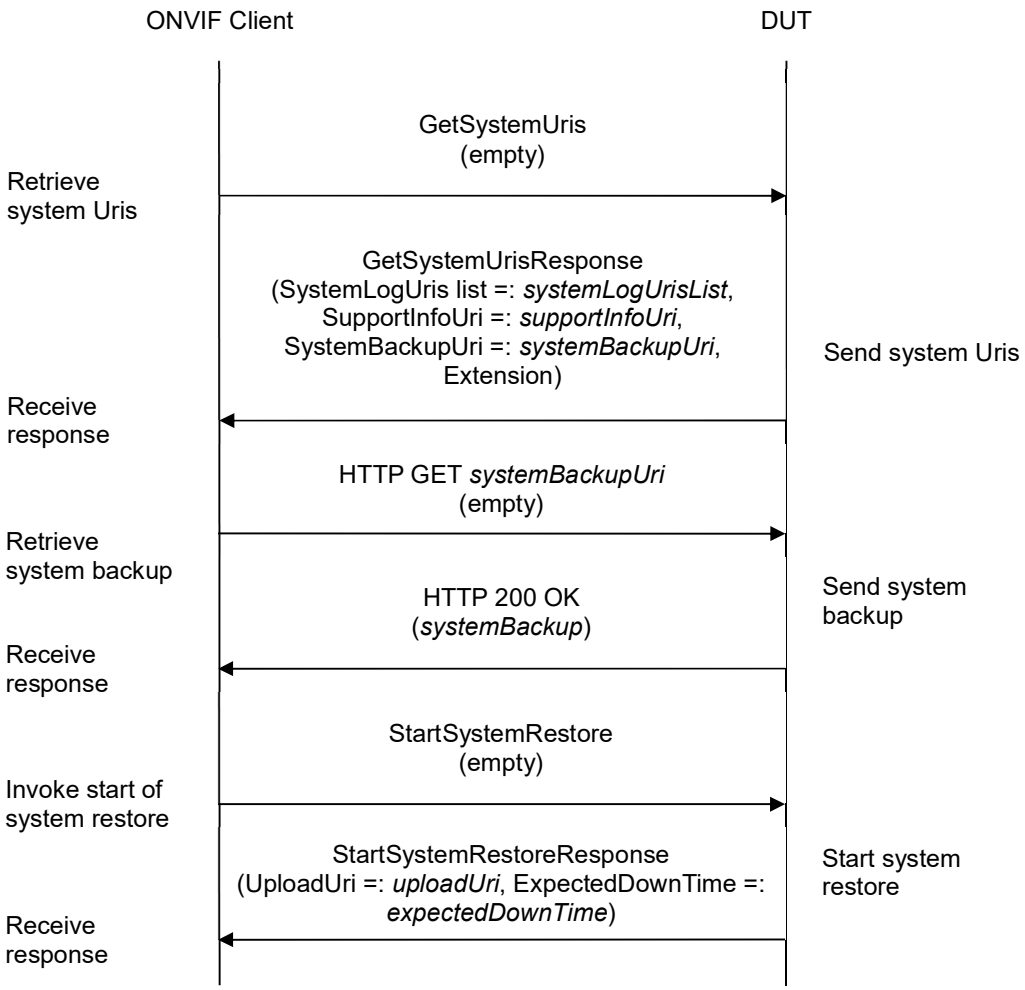
WSDL Reference: devicemgmt.wsdl

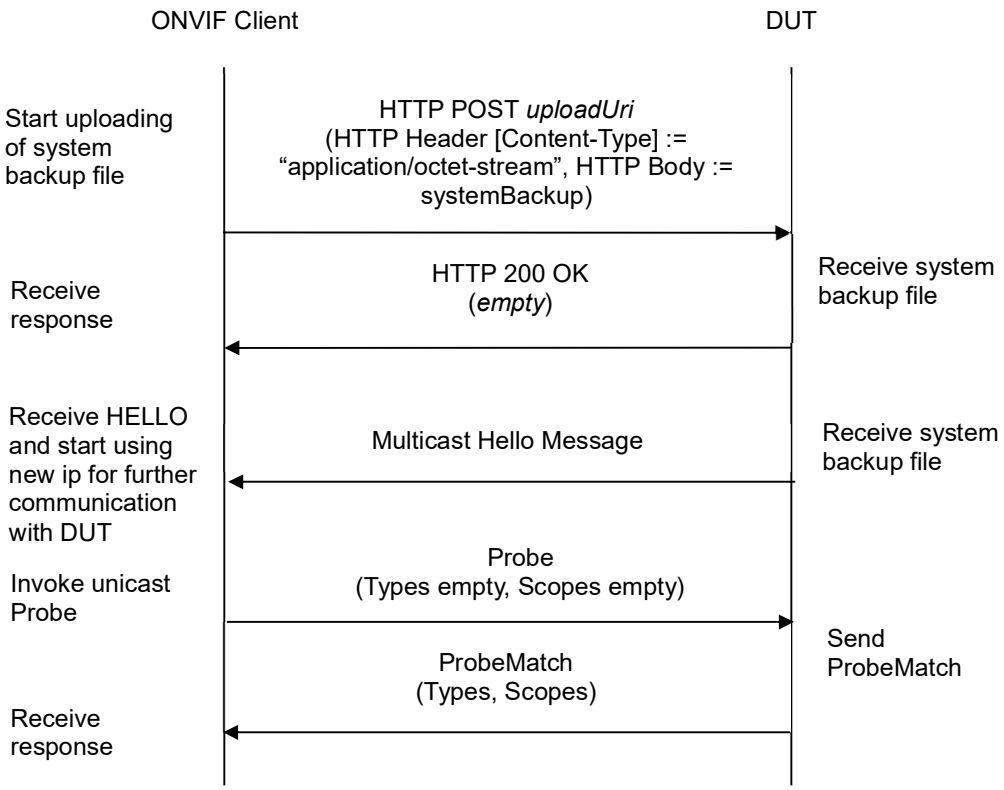
Test Purpose: To verify Start System Restore.

Pre-requisite: System backup and restore using HTTP GET and POST is supported by the DUT as indicated by the System.HttpSystemBackup capability.

Test Configuration: ONVIF Client and DUT

Test Sequence:





Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF client invokes **GetSystemUris**.
4. The DUT responds with **GetSystemUrisResponse** message with parameters
   - SystemLogUris list =: *systemLogUrisList*
   - SupportInfoUri =: *supportInfoUri*
   - SystemBackupUri =: *systemBackupUri*
   - Extension
5. If *systemBackupUri* is empty,FAIL the test and skip other steps.
6. ONVIF client invokes **HTTP GET** to *systemBackupUri*.
7. The DUT responds with **HTTP 200 OK** message with parameters
   - HTTP Body =: *systemBackup*

8. If *systemBackup* is empty, FAIL the test and skip other steps.
9. ONVIF client invokes **StartSystemRestore**.
10. The DUT responds with **StartSystemRestoreResponse** message with parameters
    - UploadUri =: *uploadUri*
    - ExpectedDownTime =: *expectedDownTime*
11. ONVIF client invokes **HTTP POST** to *uploadUri* with parameters
    - HTTP Header [Content-Type] := “application/octet-stream”
    - HTTP Body := *systemBackup*
12. The DUT responds with **HTTP 200 OK** message.
13. ONVIF Client waits *expectedDownTime+timeout1* for Hello message sent from newly configured address by the DUT. Then ONVIF Client starts using this newly configured address for further communications with DUT.
14. ONVIF Client invokes Unicast **Probe** message with the following parameters
    - Types empty
    - Scopes empty
15. The DUT responds with **ProbeMatch** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send **GetSystemUrisResponse** message.

The DUT did not send **StartSystemRestoreResponse** message.

The DUT did not send **HTTP 200 OK** response.

The DUT did not send **ProbeMatch** response.

Note: HTTP GET and **HTTP POST** requests could require authentication.

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

6.3.14 START SYSTEM RESTORE – INVALID BACKUP FILE

Test Label: Start System Restore Verification – Invalid Backup File

Test Case ID: DEVICE-3-1-15

ONVIF Core Specification Coverage: Start system restore (ONVIF Core Specification)

Command Under Test: StartSystemRestore

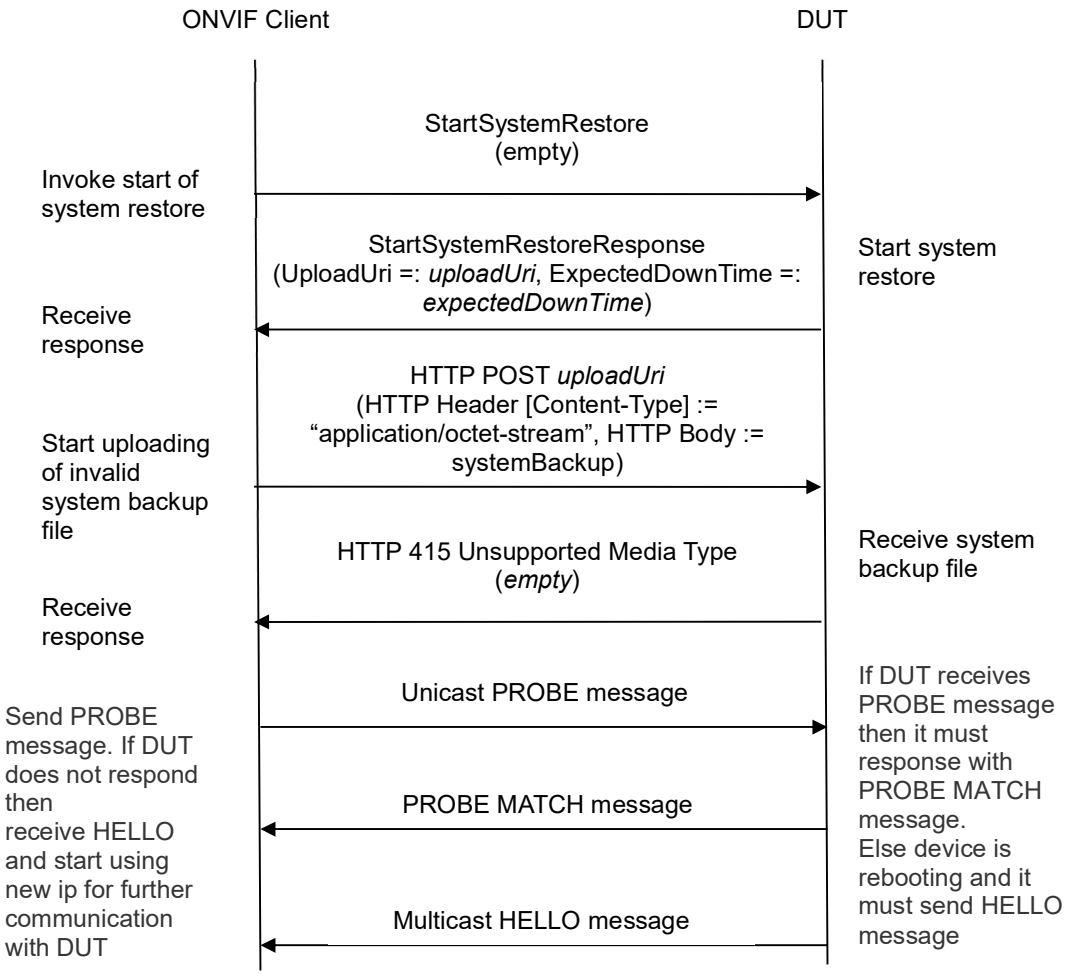
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify Start System Restore in the case if backup file is invalid.

Pre-requisite: System backup and restore using HTTP GET and POST is supported by the DUT as indicated by the System.HttpSystemBackup capability.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Set the following:
   - *systemBackup* := [random data]
4. ONVIF client invokes **StartSystemRestore**.
5. The DUT responds with **StartSystemRestoreResponse** message with parameters
   - UploadUri =: *uploadUri*
   - ExpectedDownTime =: *expectedDownTime*
6. ONVIF client invokes **HTTP POST** to *uploadUri* with parameters
   - HTTP Header [Content-Type] := “application/octet-stream”
   - HTTP Body := *systemBackup*
7. The DUT responds with **HTTP 415 Unsupported Media Type** message.
8. ONVIF client waits Reboot timeout.
9. ONVIF Client sends PROBE message and if DUT does not respond with PROBE MATCH mesage then go to the step 10, if DUT responds then finish the test.
10. ONVIF Client waits for Hello message sent from newly configured address by the DUT. Then ONVIF Client starts using this newly configured address for further communications with DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send **StartSystemRestoreResponse** message.

The DUT did not send **HTTP 415 Unsupported Media Type** response.

Note: HTTP POST request could require authentication.

6.4 Security

6.4.1 SECURITY COMMAND GETUSERS

Test Label: Device Management Security Command GetUsers Verification.

Test Case ID: DEVICE-4-1-1

ONVIF Core Specification Coverage: Get users.

Command Under Test: GetUsers.

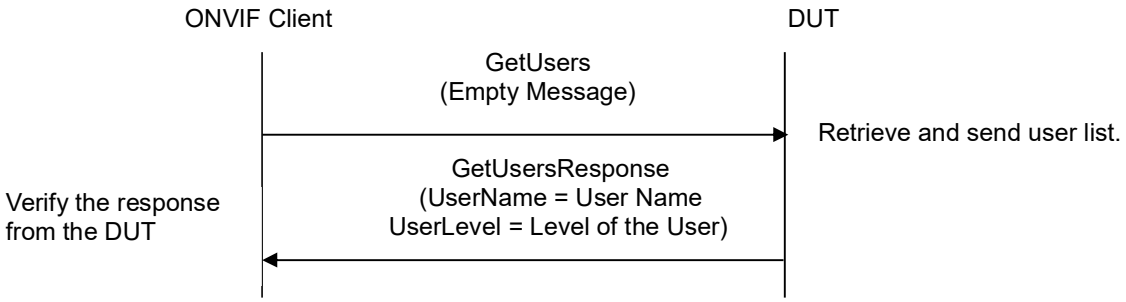
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify the behavior of GetUsers command.

Pre-Requisite: The ONVIF Client may need to operate in administrator mode to execute this test case.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetUsersRequest message (empty message), to retrieve the user list from the DUT.
4. Verify that DUT sends the GetUsersResponse message (Username, UserLevel).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetUsersResponse message.

Note: DUT may respond with none or more than one users.

6.4.2 SECURITY COMMAND CREATEUSERS ERROR CASE

Test Label: Device Management Security Command CreateUsers Verification, Creating Duplicate

Users.

Test Case ID: DEVICE-4-1-3

ONVIF Core Specification Coverage: Create Users.

Command Under Test: CreateUsers.

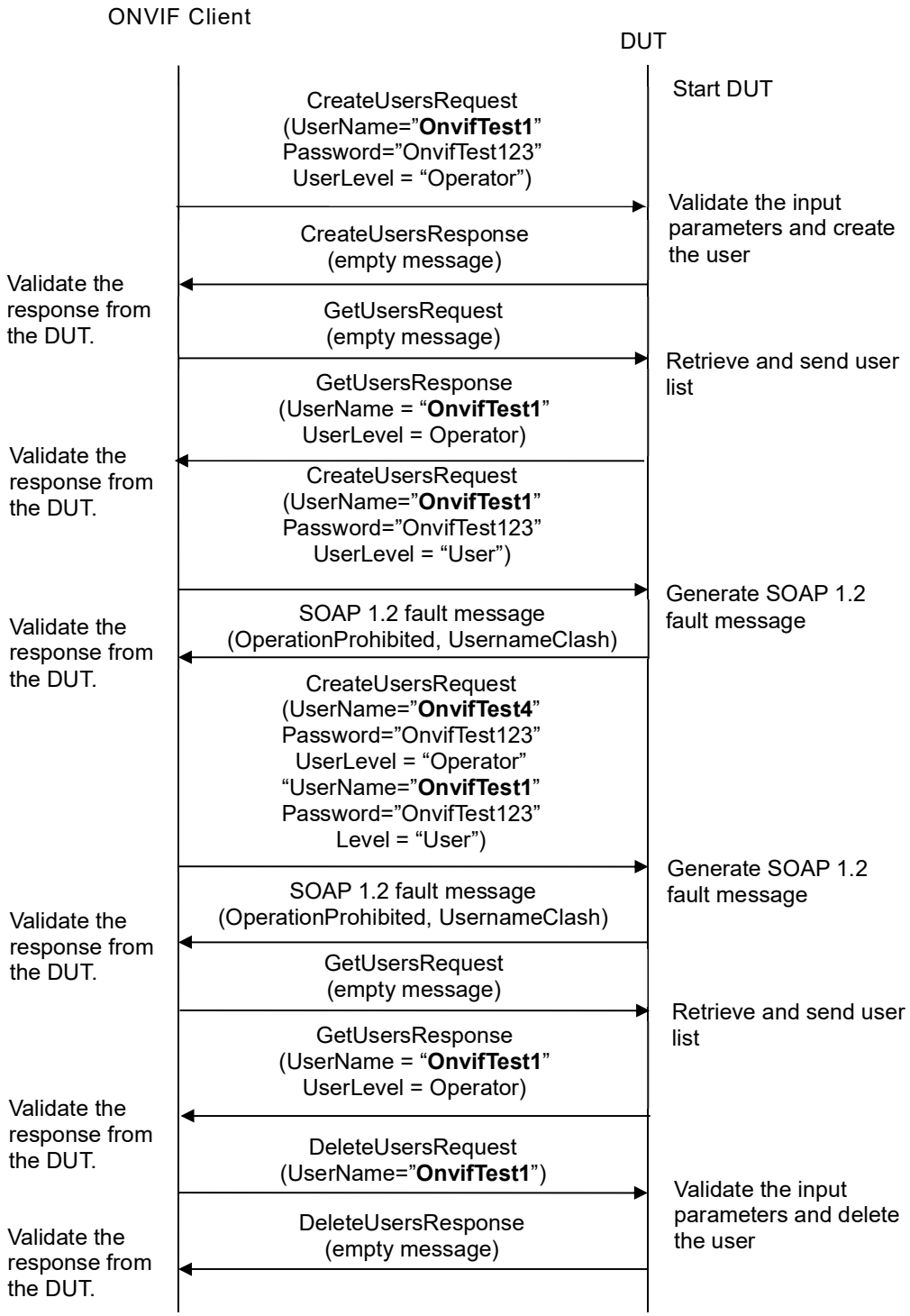
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify the behavior of CreateUsers command, if a user being created already exists.

Pre-Requisite: The ONVIF Client may need to operate in administrator mode to execute this test case.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

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

3. ONVIF Client will invoke CreateUsersRequest message (Username = ”OnvifTest1” Password = ”OnvifTest123” UserLevel = “Operator”), to create the user in the DUT.
4. Verify that the DUT sends CreateUsersResponse message (empty message).
5. ONVIF Client will invoke GetUsersRequest message (empty message), to retrieve the user list from the DUT.
6. Verify that the DUT sends GetUsersResponse message (Username = “OnvifTest1” UserLevel = Operator)
7. ONVIF Client will invoke CreateUsersRequest message (Username = ”OnvifTest1” Password = OnvifTest123 UserLevel = User) to create the user in the DUT.
8. Verify that the DUT generates SOAP 1.2 fault message (OperationProhibited, UsernameClash).
9. ONVIF Client will invoke CreateUsersRequest message (Username = “OnvifTest4” Password = “OnvifTest123” UserLevel = “Operator”, Username = “OnvifTest1” Password = “OnvifTest123” UserLevel = “User”) to create the users in the DUT.
10. Verify that the DUT generates SOAP 1.2 fault message (OperationProhibited, UsernameClash).
11. ONVIF Client will invoke GetUsersRequest message (empty message), to retrieve the user list from the DUT.
12. Verify that DUT sends GetUsersResponse message (Username = “OnvifTest1” UserLevel = Operator)
13. ONVIF Client will invoke DeleteUsersRequest message (Username = ”OnvifTest1”) to delete the user in the DUT.
14. Verify that the DUT sends DeleteUsersResponse message (empty message).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SOAP 1.2 fault message.

The DUT did not send correct SOAP 1.2 fault message (OperationProhibited, UsernameClash).

The DUT did not send GetUsers response message.

The DUT did not send CreateUsers response message.

The DUT did not send DeleteUsers response message.

The DUT creates the user “OnvifTest1” with Level = “User”.

The DUT creates the user “OnvifTest4”.

Note:

The DUT may return SOAP Fault 1.2 “TooManyUsers” for the CreateUsers command. Such SOAP 1.2 fault message shall be treated as PASS case for this test.

The DUT may return a greater number of users than actually created in this test case i.e. default users if any might be returned.

6.4.3 SECURITY COMMAND DELETEUSERS

Test Label: Device Management Security Command DeleteUsers Verification.

Test Case ID: DEVICE-4-1-4

ONVIF Core Specification Coverage: Delete users.

Command Under Test: DeleteUsers

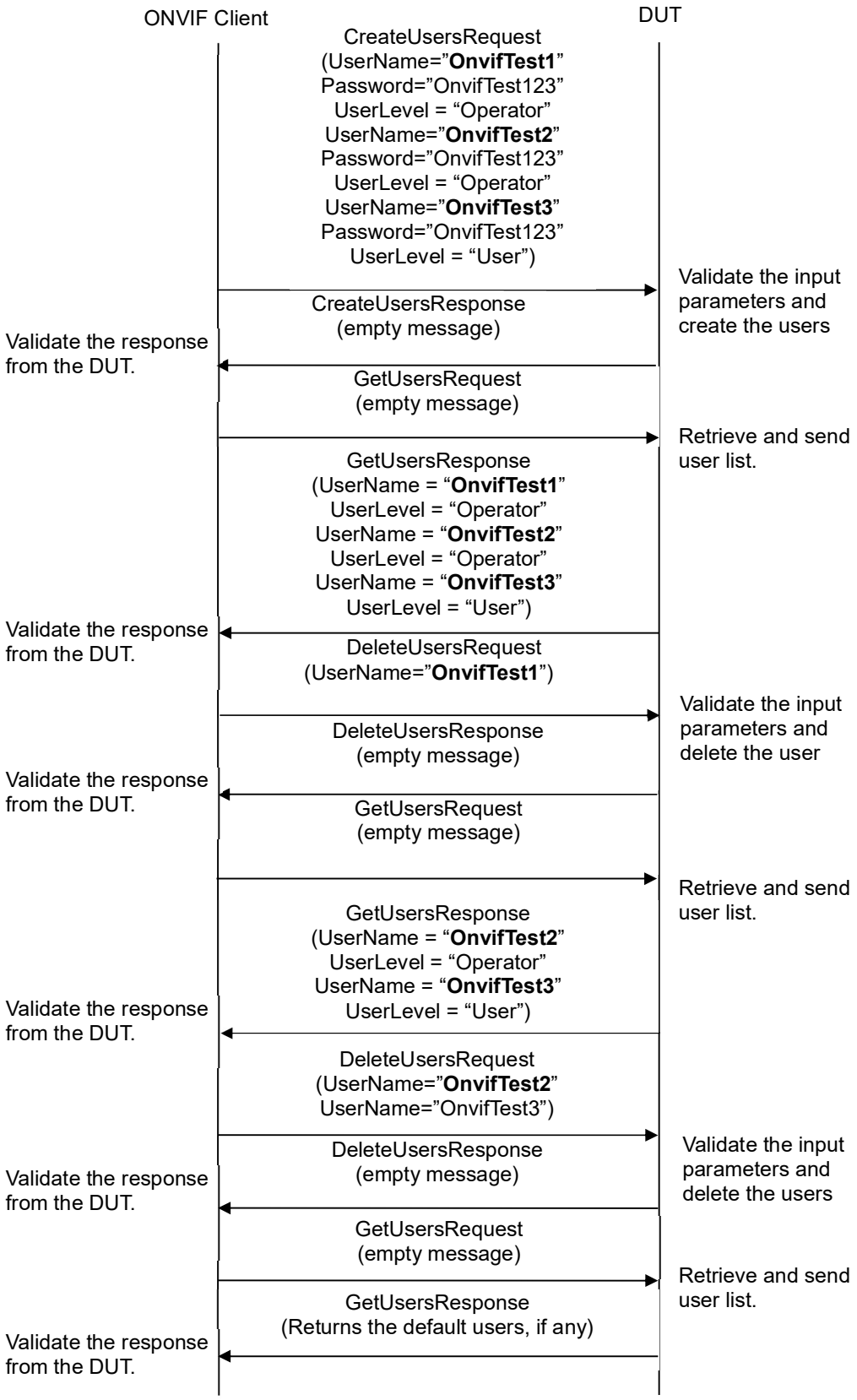
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify the behavior of DeleteUsers command.

Pre-Requisite: ONVIF Client may need to operate in administrator mode to execute this test case.

Test Configuration: ONVIF Client and DUT.

Test Sequence:


ONVIF Client
DUT
CreateUsersRequest
(UserName="OnvifTest1"
Password="OnvifTest123"
UserLevel = "Operator"
UserName="OnvifTest2"
Password="OnvifTest123"
UserLevel = "Operator"
UserName="OnvifTest3"
Password="OnvifTest123"
UserLevel = "User")
Validate the input parameters and create the users
CreateUsersResponse
(empty message)
Validate the response from the DUT.
GetUsersRequest
(empty message)
Retrieve and send user list.
GetUsersResponse
(UserName = "OnvifTest1"
UserLevel = "Operator"
UserName = "OnvifTest2"
UserLevel = "Operator"
UserName = "OnvifTest3"
UserLevel = "User")
Validate the response from the DUT.
DeleteUsersRequest
(UserName="OnvifTest1")
Validate the input parameters and delete the user
DeleteUsersResponse
(empty message)
Validate the response from the DUT.
GetUsersRequest
(empty message)
Retrieve and send user list.
GetUsersResponse
(UserName = "OnvifTest2"
UserLevel = "Operator"
UserName = "OnvifTest3"
UserLevel = "User")
Validate the response from the DUT.
DeleteUsersRequest
(UserName="OnvifTest2"
UserName="OnvifTest3")
Validate the input parameters and delete the users
DeleteUsersResponse
(empty message)
Validate the response from the DUT.
GetUsersRequest
(empty message)
Retrieve and send user list.
GetUsersResponse
(Returns the default users, if any)
Validate the response from the DUT.

Test Procedure:

1. Start an ONVIF Client
2. Start the DUT.
3. ONVIF Client will invoke CreateUsersRequest message (Username = ”OnvifTest1” Password = ”OnvifTest123” UserLevel = “Operator”, Username = ”OnvifTest2” Password = ”OnvifTest123” UserLevel = “Operator”, Username = ”OnvifTest3” Password = ”OnvifTest123” UserLevel = “User”) to create the user in the DUT.
4. Verify that DUT sends CreateUsersResponse message (empty message).
5. ONVIF Client will invoke GetUsersRequest message (empty message), to retrieve the user list from the DUT.
6. Verify that DUT sends GetUsersResponse message (Username = “OnvifTest1” UserLevel = “Operator”, Username = “OnvifTest2” UserLevel = “Operator”, Username = “OnvifTest3” UserLevel = “User”).
7. ONVIF Client will invoke DeleteUsersRequest message (Username = ”OnvifTest1”), to delete the user.
8. Verify that DUT sends DeleteUsersResponse message (empty message).
9. ONVIF Client will invoke GetUsersRequest message (empty message), to retrieve the user list from the DUT.
10. Verify that DUT sends the updated user list (Username = “OnvifTest2” UserLevel = “Operator”, Username = “OnvifTest3” UserLevel = “User”).
11. ONVIF Client will invoke DeleteUsersRequest message (Username = “OnvifTest2”, Username = “OnvifTest3”) to delete the users.
12. Verify that DUT sends the DeleteUsersResponse message (empty message).
13. ONVIF Client will invoke GetUsersRequest message (empty message), to retrieve the user list from the DUT.
14. Verify that DUT sends the GetUsersResponse message (Returns the default users, if any).

Test Result:

PASS –

The DUT passed all assertions

FAIL –

The DUT did not send DeleteUsers response message.

The DUT did not create the users at step 3.

The DUT did not send GetUsers response message.

The DUT did not send CreateUsers response message.

The DUT did not delete the users.

Note:

The DUT may return SOAP Fault 1.2 “TooManyUsers” for the CreateUsers command. Such SOAP 1.2 fault message shall be treated as PASS case for this test.

The DUT may return a greater number of users than actually created in this test case i.e. default users if any might be returned.

6.4.4 SECURITY COMMAND DELETEUSERS ERROR CASE

Test Label: Device Management Security Command DeleteUsers Verification, Deleting Non Existing Users.

Test Case ID: DEVICE-4-1-5

ONVIF Core Specification Coverage: Delete users.

Command Under Test: DeleteUsers.

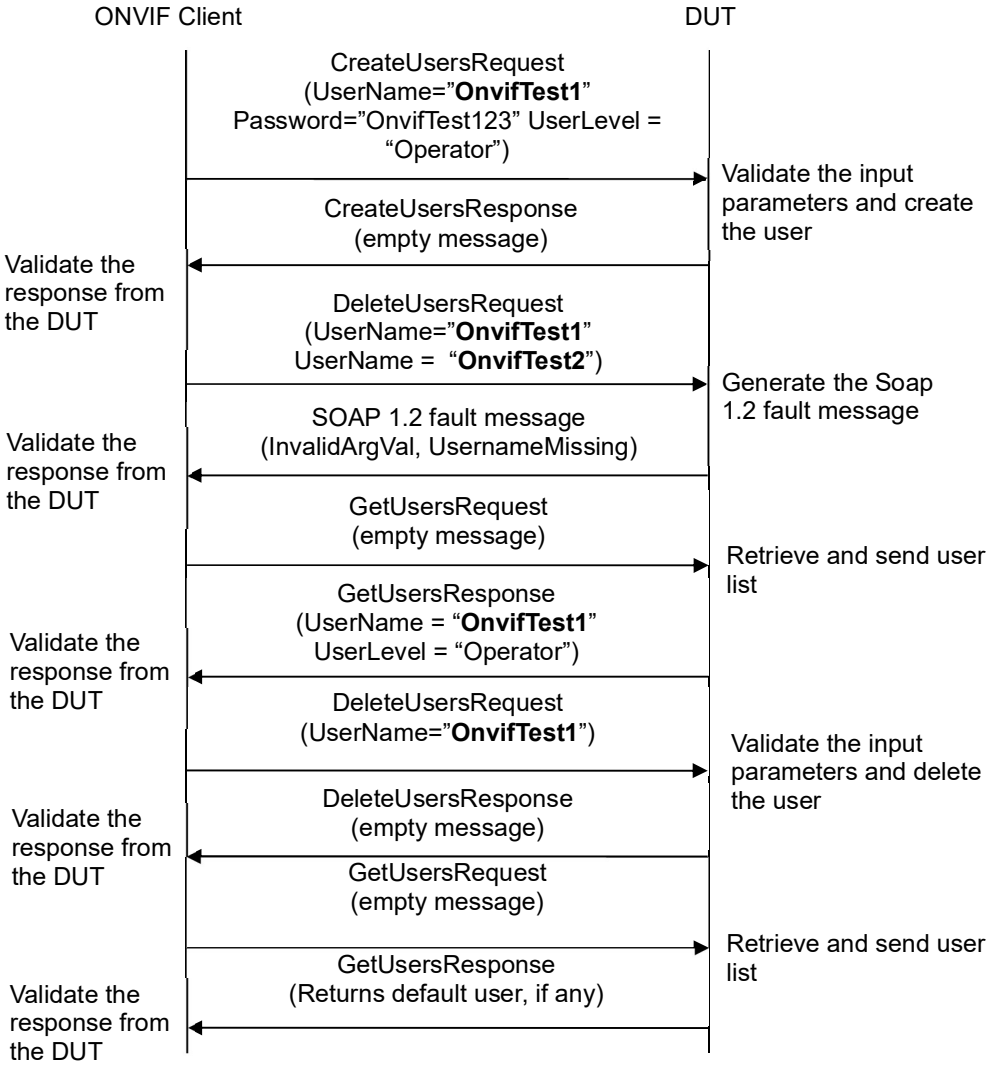
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify the behavior of the DeleteUsers command, if a non-existing user is deleted.

Pre-Requisite: ONVIF Client may need to operate in administrator mode to execute this test case.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke CreateUsersRequest message (Username = "OnvifTest1" Password = "OnvifTest123" UserLevel = "Operator"), to create the user in the DUT.
4. Verify that DUT sends CreateUsersResponse message (empty message).
5. ONVIF Client will invoke DeleteUsersRequest message (Username = "OnvifTest1", Username = "OnvifTest2"), to delete user.
6. Verify that the DUT responds with Soap 1.2 fault message (InvalidArgVal, UsernameMissing).

7. ONVIF Client will invoke GetUsersRequest message (empty message), to retrieve the user list from the DUT.
8. Verify that DUT sends GetUsersResponse message (Username = “OnvifTest1” UserLevel = Operator).
9. ONVIF Client will invoke DeleteUsersRequest message (Username = “OnvifTest1”), to delete the user.
10. Verify that DUT sends DeleteUsersResponse message (empty message).
11. ONVIF Client will invoke GetUsersRequest message (empty message), to retrieve the user list from the DUT.
12. Verify that DUT sends GetUsersResponse message (Returns default users, if any).

Test Result:

PASS –

The DUT passed all assertions

FAIL –

The DUT did not send SOAP 1.2 fault message.

The DUT did not send correct SOAP 1.2 fault message (InvalidArgVal, UsernameMissing).

The DUT deletes the user “OnvifTest1” at step 5.

The DUT did not send DeleteUsersResponse message.

The DUT did not send CreateUsersResponse message.

The DUT did not send GetUsersResponse message.

Note:

The DUT may return SOAP Fault 1.2 “TooManyUsers” for the CreateUsers command. Such SOAP 1.2 fault message shall be treated as PASS case for this test.

The DUT may return a greater number of users than actually created in this test case, i.e. default users if any might be returned.

6.4.5 SECURITY COMMAND DELETEUSERS DELETE ALL USERS

Test Label: Device Management Security Command DeleteUsers Verification, Deleting All Users.

Test Case ID: DEVICE-4-1-6

ONVIF Core Specification Coverage: Delete users.

Command Under Test: DeleteUsers.

WSDL Reference: devicemgmt.wsdl

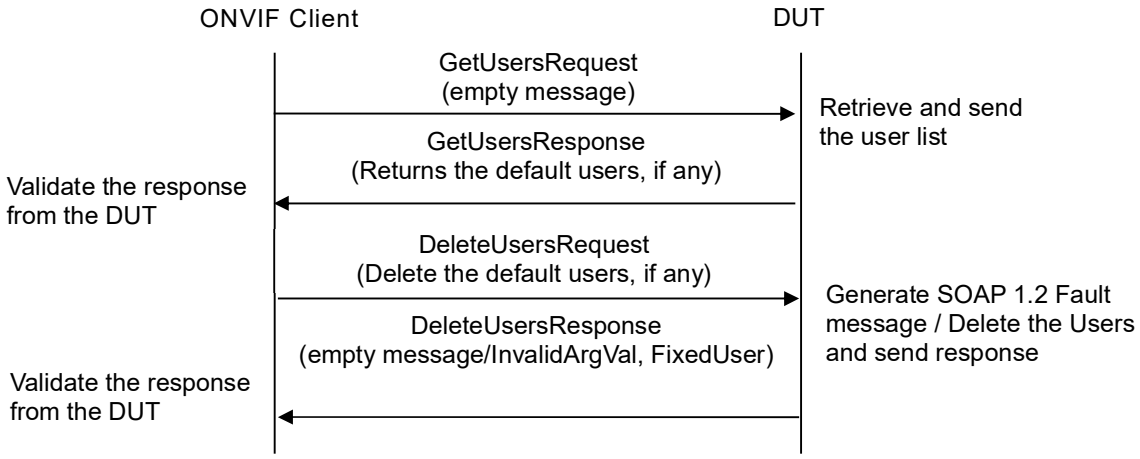
Test Purpose: To verify the behavior of the DeleteUsers command, when all the users are deleted.

Pre-Requisite: The ONVIF Client may need to operate in administrator mode to execute this test

case.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetUsersRequest message (empty message), to retrieve the user list from the DUT.
4. Verify that the DUT sends GetUsersResponse message (Returns default users, if any).
5. ONVIF Client will invoke DeleteUsersRequest message (Username= Default Users), to delete the default users if there are any.
6. Verify that the DUT sends DeleteUsersResponse message (empty message / InvalidArgVal, FixedUser). Depending upon the implementation DUT may respond with SOAP 1.2 fault message or send empty response.

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send correct SOAP 1.2 fault message (InvalidArgVal, FixedUser).

The DUT did not send GetUsersResponse message.

The DUT did not send DeleteUsersResponse message.

Note:

The DUT may return the default users, if any.

It is not possible to recover the default user if it was deleted during the test case execution.

The original user, used to access the DUT, shall be restored in case all users were deleted.

6.4.6 SECURITY COMMAND SETUSER

Test Label: Device Management Security Command SetUser Verification.

Test Case ID: DEVICE-4-1-7

ONVIF Core Specification Coverage: Set users.

Command Under Test: SetUser.

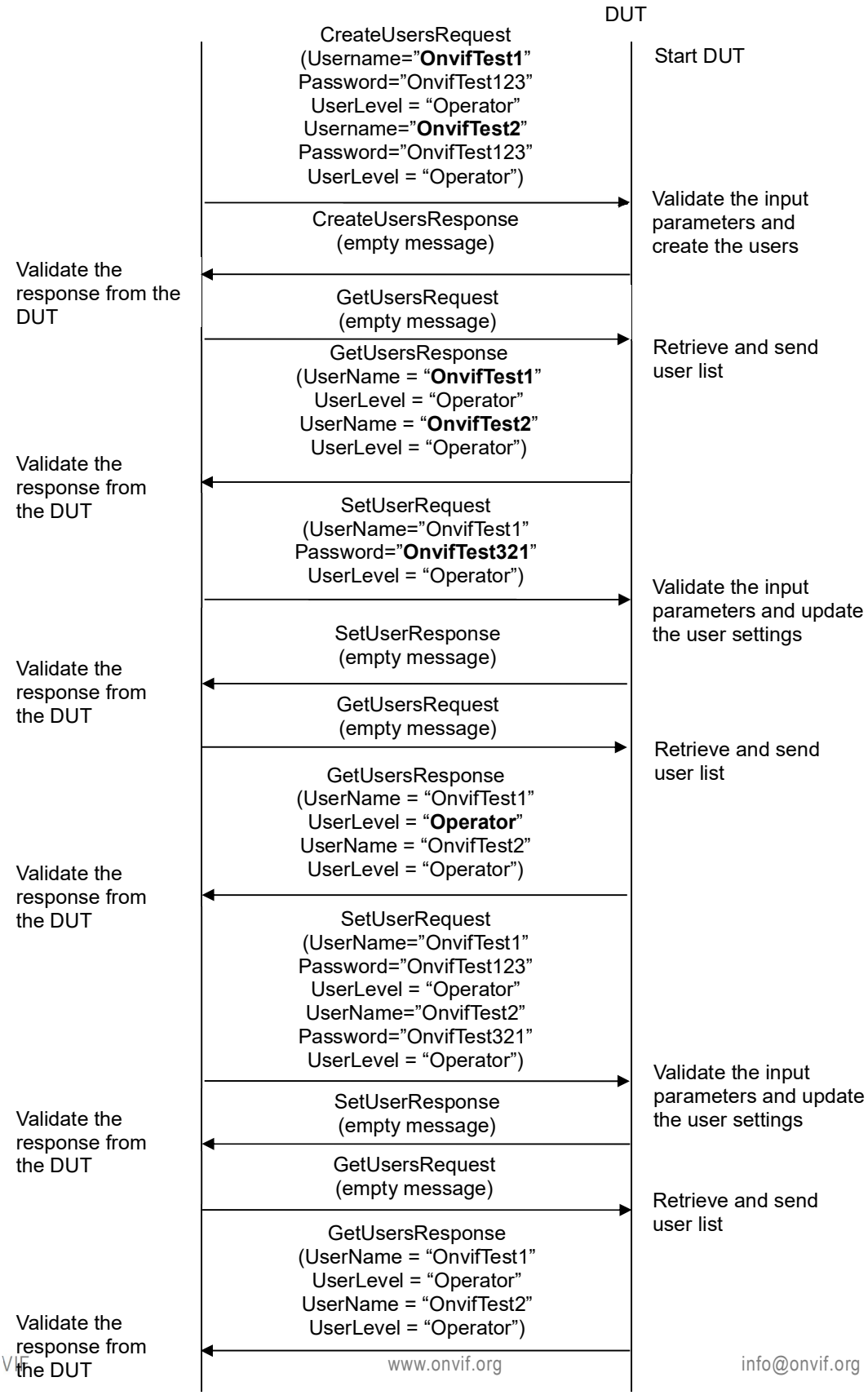
WSDL Reference: devicemgmt.wsdl

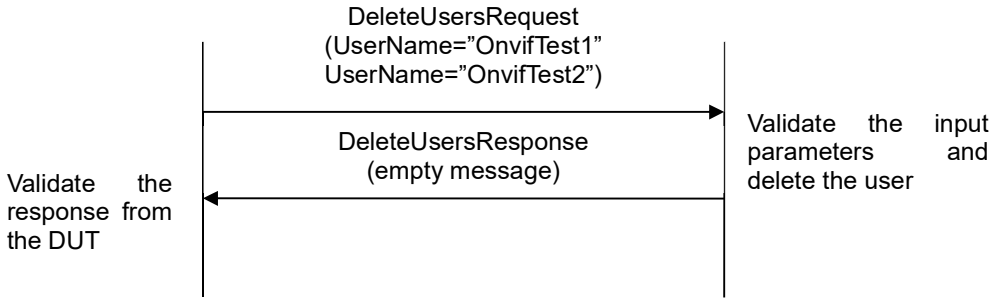
Test Purpose: To verify the behavior of SetUser command.

Pre-Requisite: The ONVIF Client may need to operate in administrator mode to execute this test case.

Test Configuration: ONVIF Client and DUT

Test Sequence:


DUT
CreateUsersRequest
(Username="OnvifTest1"
Password="OnvifTest123"
UserLevel = "Operator"
Username="OnvifTest2"
Password="OnvifTest123"
UserLevel = "Operator")
Start DUT
Validate the input parameters and create the users
CreateUsersResponse
(empty message)
Validate the response from the DUT
GetUsersRequest
(empty message)
Retrieve and send user list
GetUsersResponse
(UserName = "OnvifTest1"
UserLevel = "Operator"
UserName = "OnvifTest2"
UserLevel = "Operator")
Validate the response from the DUT
SetUserRequest
(UserName="OnvifTest1"
Password="OnvifTest321"
UserLevel = "Operator")
Validate the input parameters and update the user settings
SetUserResponse
(empty message)
Validate the response from the DUT
GetUsersRequest
(empty message)
Retrieve and send user list
GetUsersResponse
(UserName = "OnvifTest1"
UserLevel = "Operator"
UserName = "OnvifTest2"
UserLevel = "Operator")
Validate the response from the DUT
SetUserRequest
(UserName="OnvifTest1"
Password="OnvifTest123"
UserLevel = "Operator"
UserName="OnvifTest2"
Password="OnvifTest321"
UserLevel = "Operator")
Validate the input parameters and update the user settings
SetUserResponse
(empty message)
Validate the response from the DUT
GetUsersRequest
(empty message)
Retrieve and send user list
GetUsersResponse
(UserName = "OnvifTest1"
UserLevel = "Operator"
UserName = "OnvifTest2"
UserLevel = "Operator")
Validate the response from the DUT



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke CreateUsersRequest message (Username = "OnvifTest1" Password = "OnvifTest123" UserLevel = "Operator", Username = "OnvifTest2" Password = "OnvifTest123" UserLevel = "Operator"), to create the user in the DUT.
4. Verify that the DUT sends CreateUsersResponse message (empty message).
5. ONVIF Client will invoke GetUsersRequest message (empty message), to retrieve the user list from the DUT.
6. Verify that the DUT sends GetUsersResponse message (Username = "OnvifTest1" UserLevel = Operator, Username = "OnvifTest2" UserLevel = "Operator").
7. ONVIF Client will invoke SetUserRequest message (Username = "OnvifTest1" Password = "OnvifTest321" UserLevel = "Operator"), to update the user settings in the DUT.
8. Verify that the DUT sends SetUserResponse message (empty message).
9. ONVIF Client will invoke GetUsersRequest message (empty message), to retrieve the user list from the DUT.
10. Verify that the DUT sends GetUsersResponse message (Username= "OnvifTest1" UserLevel = "Operator", Username = "OnvifTest2" UserLevel = "Operator").
11. ONVIF Client will invoke SetUserRequest message (Username = "OnvifTest1" Password = "OnvifTest123" UserLevel = "Operator", Username = "OnvifTest2" Password = "OnvifTest321" UserLevel = "Operator"), to update user settings in the DUT.
12. Verify that the DUT sends SetUserResponse message (empty message).
13. ONVIF Client will invoke GetUsersRequest (empty message), to retrieve the user list from the DUT.
14. Verify that the DUT sends GetUsersResponse message (Username = OnvifTest1 UserLevel = "Operator", Username = OnvifTest2 UserLevel = "User").
15. ONVIF Client will invoke DeleteUsersRequest message (Username = "OnvifTest1", Username = "OnvifTest2") to delete the user in the DUT.
16. Verify that the DUT sends DeleteUsersResponse message (Empty Message).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not update the settings of the user(s).

The DUT did not send GetUsersResponse message.

The DUT did not send SetUserResponse message.

The DUT did not send DeleteUsersResponse message.

The DUT did not send CreateUsersResponse message.

Note:

The DUT may return SOAP Fault 1.2 “TooManyUsers” for the CreateUsers command. In this case test shall be executed with the default users if any.

The DUT may return more number of users than actually created in this test case, i.e. default users if any might be returned.

6.4.7 SECURITY COMMAND USER MANAGEMENT ERROR CASE

Test Label: Device Management Security Command SetUser Verification, User Settings Non Existing User.

Test Case ID: DEVICE-4-1-8

ONVIF Core Specification Coverage: Set users.

Command Under Test: SetUser.

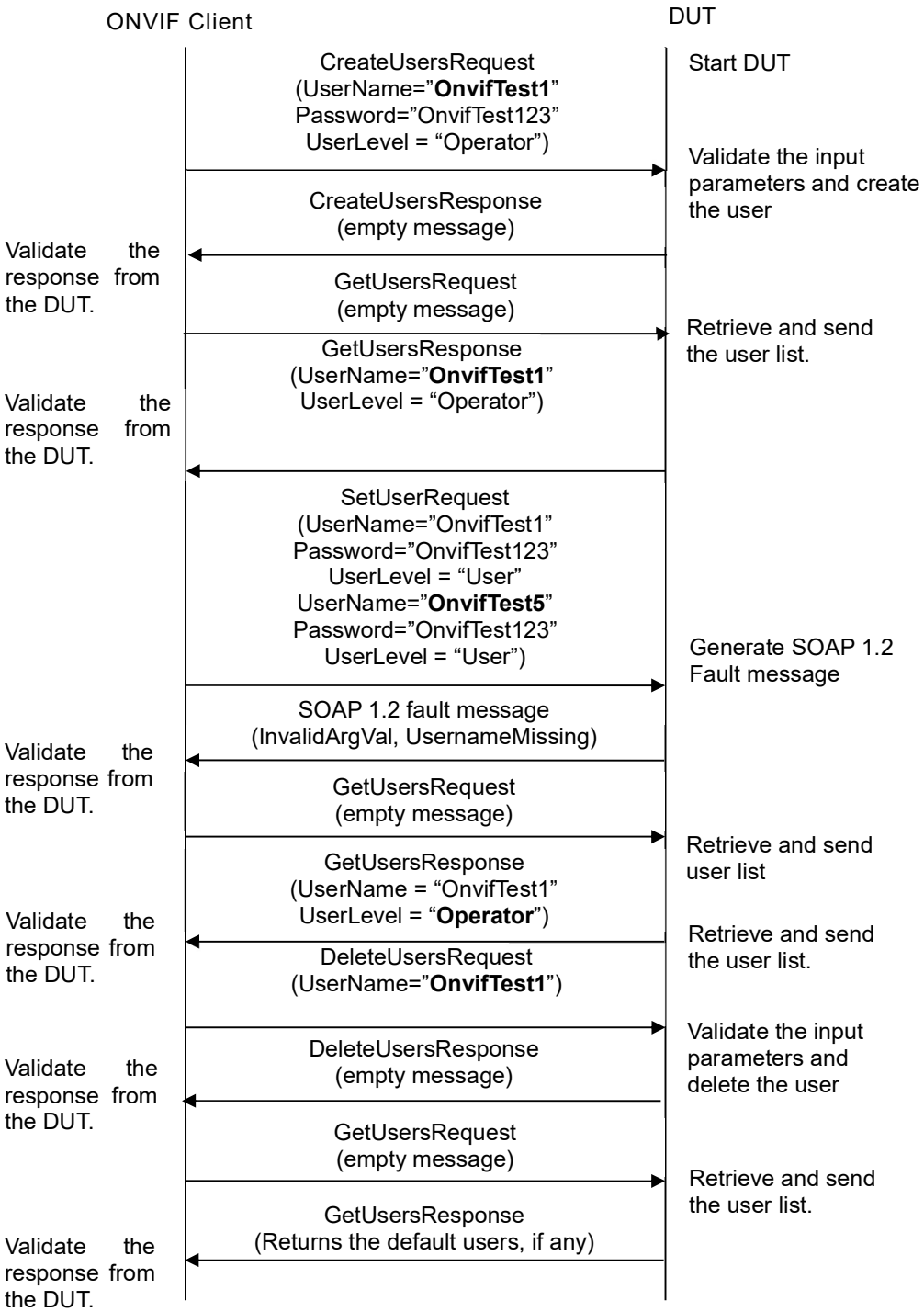
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify the behavior of the SetUser command when updating the settings of non-existing user.

Pre-Requisite: The ONVIF Client may need to operate in administrator mode to execute this test case.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

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

3. ONVIF Client will invoke CreateUsersRequest message (Username = “OnvifTest1” Password = ”OnvifTest123” UserLevel = “Operator”), to create the user in the DUT.
4. Verify that the DUT sends CreateUsersResponse message (empty message).
5. ONVIF Client will invoke GetUsersRequest message (empty message) to retrieve the user list from the DUT
6. Verify that the DUT sends the GetUsersResponse message (Username = “OnvifTest1” UserLevel = “Operator”).
7. ONVIF Client will invoke SetUserRequest message (Username = “OnvifTest1” Password=OnvifTest123 UserLevel = “User”, Username = “OnvifTest5” Password = “OnvifTest123” UserLevel = “User”) to update the user settings in the DUT.
8. Verify that the DUT generates SOAP 1.2 fault message (InvalidArgVal, UsernameMissing).
9. ONVIF Client will invoke GetUsersRequest message (empty message), to retrieve the user list from the DUT.
10. Verify that the DUT sends the GetUsersResponse message (Username = “OnvifTest1” UserLevel = “Operator”).
11. ONVIF Client will invoke DeleteUsersRequest message (Username = “OnvifTest1”), to delete the user in the DUT.
12. Verify that the DUT sends DeleteUsersResponse message (Empty Message).
13. ONVIF Client will invoke GetUsersRequest message (empty message) to retrieve the user list form the DUT
14. Verify that the DUT sends the GetUsersResponse message (returns default users, if any).

Test Result:

PASS –

The DUT passed all assertions

FAIL –

The DUT did not send SOAP 1.2 fault message.

The DUT did not send correct SOAP 1.2 fault message (InvalidArgVal, UsernameMissing).

The DUT did not send GetUsersResponse message.

The DUT did not send SetUserResponse message.

The DUT did not send CreateUsersResponse message.

The DUT did not send DeleteUsersResponse message.

The DUT updates the settings of the user “OnvifTest1”.

Note:

The DUT may return SOAP Fault 1.2 “TooManyUsers” for the CreateUsers command. In this case, test shall be executed with the default users if any.

The DUT may return a greater number of users than actually created in this test case, i.e. default users if any might be returned.

6.4.8 SECURITY COMMAND CREATEUSERS

Test Label: Device Management Security Command CreateUsers Verification.

Test Case ID: DRAFT-DEVICE-4-1-9

ONVIF Core Specification Coverage: Create users.

Command Under Test: CreateUsers

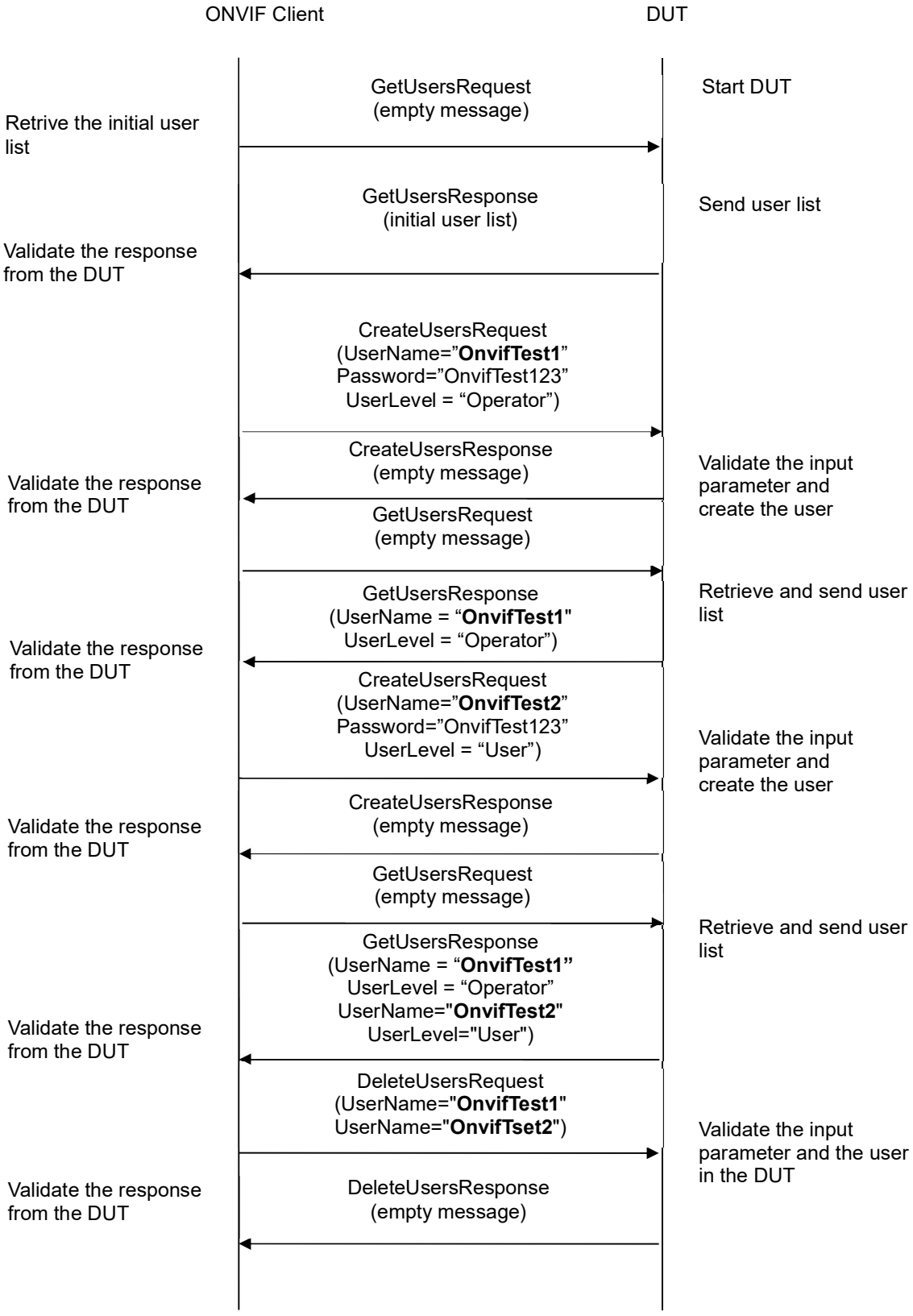
WSDL Reference: devicemgmt.wsdl

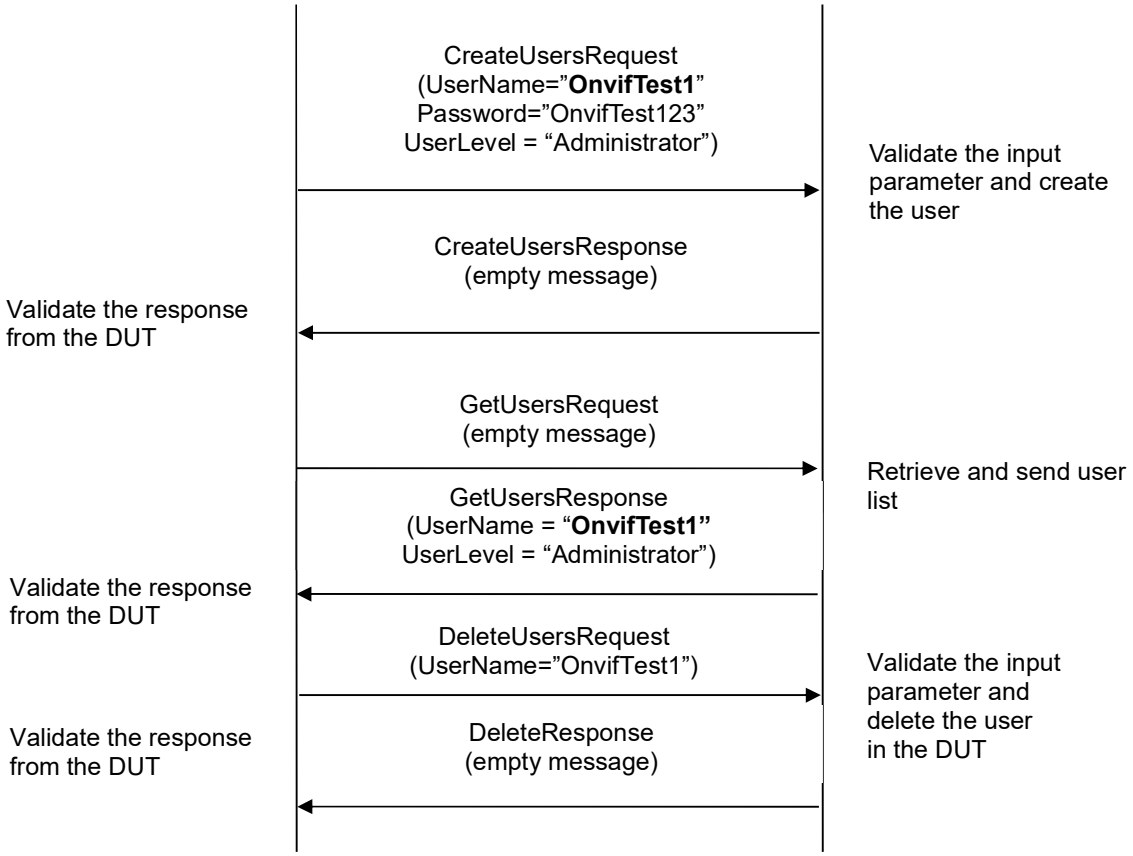
Test Purpose: To verify the behavior of CreateUsers command.

Pre-Requisite: The ONVIF Client may need to operate in administrator mode to execute this test case.

Test Configuration: ONVIF Client and DUT.

Test Sequence:


ONVIF Client
DUT
Start DUT
GetUsersRequest
(empty message)
Retrive the initial user list
GetUsersResponse
(initial user list)
Send user list
Validate the response from the DUT
CreateUsersRequest
(UserName="OnvifTest1"
Password="OnvifTest123"
UserLevel = "Operator")
CreateUsersResponse
(empty message)
Validate the input parameter and create the user
Validate the response from the DUT
GetUsersRequest
(empty message)
GetUsersResponse
(UserName = "OnvifTest1"
UserLevel = "Operator")
Retrieve and send user list
Validate the response from the DUT
CreateUsersRequest
(UserName="OnvifTest2"
Password="OnvifTest123"
UserLevel = "User")
Validate the input parameter and create the user
CreateUsersResponse
(empty message)
Validate the response from the DUT
GetUsersRequest
(empty message)
GetUsersResponse
(UserName = "OnvifTest1"
UserLevel = "Operator"
UserName="OnvifTest2"
UserLevel="User")
Retrieve and send user list
Validate the response from the DUT
DeleteUsersRequest
(UserName="OnvifTest1"
UserName="OnvifTset2")
Validate the input parameter and the user in the DUT
DeleteUsersResponse
(empty message)
Validate the response from the DUT



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetUsersRequest message (empty message), to retrieve the initial user list from the DUT.
4. Verify that the DUT sends GetUsersResponse message (empty message).
5. ONVIF Client will invoke CreateUsersRequest message (Username = "OnvifTest1" Password = "OnvifTest123" UserLevel = "Operator"), to create the user in the DUT.
6. Verify that the DUT sends CreateUsersResponse message (empty message).
7. ONVIF Client will invoke GetUsersRequest message (empty message), to retrieve the user list from the DUT.
8. Verify that the DUT sends GetUsersResponse message (Username = "OnvifTest1", UserLevel = "Operator").
9. ONVIF Client will invoke CreateUsersRequest message (Username = "OnvifTest2" Password = "OnvifTest123" UserLevel = "User") to create the user in the DUT.

10. Verify that the DUT sends CreateUsersResponse message (empty message).
11. ONVIF Client will invoke GetUsersRequest message (empty message), to retrieve the user list from the DUT.
12. Verify that DUT sends the GetUsersResponse message (Username = “OnvifTest1” UserLevel = “Operator”, Username = ”OnvifTest2” UserLevel = “User”).
13. ONVIF Client will invoke DeleteUsersRequest message (Username = ”OnvifTest1”, Username = ”OnvifTest2”) to delete the users in the DUT.
14. Verify that DUT sends DeleteUsersResponse message (empty Message).
15. ONVIF Client will invoke CreateUsersRequest message (Username = "OnvifTest1" Password = "OnvifTest123" UserLevel = "Administrator"), to create the user in the DUT.
16. Verify that the DUT sends CreateUsersResponse message (empty message).
17. ONVIF Client will invoke GetUsersRequest message (empty message) to retrieve the user list from the DUT.
18. Verify that the DUT sends GetUsersResponse message (Username = "OnvifTest1", UserLevel = "Administrator").
19. ONVIF Client will invoke DeleteUsersRequest message (Username = "OnvifTest1") to delete the users in the DUT.
20. Verify that the DUT sends DeleteUsersResponse message (empty message).

Test Result:

PASS –

The DUT passed all assertions.

The DUT creates a user either at step 6 or at step 10 or at step 16 successfully or the DUT creates users at step 6, step 10 and step 16 successfully.

FAIL –

The DUT did not send GetUsersResponse message.

The DUT did not send CreateUsersResponse message.

The DUT did not send DeleteUsersResponse message.

Note:

The DUT may return SOAP Fault 1.2 “TooManyUsers” for the CreateUsers command if the capability MaxUsers is not present or equals the size of the users list. Such SOAP 1.2 fault message shall be treated as PASS case in this case. See Annex A.20 TooManyUsers fault check for details.

The DUT may return a greater number of users than actually created in this test case, i.e. default users if any might be returned.

At some DUTs it might not be possible to create user with all levels. So if the DUT successfully creates a user either at step 6 or step 10 or step 16 successfully, then this test case shall be treated as PASS case.

6.4.9 GET REMOTE USER

Test Label: Get Remote User Verification

Test Case ID: DEVICE-4-1-10

ONVIF Core Specification Coverage: Get remote user (ONVIF Core Specification)

Command Under Test: GetRemoteUser

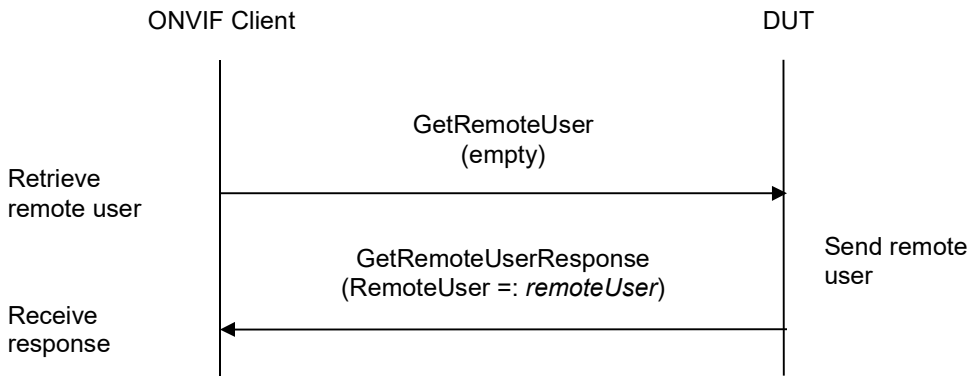
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify Get Remote User.

Pre-requisite: Remote user handlin is supported by the DUT as indicated by the Security.RemoteUserHandling capability.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF client invokes **GetRemoteUser**.
4. The DUT responds with **GetRemoteUserResponse** message with parameters.
   - RemoteUser =: *remoteUser*
5. If *remoteUser* is not empty and *remoteUser*.Password is not skkiped, FAIL the test.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send **GetRemoteUserResponse** message.

Driving IP-based physical security through global standardization

6.4.10 SET REMOTE USER

Test Label: Set Remote User Verification

Test Case ID: DEVICE-4-1-11

ONVIF Core Specification Coverage: Set remote user (ONVIF Core Specification)

Command Under Test: SetRemoteUser

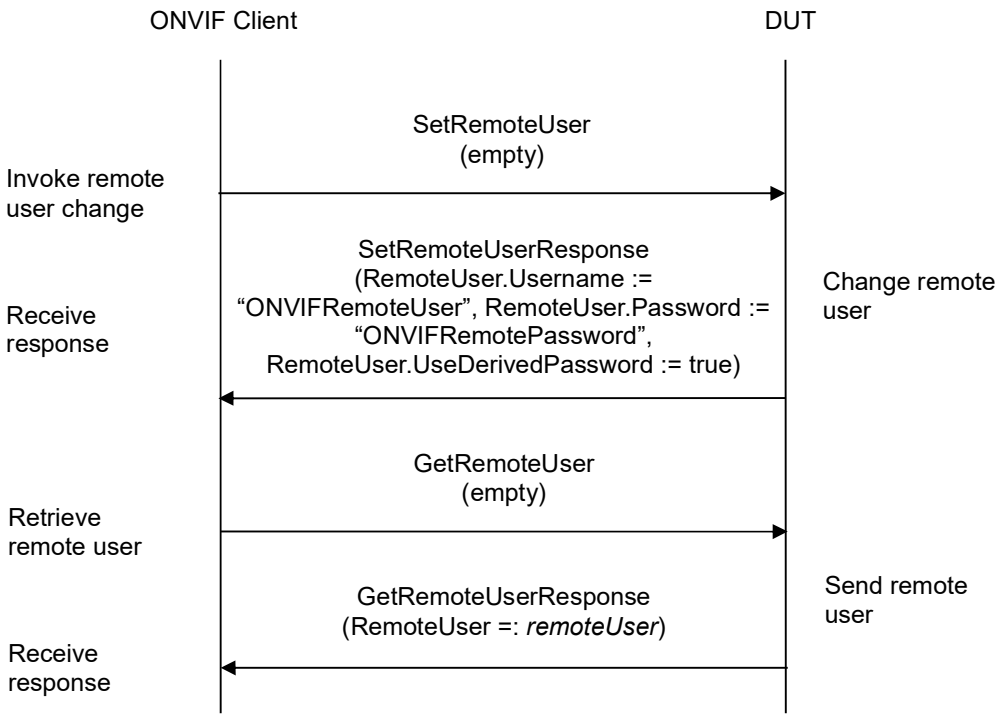
WSDL Reference: devicemgmt.wsdl

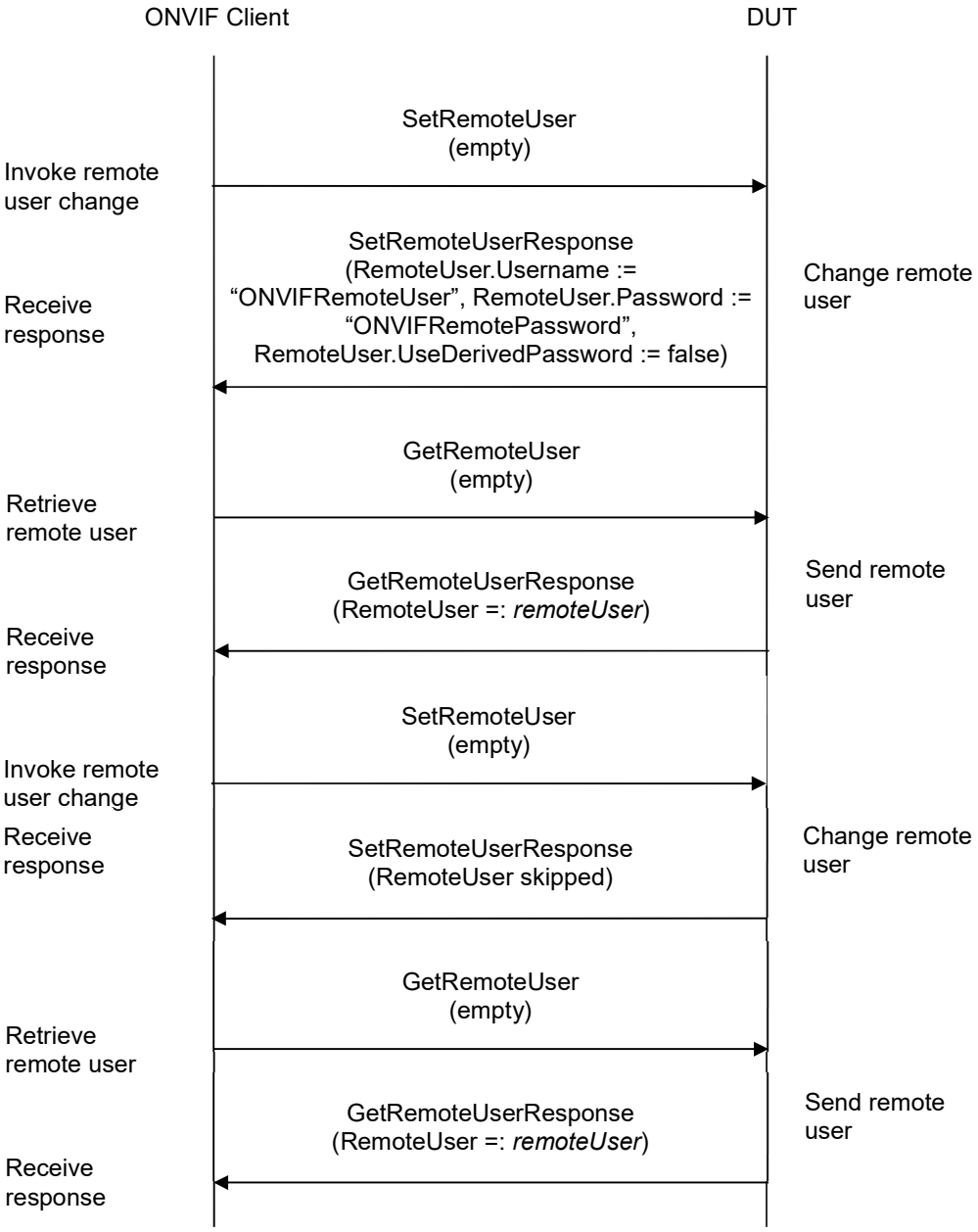
Test Purpose: To verify Set Remote User.

Pre-requisite: Remote user handling is supported by the DUT as indicated by the Security.RemoteUserHandling capability.

Test Configuration: ONVIF Client and DUT

Test Sequence:





Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF client invokes **SetRemoteUser** with parameters
   - RemoteUser.Username := "ONVIFRemoteUser"

- RemoteUser.Password := “ONVIFRemotePassword”
- RemoteUser.UseDerivedPassword := true

4. The DUT responds with **SetRemoteUserResponse** message.
5. ONVIF client invokes **GetRemoteUser**.
6. The DUT responds with **GetRemoteUserResponse** message with parameters.
   - RemoteUser =: *remoteUser*
7. If *remoteUser* is empty, FAIL the test and skip other steps.
8. If *remoteUser*.User is not equal to “ONVIFRemoteUser”, FAIL the test and skip other steps.
9. If *remoteUser*.UseDerivedPassword is not equal to true, FAIL the test and skip other steps.
10. If *remoteUser*.Password is not skkiped, FAIL the test and skip other steps.
11. ONVIF client invokes **SetRemoteUser** with parameters
    - RemoteUser.Username := “ONVIFRemoteUser”
    - RemoteUser.Password := “ONVIFRemotePassword”
    - RemoteUser.UseDerivedPassword := false
12. The DUT responds with **SetRemoteUserResponse** message.
13. ONVIF client invokes **GetRemoteUser**.
14. The DUT responds with **GetRemoteUserResponse** message with parameters.
    - RemoteUser =: *remoteUser*
15. If *remoteUser* is empty, FAIL the test and skip other steps.
16. If *remoteUser*.User is not equal to “ONVIFRemoteUser”, FAIL the test and skip other steps.
17. If *remoteUser*.UseDerivedPassword is not equal to false, FAIL the test and skip other steps.
18. If *remoteUser*.Password is not skkiped, FAIL the test and skip other steps.
19. ONVIF client invokes **SetRemoteUser** with parameters
    - RemoteUser skipped
20. The DUT responds with **SetRemoteUserResponse** message.
21. ONVIF client invokes **GetRemoteUser**.
22. The DUT responds with **GetRemoteUserResponse** message with parameters.
    - RemoteUser =: *remoteUser*
23. If *remoteUser* is not empty, FAIL the test.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send **GetRemoteUserResponse** message.

The DUT did not send **SetRemoteUserResponse** message.

I/O

6.4.11 IO COMMAND GETRELAYOUTPUTS

Test Label: Device Management DUT Input/Output (I/O) Command GetRelayOutputs Test.

Test Case ID: DEVICE-5-1-1

ONVIF Core Specification Coverage: Get relay outputs

Command under Test: GetRelayOutputs

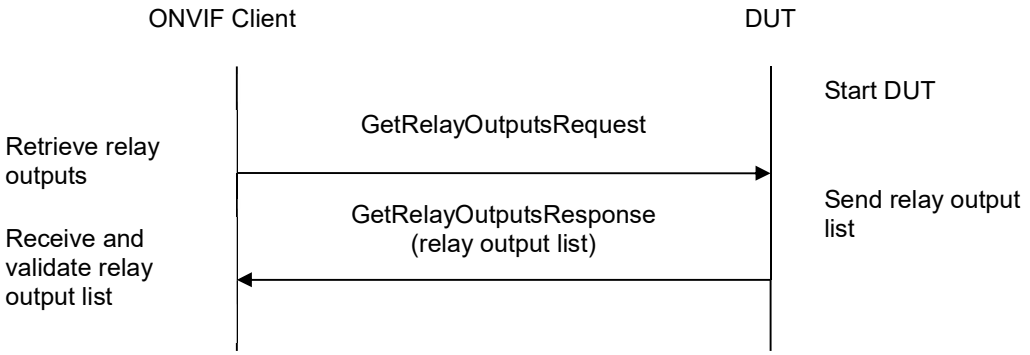
WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: 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 –

The DUT passed 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.

The DUT sent an empty list of RelayOutputs.

6.4.12 RELAY OUTPUTS COUNT IN GETRELAYOUTPUTS AND GETCAPABILITIES

Test Label: Relay Outputs Count in GetRelayOutputsResponse Message and in GetCapabilitiesResponse Message Test.

Test Case ID: DEVICE-5-1-2

ONVIF Core Specification Coverage: Get relay outputs, Capability exchange

Command under Test: GetRelayOutputs

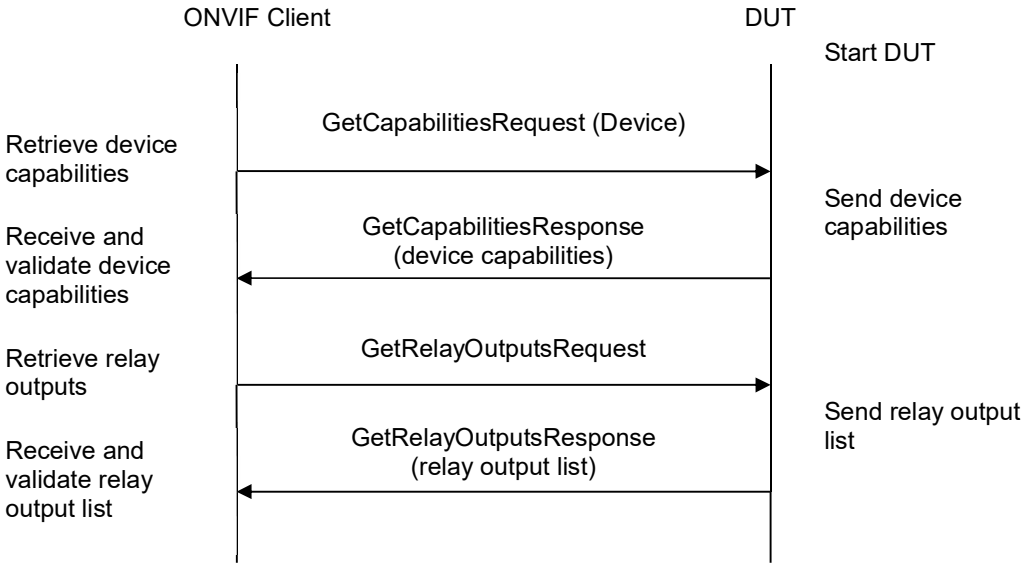
WSDL Reference: devicemgmt.wsdl

Test Purpose: To check that the number of Relay outputs is the same in GetRelayOutputsResponse message and in GetCapabilitiesResponse.

Pre-Requisite: 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 GetCapabilitiesRequest message (**Device**) to retrieve device capabilities.
4. Verify the GetCapabilitiesResponse message from the DUT.
5. ONVIF Client invokes GetRelayOutputsRequest message to retrieve relay outputs supported by the DUT.
6. Verify the GetRelayOutputsResponse message from the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetCapabilitiesResponse message.

The DUT did not send valid GetCapabilitiesResponse message.

The DUT did not send GetRelayOutputsResponse message.

The DUT did not send valid GetRelayOutputsResponse message.

The DUT sent Relay Outputs, its number in GetRelayOutputsResponse message differs from the one in Device.IO.RelayOutputs from GetCapabilitiesResponse message.

6.4.13 IO COMMAND SETRELAYOUTPUTSETTINGS

Test Label: Device Management DUT Input/Output (I/O) Command SetRelayOutputSettings Test.

Test Case ID: DEVICE-5-1-3

ONVIF Core Specification Coverage: Get relay outputs, Set relay output settings

Command under Test: GetRelayOutputs, SetRelayOutputSettings

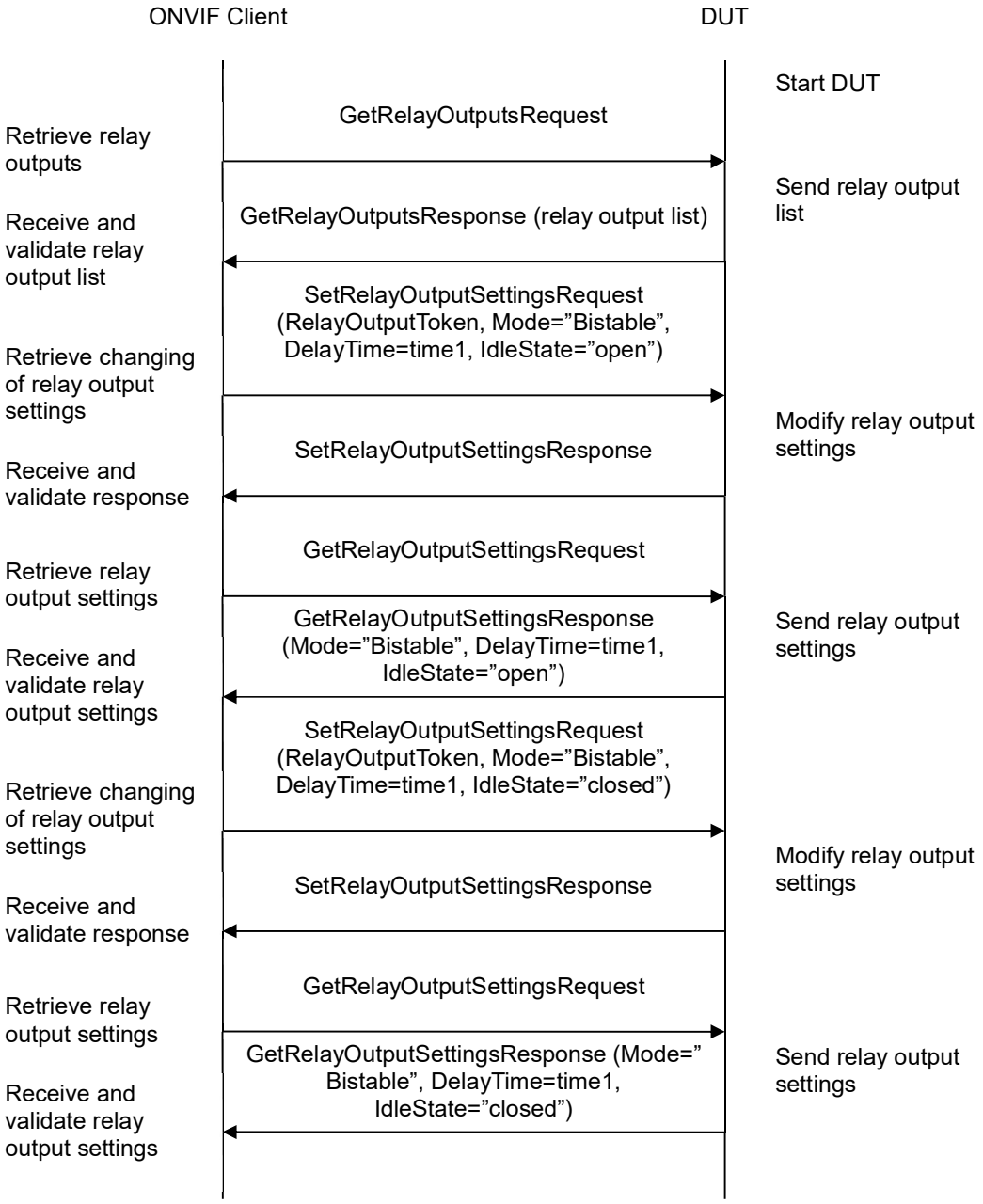
WSDL Reference: devicemgmt.wsdl

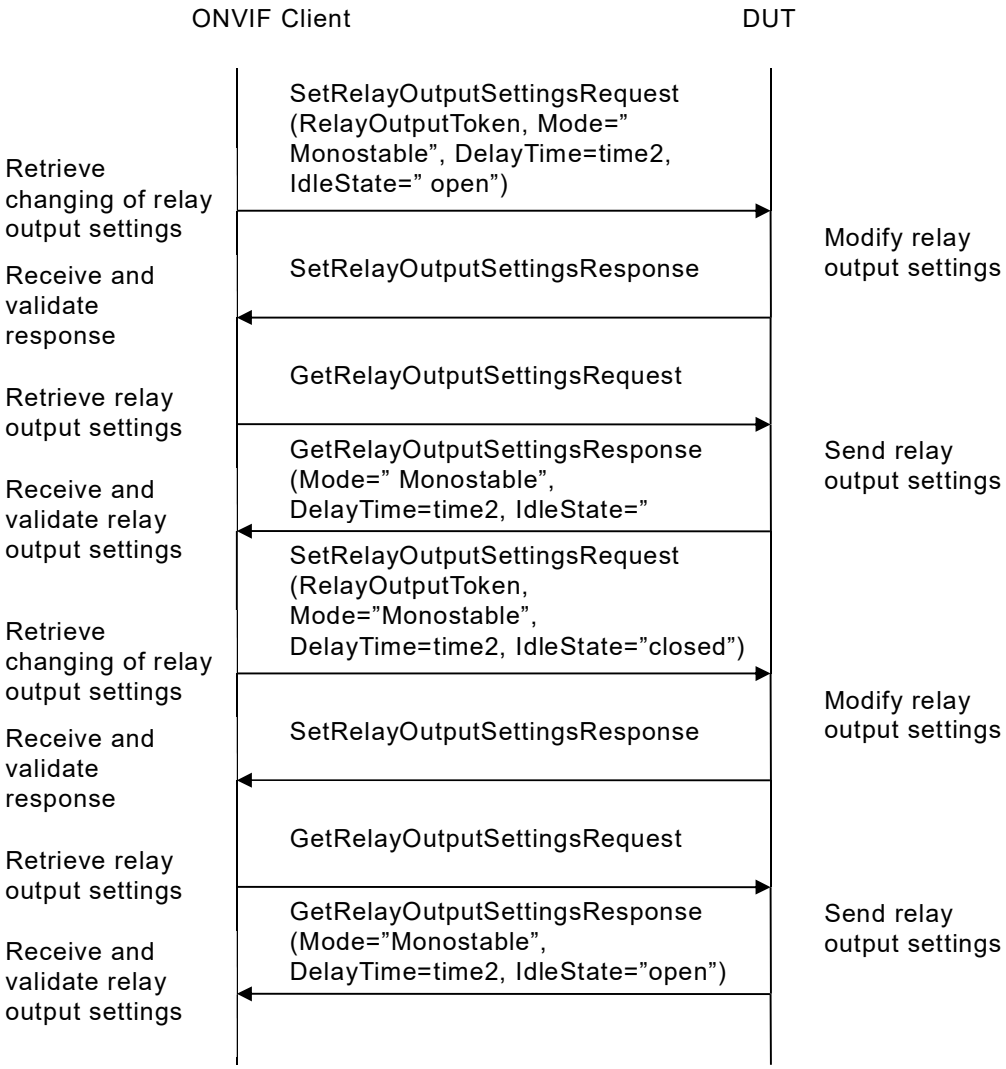
Test Purpose: To verify the behavior of SetRelayOutputSettings command.

Pre-Requisite: Relay Outputs supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
GetRelayOutputsRequest
Retrieve relay outputs
Send relay output list
GetRelayOutputsResponse (relay output list)
Receive and validate relay output list
SetRelayOutputSettingsRequest (RelayOutputToken, Mode="Bistable", DelayTime=time1, IdleState="open")
Retrieve changing of relay output settings
Modify relay output settings
SetRelayOutputSettingsResponse
Receive and validate response
GetRelayOutputSettingsRequest
Retrieve relay output settings
GetRelayOutputSettingsResponse (Mode="Bistable", DelayTime=time1, IdleState="open")
Send relay output settings
Receive and validate relay output settings
SetRelayOutputSettingsRequest (RelayOutputToken, Mode="Bistable", DelayTime=time1, IdleState="closed")
Retrieve changing of relay output settings
Modify relay output settings
SetRelayOutputSettingsResponse
Receive and validate response
GetRelayOutputSettingsRequest
Retrieve relay output settings
GetRelayOutputSettingsResponse (Mode="Bistable", DelayTime=time1, IdleState="closed")
Send relay output settings
Receive and validate relay output settings



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. Verify the GetRelayOutputsResponse message from the DUT.
6. If the DUT supports Bistable Mode with Open Idle state, ONVIF Client invokes SetRelayOutputSettingsRequest message (**RelayOutputToken, Mode="Bistable",**

DelayTime=time1, IdleState="open") to set new relay output settings. If the DUT does not support Bistable Mode with Open Idle state, then go to the step 10.

7. Verify the SetRelayOutputSettingsResponse message.
8. ONVIF Client invokes GetRelayOutputsRequest message to retrieve a list of all available relay outputs and their settings.
9. Verify GetRelayOutputsResponse message from the DUT. Check that values correspond the values from step 6 (except DelayTime value).
10. If the DUT supports Bistable Mode with Closed Idle state, ONVIF Client invokes SetRelayOutputSettingsRequest message (**RelayOutputToken, Mode="Bistable", DelayTime=time1, IdleState="closed"**) to set new relay output settings. If the DUT does not support Bistable Mode with Closed Idle state, then go to the step 14.
11. Verify the SetRelayOutputSettingsResponse message.
12. ONVIF Client invokes GetRelayOutputSettingsRequest message to retrieve a list of all available relay outputs and their settings.
13. Verify GetRelayOutputSettingsResponse message from the DUT. Check that values correspond the values from step 10 (except DelayTime value).
14. If the DUT supports Monostable Mode with Closed Idle state, ONVIF Client invokes SetRelayOutputSettingsRequest message (**RelayOutputToken, Mode="Monostable", DelayTime=time2, IdleState="open"**) to set new relay output settings. If the DUT does not support Monostable Mode with Closed Idle state, then go to the step 18.
15. Verify the SetRelayOutputSettingsResponse message.
16. ONVIF Client invokes GetRelayOutputSettingsRequest message to retrieve a list of all available relay outputs and their settings.
17. Verify GetRelayOutputSettingsResponse message from the DUT. Check that values correspond the values from step 14.
18. If the DUT supports Monostable Mode with Open Idle state, ONVIF Client invokes SetRelayOutputSettingsRequest message (**RelayOutputToken, Mode="Monostable", DelayTime=time2, IdleState="closed"**) to set new relay output settings. If the DUT does not support Monostable Mode with Open Idle state, then go to the step 22.
19. Verify the SetRelayOutputSettingsResponse message.
20. ONVIF Client invokes GetRelayOutputSettingsRequest message to retrieve a list of all available relay outputs and their settings.
21. Verify GetRelayOutputSettingsResponse message from the DUT. Check that values correspond the values from step 18.
22. Repeat 6-21 for all relay outputs from the list.

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 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 support at least one of the following: Bistable Mode with Open Idle state, Bistable Mode with Closed Idle state, Monostable Mode with Open Idle state, Monostable Mode with Closed Idle state

6.4.14 IO COMMAND SETRELAYOUTPUTSTATE – BISTABLE MODE (OPENED IDLE STATE)

Test Label: Device Management SetRelayOutputState Command Validation in the Case of Bistable Mode (Opened Idle State).

Test Case ID: DEVICE-5-1-5

ONVIF Core Specification Coverage: Trigger relay output

Command under Test: SetRelayOutputState

WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: Relay Outputs supported by the 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 invokes SetRelayOutputSettingsRequest message (RelayOutputToken, Mode="Bistable", DelayTime=time1, IdleState="open").
6. The DUT sends the SetRelayOutputSettingsResponse message.
7. ONVIF Client invokes SetRelayOutputStateRequest message (**RelayOutputToken, LogicalState="active"**)

8. Verify the SetRelayOutputStateResponse message from the DUT.
9. ONVIF Client invokes SetRelayOutputStateRequest message (**RelayOutputToken, LogicalState="inactive"**).
10. Verify the SetRelayOutputStateResponse message from the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetRelayOutputsResponse message.

The DUT did not send SetRelayOutputSettingsResponse message.

The DUT did not send SetRelayOutputStateResponse message.

The DUT did not send a valid SetRelayOutputStateResponse message.

6.4.15 IO COMMAND SETRELAYOUTPUTSTATE – BISTABLE MODE (CLOSED IDLE STATE)

Test Label: Device Management SetRelayOutputState Command Validation in the Case of Bistable Mode (Closed Idle State).

Test Case ID: DEVICE-5-1-6

ONVIF Core Specification Coverage: Trigger relay output

Command under Test: SetRelayOutputState

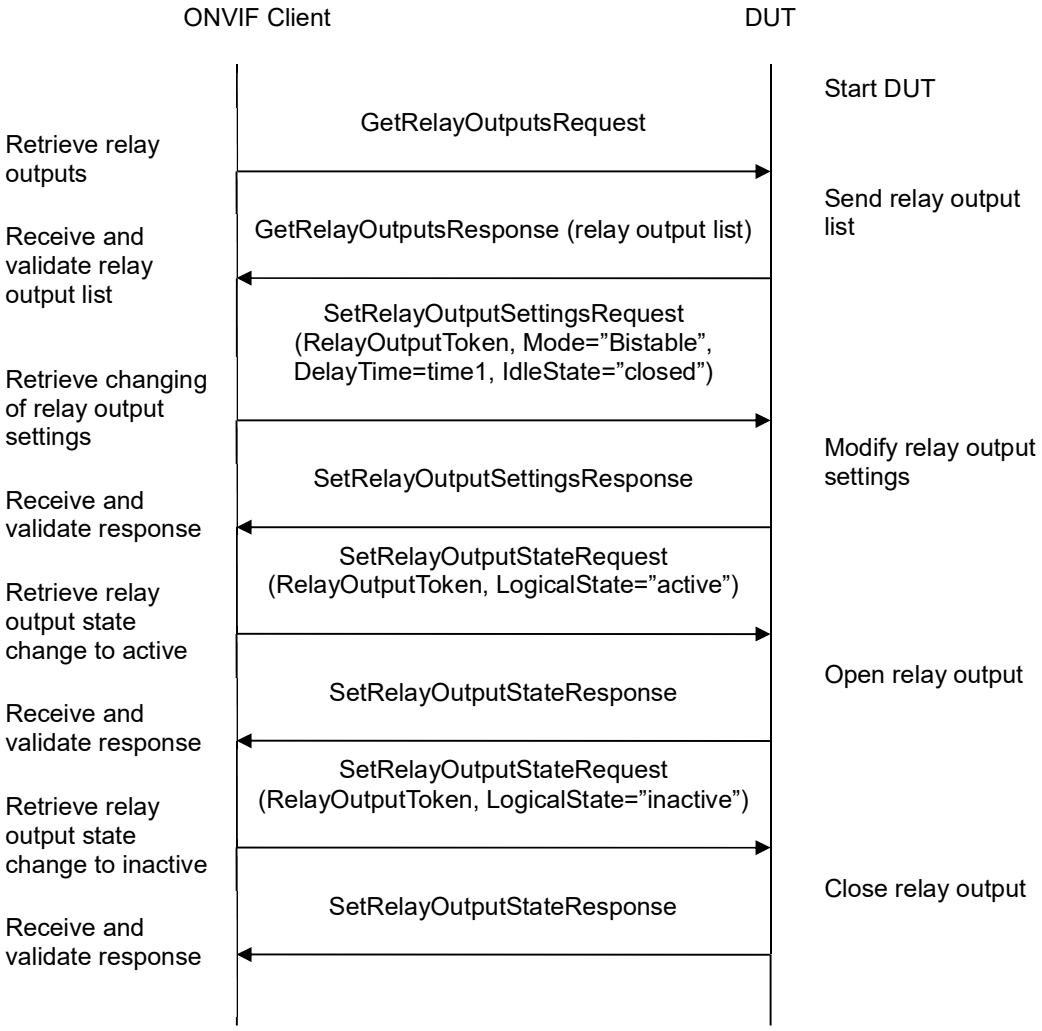
WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: 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 invokes SetRelayOutputSettingsRequest message (RelayOutputToken, Mode="Bistable", DelayTime=time1, IdleState="closed").
6. The DUT sends the SetRelayOutputSettingsResponse message.
7. ONVIF Client invokes SetRelayOutputStateRequest message (**RelayOutputToken, LogicalState="active"**)

8. Verify the SetRelayOutputStateResponse message from the DUT.
9. ONVIF Client invokes SetRelayOutputStateRequest message (RelayOutputToken, LogicalState="inactive").
10. Verify the SetRelayOutputStateResponse message from the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetRelayOutputsResponse message.

The DUT did not send SetRelayOutputSettingsResponse message.

The DUT did not send valid SetRelayOutputSettingsResponse message.

The DUT did not send SetRelayOutputStateResponse message.

6.4.16 IO COMMAND SETRELAYOUTPUTSTATE – MONOSTABLE MODE (OPENED IDLE STATE)

Test Label: Device Management SetRelayOutputState Command Validation in the Case of Monostable Mode (Opened Idle State).

Test Case ID: DEVICE-5-1-7

ONVIF Core Specification Coverage: Trigger relay output

Command under Test: SetRelayOutputState

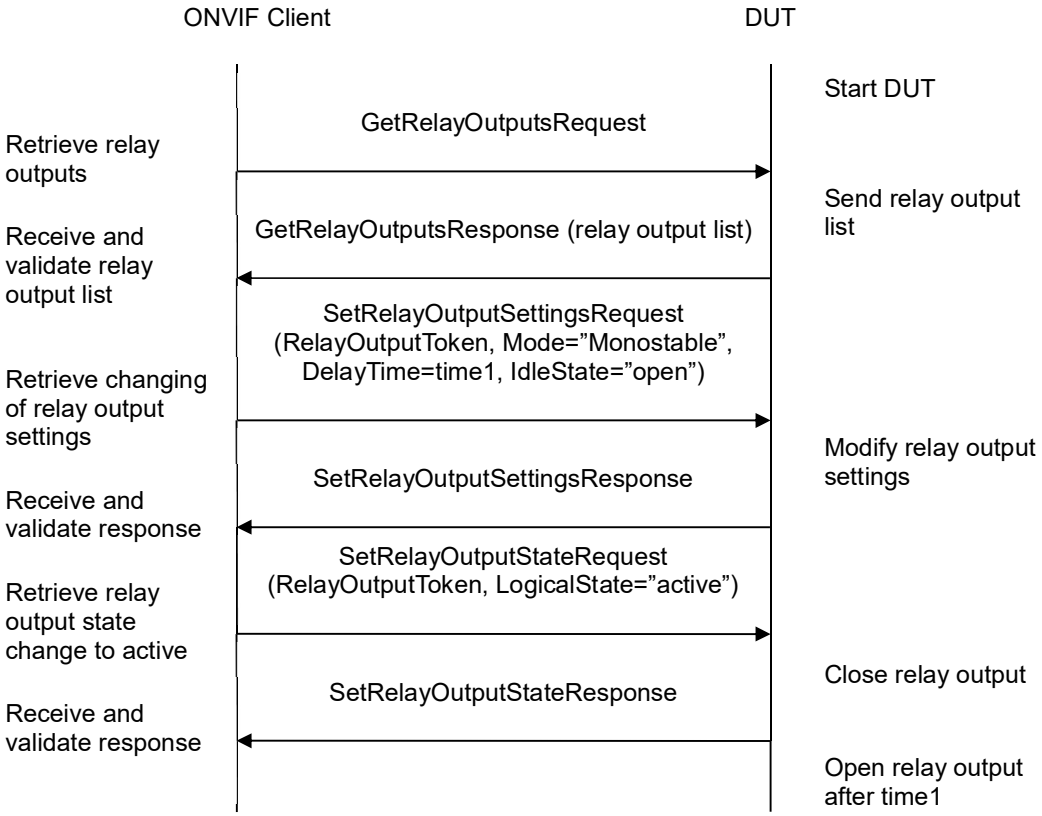
WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: 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 invokes SetRelayOutputSettingsRequest message (RelayOutputToken, Mode="Monostable", DelayTime=time1, IdleState="open").
6. The DUT sends the SetRelayOutputSettingsResponse message.
7. ONVIF Client invokes SetRelayOutputStateRequest message (**RelayOutputToken, LogicalState="active"**).
8. Verify the SetRelayOutputStateResponse message from the DUT.
9. Wait until time out time1 expires.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetRelayOutputsResponse message.

The DUT did not send SetRelayOutputSettingsResponse message.

The DUT did not send valid SetRelayOutputSettingsResponse message.

The DUT did not send SetRelayOutputStateResponse message.

6.4.17 IO COMMAND SETRELAYOUTPUTSTATE – MONOSTABLE MODE (CLOSED IDLE STATE)

Test Label: Device Management SetRelayOutputState Command Validation in the Case of Monostable Mode (Closed Idle State).

Test Case ID: DEVICE-5-1-8

ONVIF Core Specification Coverage: Trigger relay output

Command under Test: SetRelayOutputState

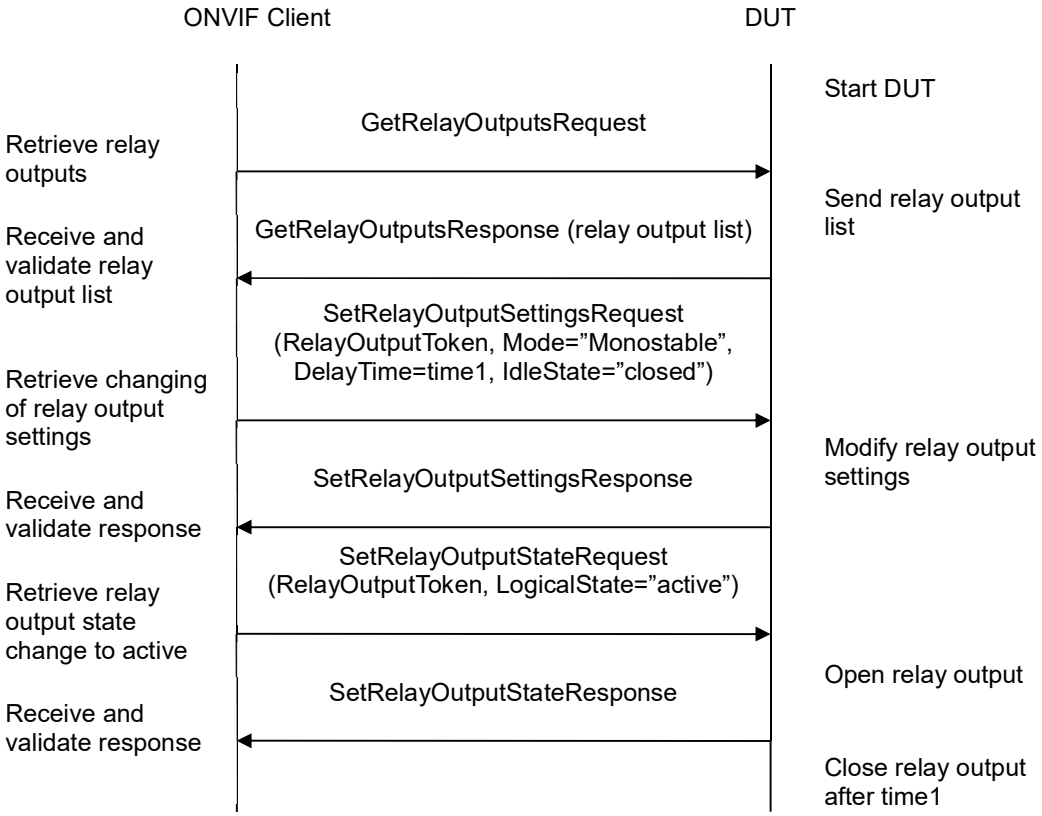
WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: 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 invokes SetRelayOutputSettingsRequest message (RelayOutputToken, Mode="Monostable", DelayTime=time1, IdleState="closed").
6. The DUT sends the SetRelayOutputSettingsResponse message.
7. ONVIF Client invokes SetRelayOutputStateRequest message (**RelayOutputToken, LogicalState="active"**).
8. Verify the SetRelayOutputStateResponse message from the DUT.
9. Wait until time out time1 expires.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetRelayOutputsResponse message.

The DUT did not send SetRelayOutputSettingsResponse message.

The DUT did not send valid SetRelayOutputSettingsResponse message.

The DUT did not send SetRelayOutputStateResponse message.

6.4.18 IO COMMAND SETRELAYOUTPUTSTATE – MONOSTABLE MODE (INACTIVE BEFORE DELAYTIME EXPIRED)

Test Label: Device Management SetRelayOutputState Command Validation in the Case of Monostable Mode (Inactive before DelayTime Expired).

Test Case ID: DEVICE-5-1-9

ONVIF Core Specification Coverage: Trigger relay output

Command under Test: SetRelayOutputState

WSDL Reference: devicemgmt.wsdl

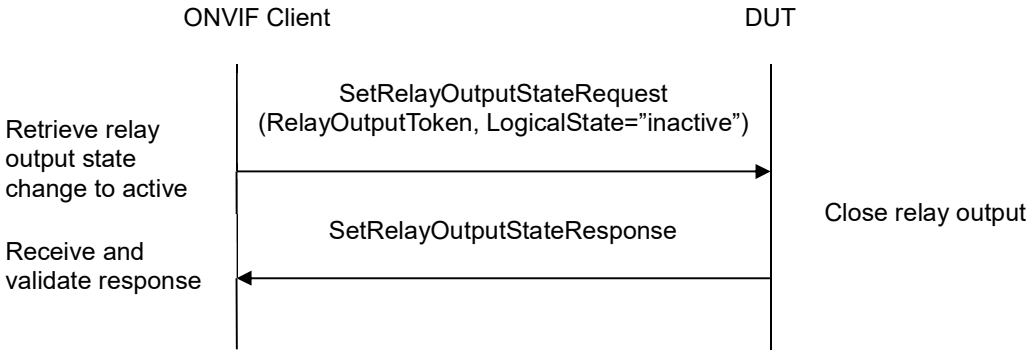
Test Purpose: To verify the behavior of SetRelayOutputState command in case of monostable mode for inactive state.

Pre-Requisite: Relay Outputs are supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
GetRelayOutputsRequest
Retrieve relay outputs
Send relay output list
GetRelayOutputsResponse (relay output list)
Receive and validate relay output list
SetRelayOutputSettingsRequest (RelayOutputToken, Mode="Monostable", DelayTime=time1, IdleState="open")
Retrieve changing of relay output settings
Modify relay output settings
SetRelayOutputSettingsResponse
Receive and validate response
SetRelayOutputStateRequest (RelayOutputToken, LogicalState="active")
Retrieve relay output state change to active
Close relay output
SetRelayOutputStateResponse
Receive and validate response
SetRelayOutputStateRequest (RelayOutputToken, LogicalState="inactive")
Retrieve relay output state change to active
Open relay output
SetRelayOutputStateResponse
Receive and validate response
SetRelayOutputSettingsRequest (RelayOutputToken, Mode="Monostable", DelayTime=time1, IdleState="closed")
Retrieve changing of relay output settings
Modify relay output settings
SetRelayOutputSettingsResponse
Receive and validate response
SetRelayOutputStateRequest (RelayOutputToken, LogicalState="active")
Retrieve relay output state change to active
Open relay output
SetRelayOutputStateResponse
Receive and validate response



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. If the DUT supports Monostable Mode with Open Idle state, ONVIF Client invokes SetRelayOutputSettingsRequest message (**RelayOutputToken, Mode = "Monostable", DelayTime=time1, IdleState="open"**) to set new relay output settings. If the DUT does not support Monostable Mode with Open Idle state, then go to the step 12.
6. The DUT sends the SetRelayOutputSettingsResponse message.
7. ONVIF Client invokes SetRelayOutputStateRequest message (**RelayOutputToken, LogicalState="active"**).
8. Verify the SetRelayOutputStateResponse message from the DUT.
9. ONVIF Client invokes SetRelayOutputStateRequest message (**RelayOutputToken, LogicalState="inactive"**), when time1 is not expired yet.
10. Verify the SetRelayOutputStateResponse message from the DUT.
11. Wait until timeout time1 expires.
12. If the DUT supports Monostable Mode with Closed Idle state, ONVIF Client invokes SetRelayOutputSettingsRequest message (**RelayOutputToken, Mode = "Monostable", DelayTime=time1, IdleState="closed"**) to set new relay output settings. If the DUT does not support Monostable Mode with Open Idle state, then skip other steps of the test.
13. The DUT sends the SetRelayOutputSettingsResponse message.
14. ONVIF Client invokes SetRelayOutputStateRequest message (**RelayOutputToken, LogicalState="active"**).
15. Verify the SetRelayOutputStateResponse message from the DUT.
16. ONVIF Client invokes SetRelayOutputStateRequest message (**RelayOutputToken, LogicalState="inactive"**), when time1 is not expired yet.

17. Verify the SetRelayOutputStateResponse message from the DUT.
18. Wait until time out time1 will expire.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetRelayOutputsResponse message.

The DUT did not send SetRelayOutputSettingsResponse message.

The DUT did not send valid SetRelayOutputSettingsResponse message.

The DUT did not send SetRelayOutputStateResponse message.

The DUT did not change relay state as expected.

6.4.19 IO COMMAND SETRELAYOUTPUTSETTINGS – INVALID TOKEN

Test Label: Device Management DUT Input/Output (I/O) Command SetRelayOutputSettings Test (Invalid Token).

Test Case ID: DEVICE-5-1-11

ONVIF Core Specification Coverage: Set relay output settings

Command under Test: SetRelayOutputSettings

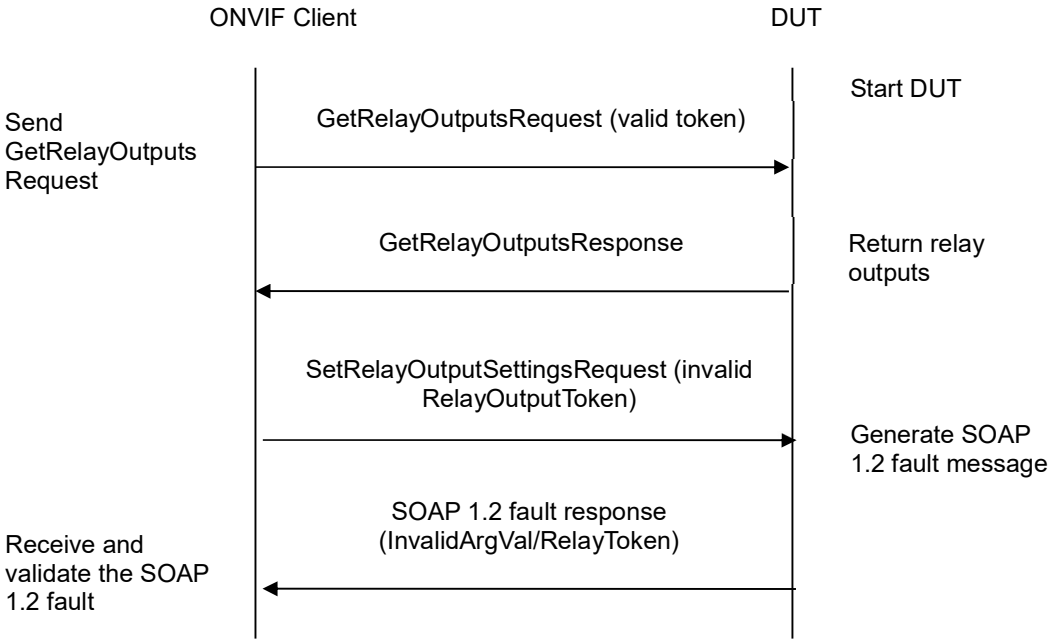
WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: 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 sends a GetRelayOutputsRequest to retrieve the available relay outputs.
4. The DUT returns its relay outputs.
5. ONVIF Client invokes SetRelayOutputSettingsRequest message (**invalid RelayOutputToken**).
6. The DUT will generate SOAP 1.2 fault message (**InvalidArgVal/RelayToken**)

Test Result:

PASS –

The DUT passed 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 though the specified are preferable.

Note: See Annex A.18. for Name and Token Parameters Length limitations.

6.4.20 IO COMMAND SETRELAYOUTPUTSTATE – INVALID TOKEN

Test Label: Device Management SetRelayOutputState Command Validation (Invalid Token).

Test Case ID: DEVICE-5-1-12

ONVIF Core Specification Coverage: Trigger relay output

Command under Test: SetRelayOutputState

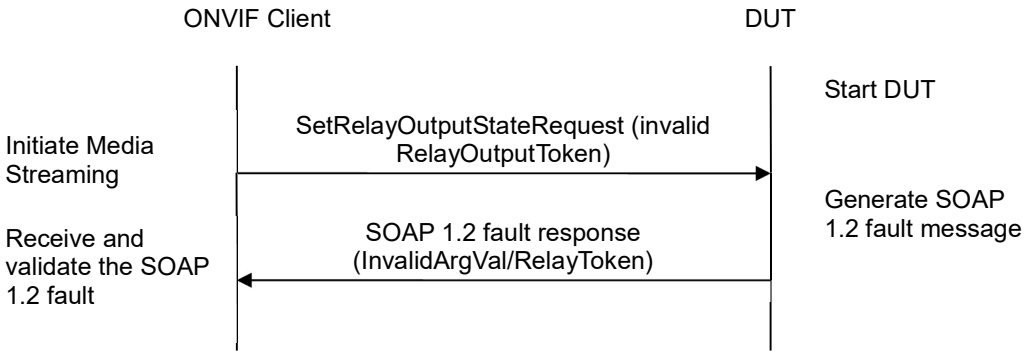
WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: 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 SetRelayOutputStateRequest message (**invalid RelayOutputToken**).
4. The DUT will generate SOAP 1.2 fault message (**InvalidArgVal/RelayToken**).

Test Result:

PASS –

The DUT passed 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 though the specified are preferable.

Note: See Annex A.18 for Name and Token Parameters Length limitations.

6.5 Namespace Handling

6.5.1 DEVICE MANAGEMENT - NAMESPACES (DEFAULT NAMESPACES FOR EACH TAG)

Test Label: Device Management Service Different Namespaces Definition Test (Default Namespaces for Each Tag).

Test Case ID: DEVICE-6-1-1

ONVIF Core Specification Coverage: None

Command under Test: None

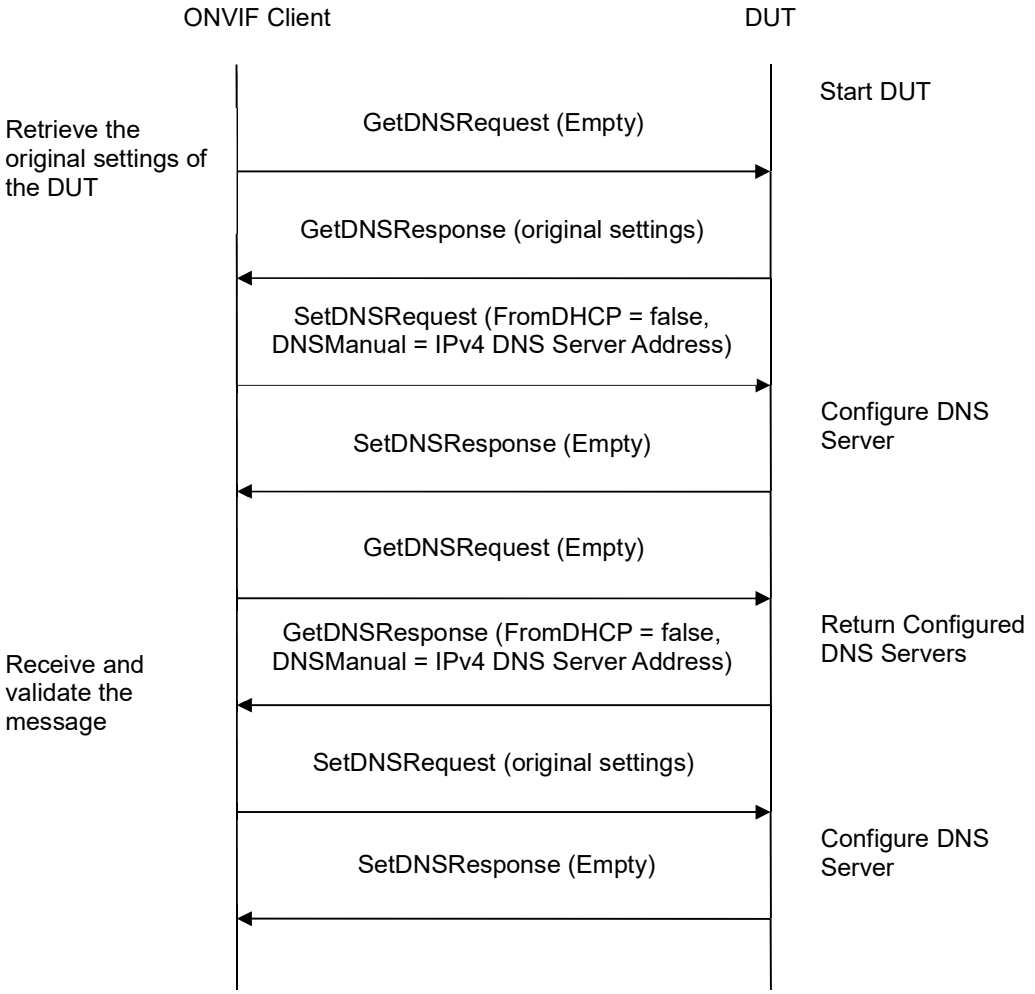
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify that the DUT accepts requests for Device Management Service with different namespaces definition.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetDNSRequest message to retrieve the original settings of the DUT.
4. Verify the GetDNSResponse message from the DUT.
5. ONVIF Client invokes SetDNSRequest message (FromDHCP = false, DNSManual = [“IPv4”, “DNS IP”]) to retrieve the original settings of the DUT.
6. Verify the SetDNSResponse message from the DUT.
7. ONVIF Client invokes GetDNSRequest message to retrieve the settings of the DUT.
8. Verify the GetDNSResponse message from the DUT and verify DNS configurations in the DUT.
9. ONVIF Client will invoke SetDNS Request message to restore the original settings of the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetDNSResponse message.

The DUT did not send SetDNSResponse message.

The DUT did not send the correct information in the GetDNSResponse message (i.e. FromDHCP = false, DNSManual = [“IPv4”, “DNS IP”]).

Note: All requests to the DUT shall have default namespaces definition in each tag (see examples in Annex A.11).

6.5.2 DEVICE MANAGEMENT - NAMESPACES (DEFAULT NAMESPACES FOR PARENT TAG)

Test Label: Device Management Service Different Namespaces Definition Test (Default Namespaces for Parent Tag).

Test Case ID: DEVICE-6-1-2

ONVIF Core Specification Coverage: None

Command under Test: None

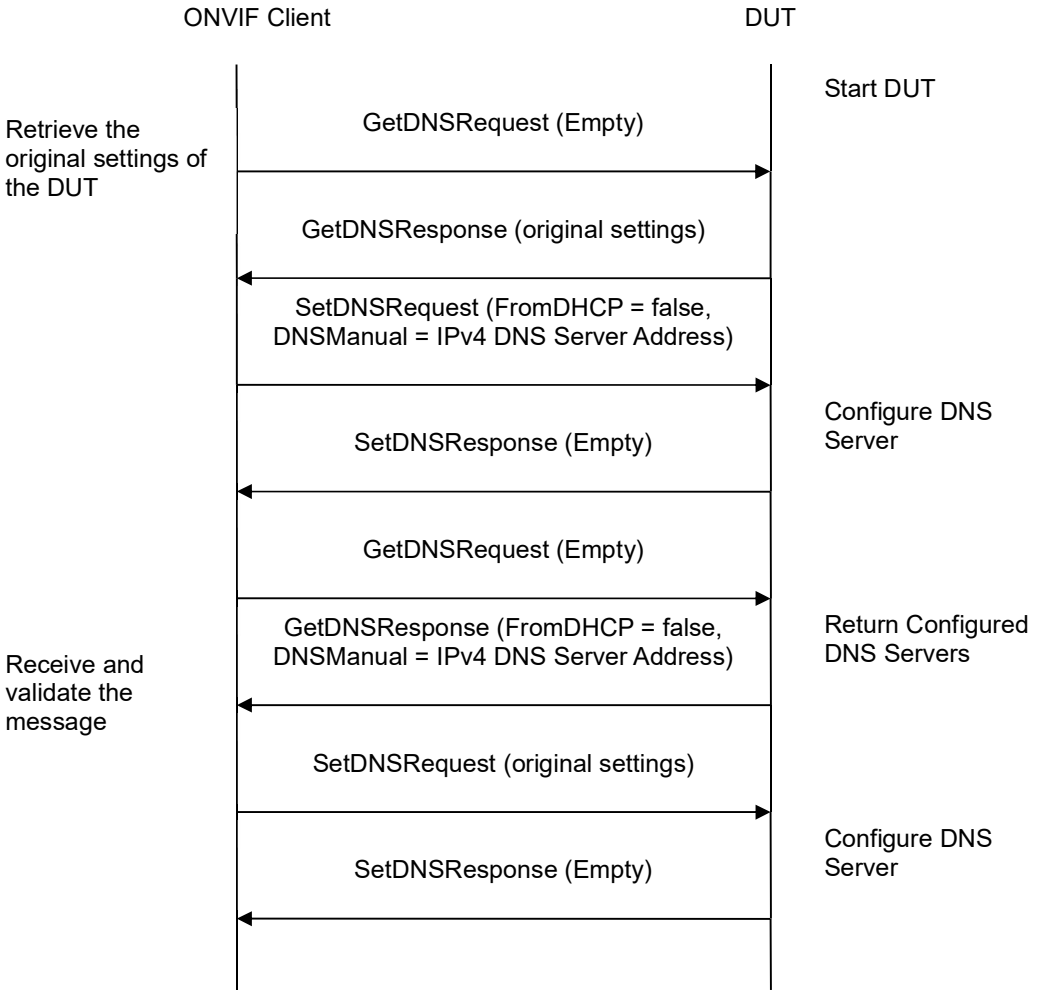
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify that the DUT accepts requests for Device Management Service with different namespaces definition.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetDNSRequest message to retrieve the original settings of the DUT.
4. Verify the GetDNSResponse message from the DUT.
5. ONVIF Client invokes SetDNSRequest message (FromDHCP = false, DNSManual = [“IPv4”, “DNS IP”]) to retrieve the original settings of the DUT.
6. Verify the SetDNSResponse message from the DUT.
7. ONVIF Client invokes GetDNSRequest message to retrieve the settings of the DUT.
8. Verify the GetDNSResponse message from the DUT and verify DNS configurations in the DUT.
9. ONVIF Client will invoke SetDNS Request message to restore the original settings of the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetDNSResponse message.

The DUT did not send SetDNSResponse message.

The DUT did not send correct information in the GetDNSResponse message (i.e. FromDHCP = false, DNSManual = [“IPv4”, “DNS IP”]).

Note: All requests to the DUT shall have default namespaces definition in parent tag (see examples in Annex A.11).

6.5.3 DEVICE MANAGEMENT - NAMESPACES (NOT STANDARD PREFIXES)

Test Label: Device Management Service Different Namespaces Definition Test (Not Standard Prefixes).

Test Case ID: DEVICE-6-1-3

ONVIF Core Specification Coverage: None

Command under Test: None

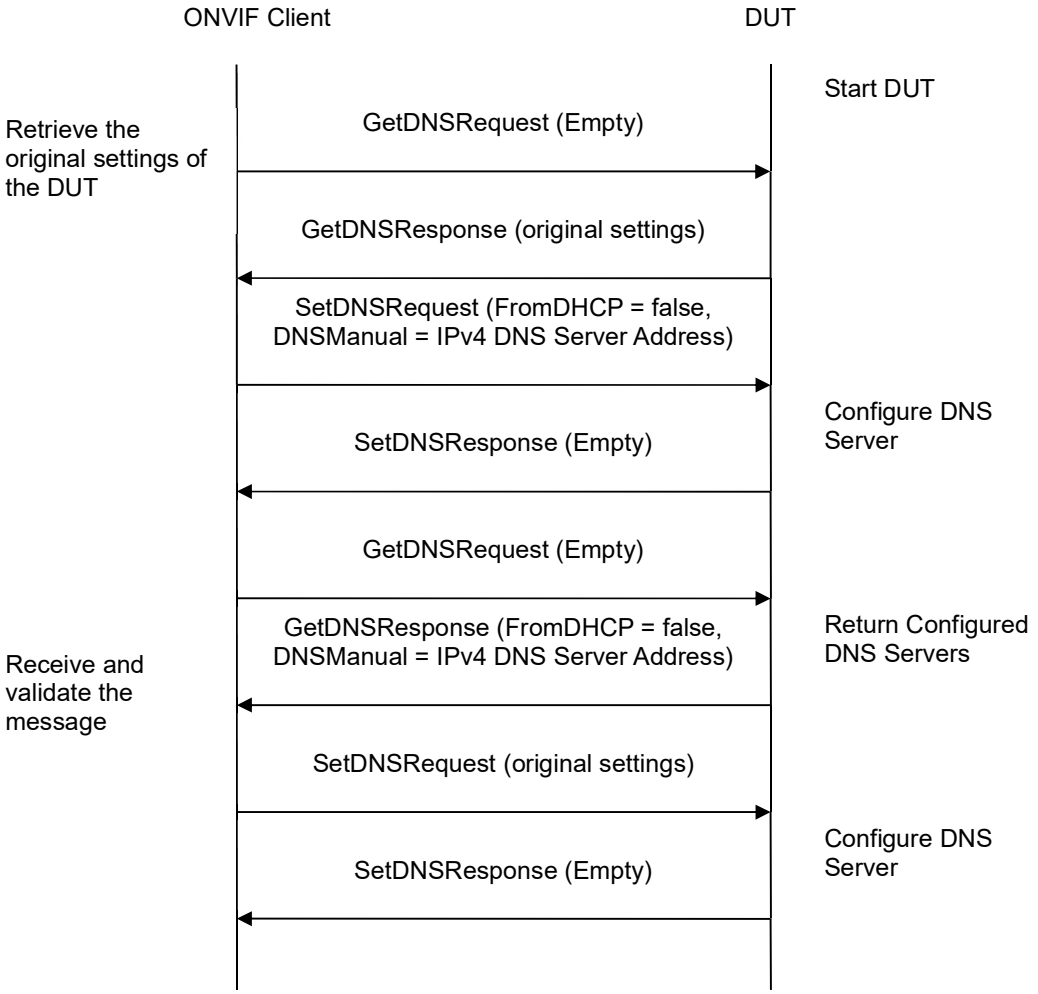
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify that the DUT accepts requests for Device Management Service with different namespaces definition.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetDNSRequest message to retrieve the original settings of the DUT.
4. Verify the GetDNSResponse message from the DUT.
5. ONVIF Client invokes SetDNSRequest message (FromDHCP = false, DNSManual = [“IPv4”, “DNS IP”]) to retrieve the original settings of the DUT.
6. Verify the SetDNSResponse message from the DUT.
7. ONVIF Client invokes GetDNSRequest message to retrieve the settings of the DUT.
8. Verify the GetDNSResponse message from the DUT and verify DNS configurations in the DUT.
9. ONVIF Client will invoke SetDNS Request message to restore the original settings of the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetDNSResponse message.

The DUT did not send SetDNSResponse message.

The DUT did not send correct information in the GetDNSResponse message (i.e. FromDHCP = false, DNSManual = [“IPv4”, “DNS IP”]).

Note: All requests to the DUT shall have namespaces definition with not standard prefixes (see examples in Annex A.11).

6.5.4 DEVICE MANAGEMENT - NAMESPACES (DIFFERENT PREFIXES FOR THE SAME NAMESPACE)

Test Label: Device Management Service Different Namespaces Definition Test (Different Prefixes for the Same Namespace).

Test Case ID: DEVICE-6-1-4

ONVIF Core Specification Coverage: None

Command under Test: None

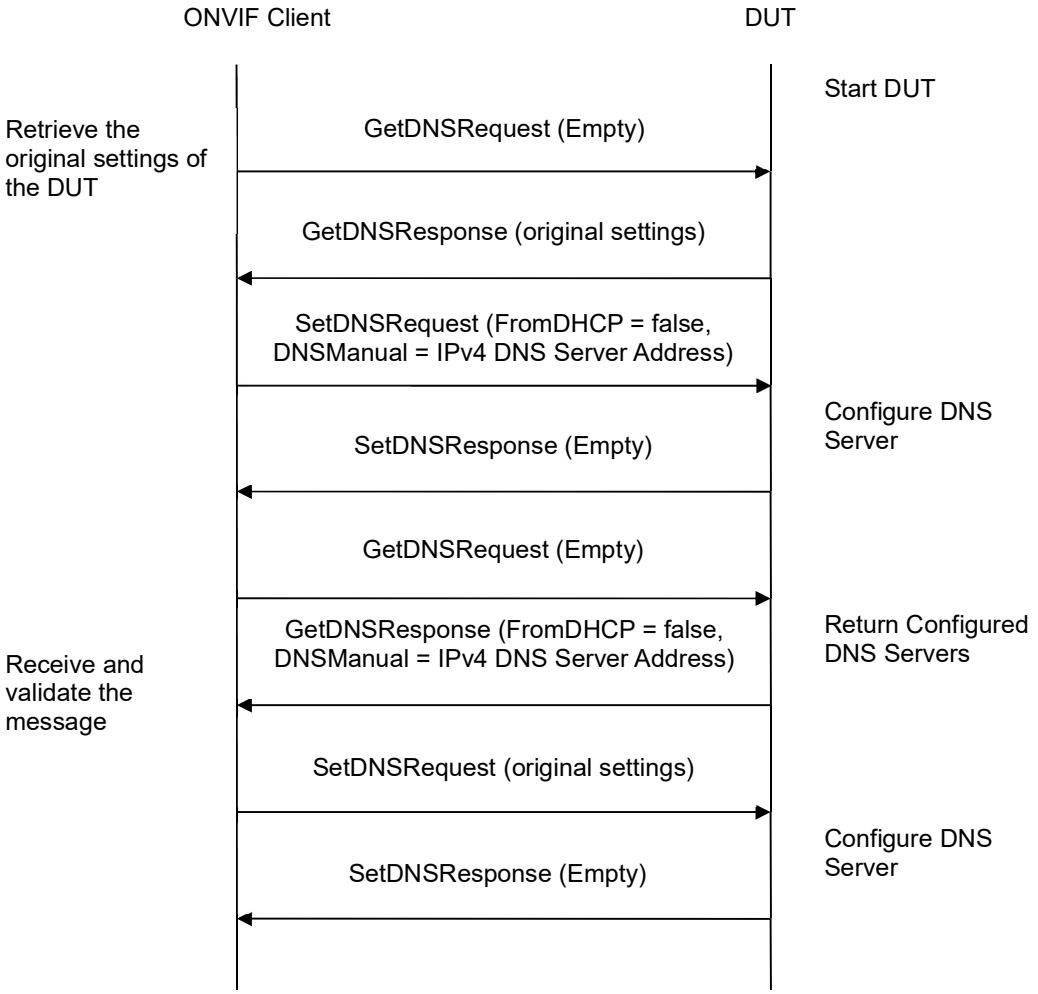
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify that the DUT accepts requests for Device Management Service with different namespaces definition.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetDNSRequest message to retrieve the original settings of the DUT.
4. Verify the GetDNSResponse message from the DUT.
5. ONVIF Client invokes SetDNSRequest message (FromDHCP = false, DNSManual = [“IPv4”, “DNS IP”]) to retrieve the original settings of the DUT.
6. Verify the SetDNSResponse message from the DUT.
7. ONVIF Client invokes GetDNSRequest message to retrieve the settings of the DUT.
8. Verify the GetDNSResponse message from the DUT and verify DNS configurations in the DUT.
9. ONVIF Client will invoke SetDNS Request message to restore the original settings of the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetDNSResponse message.

The DUT did not send SetDNSResponse message.

The DUT did not send correct information in the GetDNSResponse message (i.e. FromDHCP = false, DNSManual = [“IPv4”, “DNS IP”]).

Note: All requests to the DUT shall have namespaces definition with different prefixes for the same namespace (see examples in Annex A.11).

6.5.5 DEVICE MANAGEMENT - NAMESPACES (THE SAME PREFIX FOR DIFFERENT NAMESPACES)

Test Label: Device Management Service Different Namespaces Definition Test (the Same Prefix for Different Namespaces).

Test Case ID: DEVICE-6-1-5

ONVIF Core Specification Coverage: None

Command under Test: None

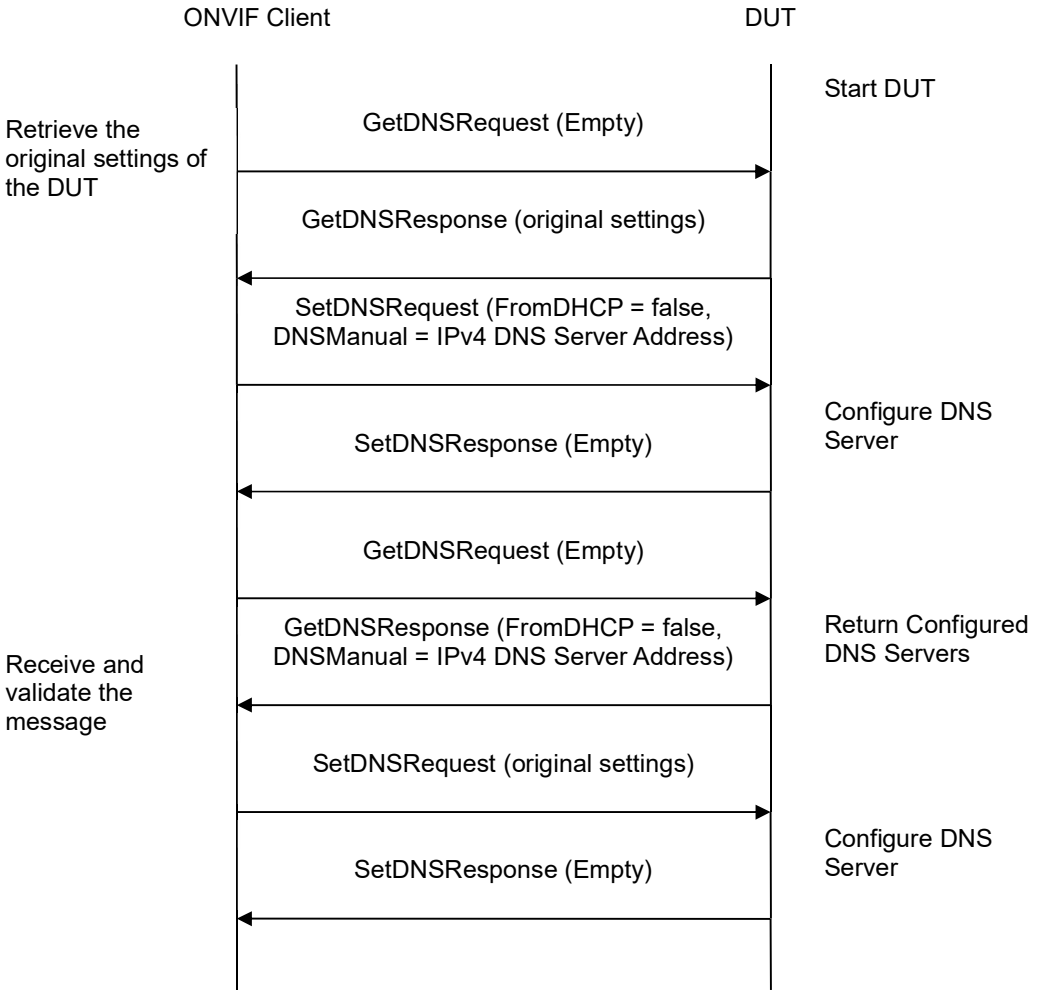
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify that the DUT accepts requests for Device Management Service with different namespaces definition.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetDNSRequest message to retrieve the original settings of the DUT.
4. Verify the GetDNSResponse message from the DUT.
5. ONVIF Client invokes SetDNSRequest message (FromDHCP = false, DNSManual = [“IPv4”, “DNS IP”]) to retrieve the original settings of the DUT.
6. Verify the SetDNSResponse message from the DUT.
7. ONVIF Client invokes GetDNSRequest message to retrieve the settings of the DUT.
8. Verify the GetDNSResponse message from the DUT and verify DNS configurations in the DUT.
9. ONVIF Client will invoke SetDNS Request message to restore the original settings of the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetDNSResponse message.

The DUT did not send SetDNSResponse message.

The DUT did not send correct information in the GetDNSResponse message (i.e. FromDHCP = false, DNSManual = [“IPv4”, “DNS IP”]).

Note: All requests to the DUT shall have namespaces definition with the same prefixes for different namespaces (see examples in Annex A.11).

6.6 IP Filtering test cases

6.6.1 GET IP ADDRESS FILTER

Test Label: Device Management Network Command GetIPAddressFilter Test.

Test Case ID: DEVICE-7-1-1

ONVIF Core Specification Coverage: Get IP address filter

Command Under Test: GetIPAddressFilter

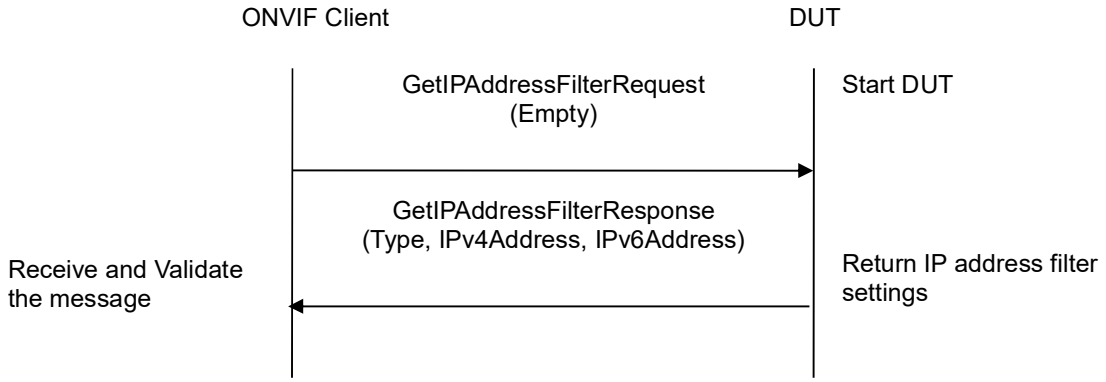
WSDL Reference: devicemgmt.wsdl

Test Purpose: To retrieve IP address filter settings for the DUT using GetIPAddressFilter command.

Pre-Requisite: The device supports device access control based on IP filtering rules (denied or accepted ranges of IP addresses)

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetIPAddressFilterRequest message to retrieve IP address filter settings of the DUT.
4. Verify the GetIPAddressFilterResponse from the DUT (IPAddressFilter [Type = 'Allow' or 'Deny', IPv4Address = list of IPv4 filter addresses, IPv6Address = list of IPv6 filter addresses]).
5. If the DUT returned IPv4Address, validate IPv4Address.Address and IPv4Address.PrefixLength. Check that IPv4Address.Address represented at dot-decimal notation format. Check that 0>=IPv4Address.PrefixLength>=32.
6. Check that IPv4Address.Address along with IPv4Address.PrefixLength is a valid range of IPv4 addresses according to Classless Inter-Domain Routing (CIDR) method.

7. If the DUT returned IPv6Address, validate IPv6Address.Address and IPv6Address.PrefixLength. Check that IPv6Address.Address represented as eight groups of four hexadecimal digits (or simplified). Check that 0>=IPv6Address.PrefixLength>=128.

8. Check that IPv6Address.Address along with IPv6Address.PrefixLength is a valid range of IPv4 addresses according to Classless Inter-Domain Routing (CIDR) method.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetIPAddressFilterResponse message.

The DUT did not send correct information (i.e. IPAddressFilter [Type = 'Allow' or 'Deny', IPv4Address = list of IPv4 filter addresses, IPv6Address = list of IPv6 filter addresses]) in the GetIPAddressFilterResponse message.

The DUT did not send correct IPv4Address (IPv4Address.Address represented at dot-decimal notation format, 0>=IPv4Address.PrefixLength>=32)

The DUT did not send correct IPv4Address range.

The DUT did not send correct IPv6Address (IPv6Address.Address represented as eight groups of four hexadecimal digits (or simplified, 0>=IPv6Address.PrefixLength>=128)

The DUT did not send correct IPv6Address range.

Note: See Annex A.10 for valid expression in terms of empty IP address.

6.6.2 SET IP ADDRESS FILTER – IPv4

Test Label: Device Management Network Command SetIPAddressFilter Test.

Test Case ID: DEVICE-7-1-2

ONVIF Core Specification Coverage: Set IP address filter

Command Under Test: SetIPAddressFilter

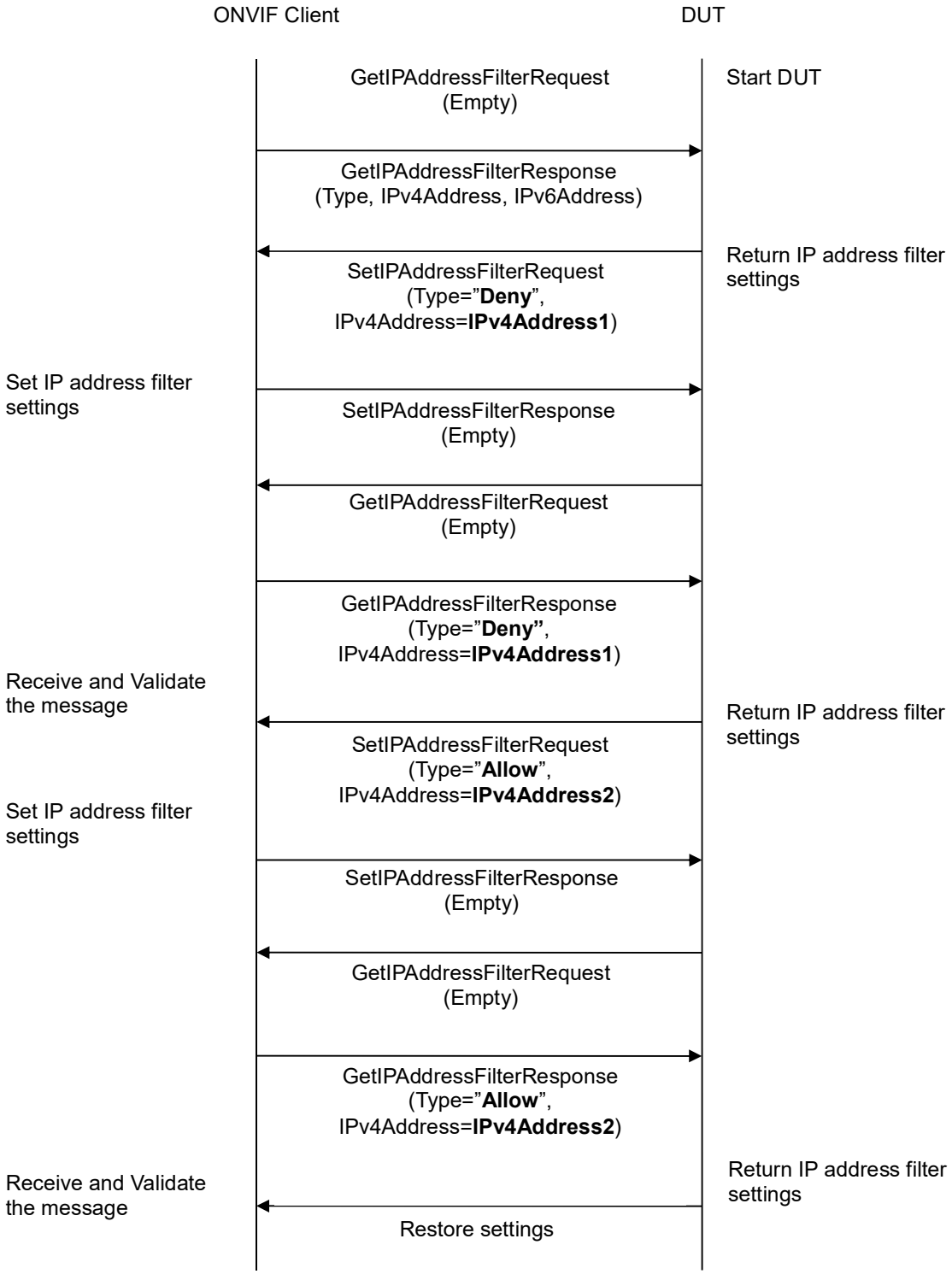
WSDL Reference: devicemgmt.wsdl

Test Purpose: To configure IP address filter settings for the DUT using SetIPAddressFilter command.

Pre-Requisite: The device supports device access control based on IP filtering rules (denied or accepted ranges of IP addresses)

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

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

3. ONVIF Client will invoke GetIPAddressFilterRequest message to retrieve the original settings of the DUT.
4. Verify GetIPAddressFilterResponse message from the DUT.
5. ONVIF Client will invoke SetIPAddressFilterRequest message (IPAddressFilter [Type = **Deny**, IPv4Address = IPv4Address1]).
6. Verify that the DUT sends SetIPAddressFilterResponse (empty message)
7. ONVIF Client will invoke GetIPAddressFilterRequest message to retrieve updated settings of the DUT.
8. Verify GetIPAddressFilterResponse message from the DUT. Check that the settings were applied.
9. ONVIF Client will invoke SetIPAddressFilterRequest message (IPAddressFilter [Type = **Allow**, IPv4Address = IPv4Address2]).
10. Verify that the DUT sends SetIPAddressFilterResponse (empty message).
11. ONVIF Client will invoke GetIPAddressFilterRequest message to retrieve updated settings of DUT.
12. Verify GetIPAddressFilterResponse message from the DUT. Check that the settings were applied.
13. ONVIF Client restores the original settings of DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetIPAddressFilterResponse message.

The DUT did not send SetIPAddressFilterResponse message.

The DUT did not apply new settings after sending SetIPAddressFilter command.

Note: IPv4Address1 shall be different from the current IP address of ONVIF Client.

Note: IPv4Address2 shall be the same with current IP address of ONVIF Client.

6.6.3 ADD IP ADDRESS FILTER – IPv4

Test Label: Device Management Network Command AddIPAddressFilter Test.

Test Case ID: DEVICE-7-1-3

ONVIF Core Specification Coverage: Add IP address to filter

Command Under Test: AddIPAddressFilter

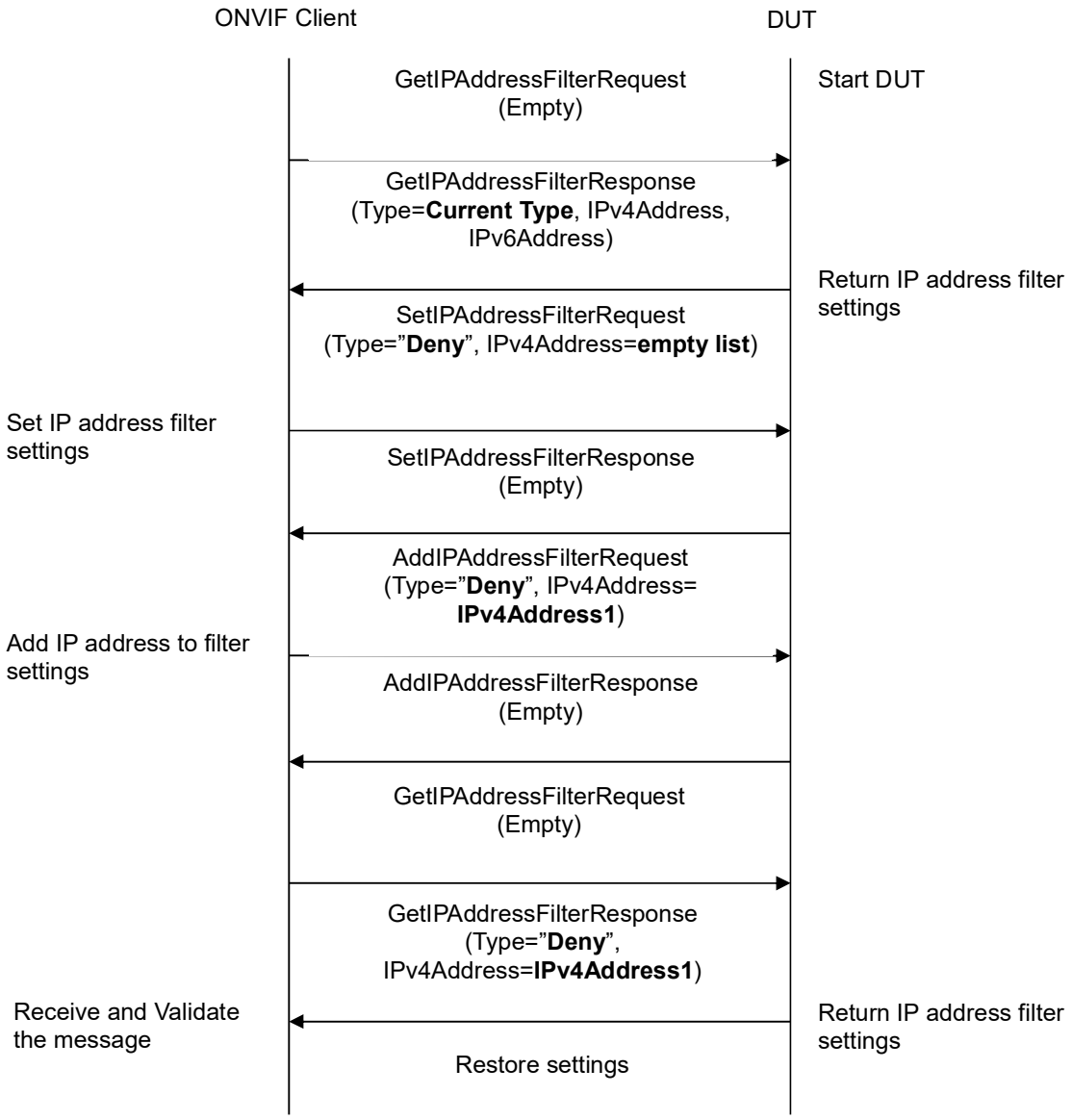
WSDL Reference: devicemgmt.wsdl

Test Purpose: To add IP filter address to the DUT using AddIPAddressFilter command.

Pre-Requisite: The device supports device access control based on IP filtering rules (denied or accepted ranges of IP addresses)

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

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

3. ONVIF Client will invoke GetIPAddressFilterRequest message to retrieve IP address filter settings of the DUT.
4. Verify the GetIPAddressFilterResponse from the DUT (IPAddressFilter [Type = **CurrentType** ('Allow' or 'Deny'), IPv4Address = list of IPv4 filter addresses, IPv6Address = list of IPv6 filter addresses]).
5. ONVIF Client will invoke SetIPAddressFilterRequest message (IPAddressFilter [Type = **Deny**, IPv4Address empty list]).
6. Verify that the DUT sends SetIPAddressFilterResponse (empty message)
7. ONVIF Client will invoke AddIPAddressFilterRequest (IPAddressFilter [Type = **Deny**, IPv4Address = IPv4Address1]) message to add IP filter address to the DUT.
8. Verify that the DUT sends AddIPAddressFilterResponse message.
9. ONVIF Client will invoke GetIPAddressFilterRequest message to retrieve updated IP address filter settings of the DUT.
10. Verify the GetIPAddressFilterResponse from DUT (IPAddressFilter [Type = **Deny**, IPv4Address = IPv4Address1]).
11. ONVIF Client restores the original settings of the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetIPAddressFilterResponse message.

The DUT did not send AddIPAddressFilterResponse message.

The DUT did not return a new IP address at step 9.

Note: IPv4Address1 shall be different from the current IP address of ONVIF Client.

6.6.4 REMOVE IP ADDRESS FILTER – IPv4

Test Label: Device Management Network Command RemoveIPAddressFilter Test.

Test Case ID: DEVICE-7-1-4

ONVIF Core Specification Coverage: Remove IP address from filter

Command Under Test: RemoveIPAddressFilter

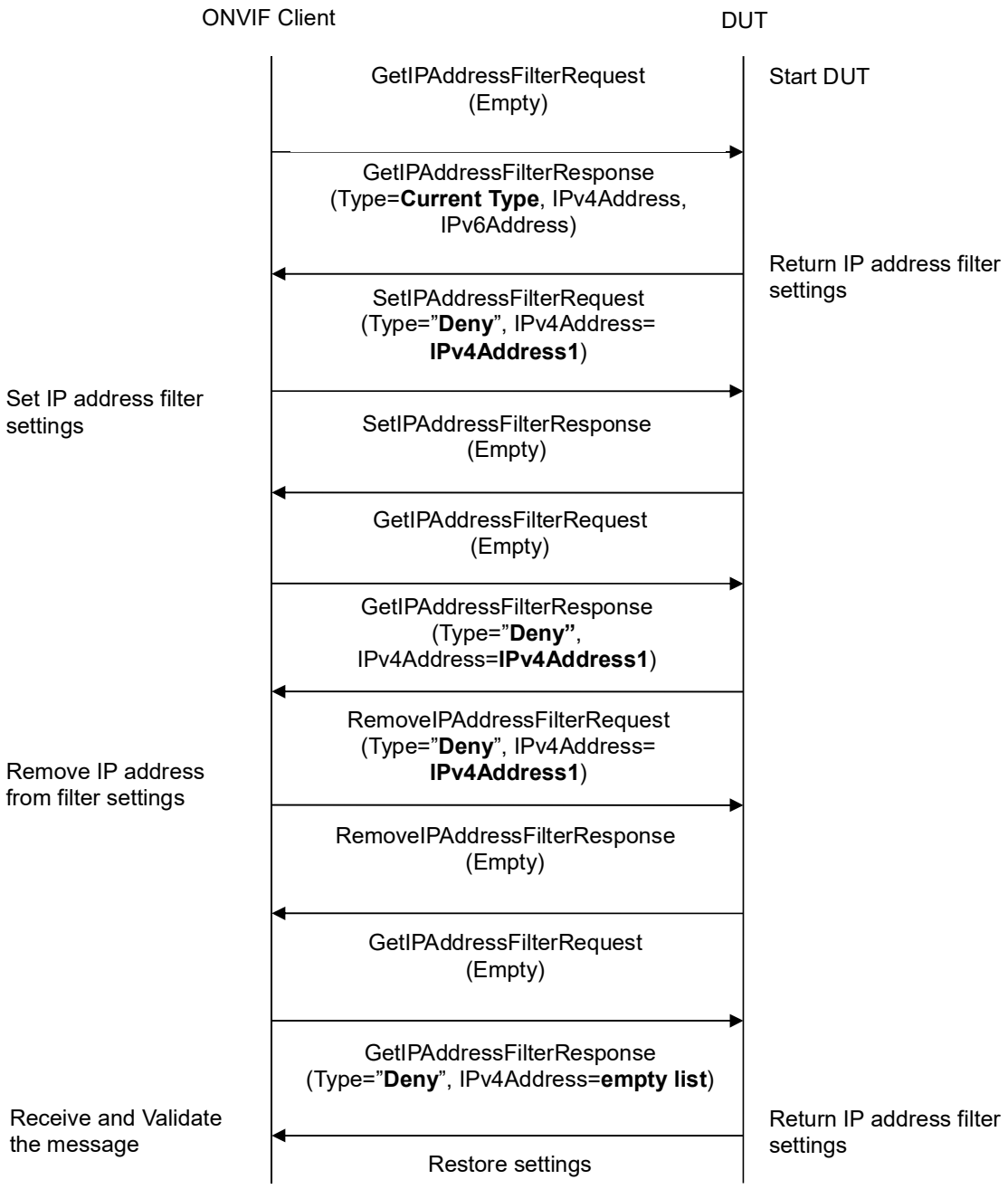
WSDL Reference: devicemgmt.wsdl

Test Purpose: To remove IP filter address from the DUT using RemoveIPAddressFilter command.

Pre-Requisite: The device supports device access control based on IP filtering rules (denied or accepted ranges of IP addresses)

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetIPAddressFilterRequest message to retrieve IP address filter settings of the DUT.

4. Verify the GetIPAddressFilterResponse from the DUT (IPAddressFilter [Type = **CurrentType** ('Allow' or 'Deny'), IPv4Address = list of IPv4 filter addresses, IPv6Address = list of IPv6 filter addresses]).
5. ONVIF Client will invoke SetIPAddressFilterRequest message (IPAddressFilter [Type = **Deny**, IPv4Address = IPv4Address1]).
6. Verify that the DUT sends SetIPAddressFilterResponse (empty message).
7. ONVIF Client will invoke GetIPAddressFilterRequest message to retrieve updated settings of the DUT.
8. Verify GetIPAddressFilterResponse message from the DUT. Check that the settings were applied.
9. ONVIF Client will invoke RemoveIPAddressFilterRequest (IPAddressFilter [Type = Allow, IPv4Address = IPv4Address1]) to remove IP filter address in the list.
10. Verify that the DUT sends RemoveIPAddressFilterResponse (empty message).
11. ONVIF Client will invoke GetIPAddressFilterRequest message to retrieve updated IP address filter settings of the DUT.
12. Verify the GetIPAddressFilterResponse from the DUT (IPAddressFilter [Type = **Deny**, IPv4Address = empty list]).
13. ONVIF Client restores the original settings of the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SetIPAddressFilterResponse message.

The DUT did not send GetIPAddressFilterResponse message.

The DUT did not send RemoveIPAddressFilterResponse message.

The DUT did not send an empty IPv4Address list at step 12.

Note: IPv4Address1 shall be different from the current IP address of ONVIF Client.

6.7 Auxiliary operation

6.7.1 AUXILIARY COMMANDS

Test Label: Auxiliary Commands.

Test Case ID: DEVICE-8-1-1

ONVIF Core Specification Coverage: None

Command under Test: None

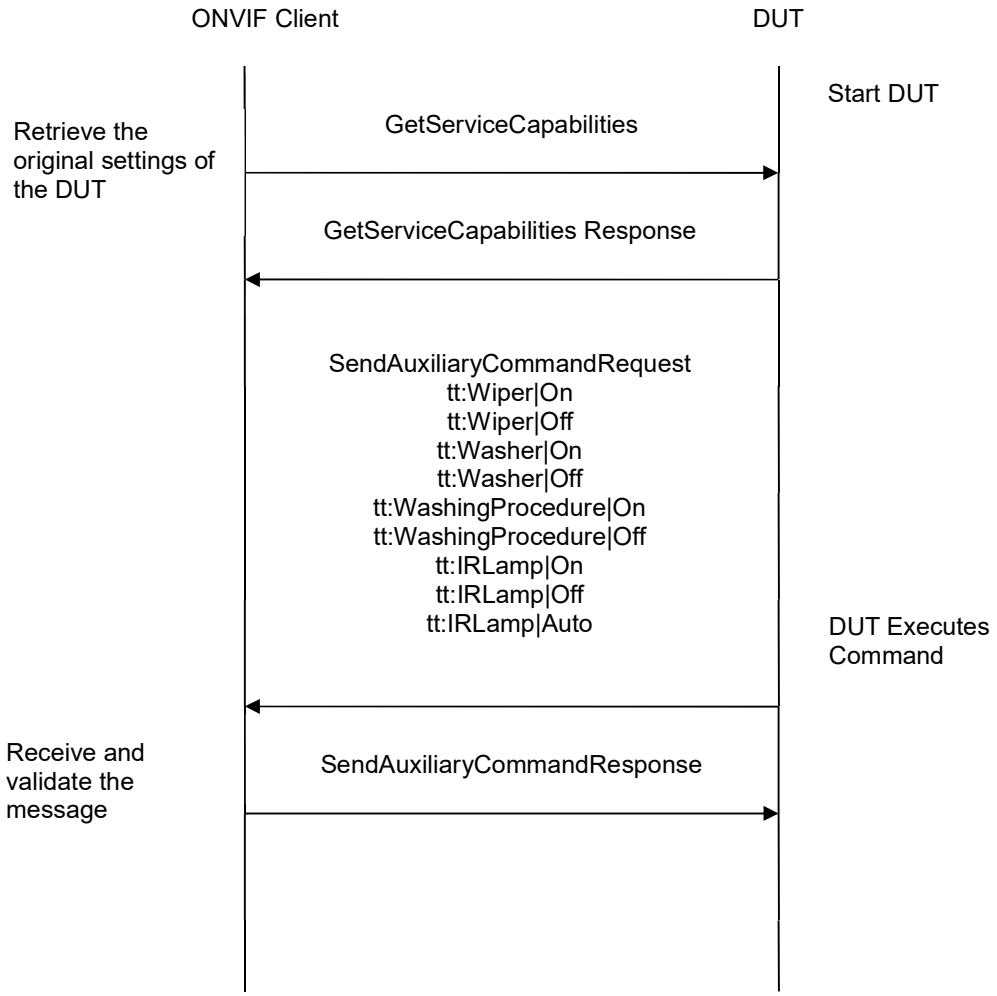
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify that the DUT accepts requests for Device Management Service with different namespaces definition.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetServiceCapabilities message to retrieve the original settings of Device Service.
4. Verify the response message from the DUT. Identify supported commands stored in the GetServiceCapabilitiesResponse Misc->AuxiliaryCommands list.

5. For each supported Auxiliary Command

   (tt:Wiper|On

   tt:Wiper|Off

   tt:Washer|On

   tt:Washer|Off

   tt:WashingProcedure|On

   tt:WashingProcedure|Off

   tt:IRLamp|On

   tt:IRLamp|Off

   tt:IRLamp|Auto)

   execute SendAuxiliaryCommand Request

6. Verify the response message from the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send proper SendAuxiliaryCommandResponce.

6.8 Monitoring Events

6.8.1 Processor Usage event

Test Label: Processor Usage event

Test Case ID: DEVICE-9-1-1

ONVIF Core Specification Coverage: Monitoring Event ProcessorUsage

Command Under Test: GetServices, GetEventProperties, CreatePullPointSubscription, PullMessages, Monitoring/ProcessorUsage event

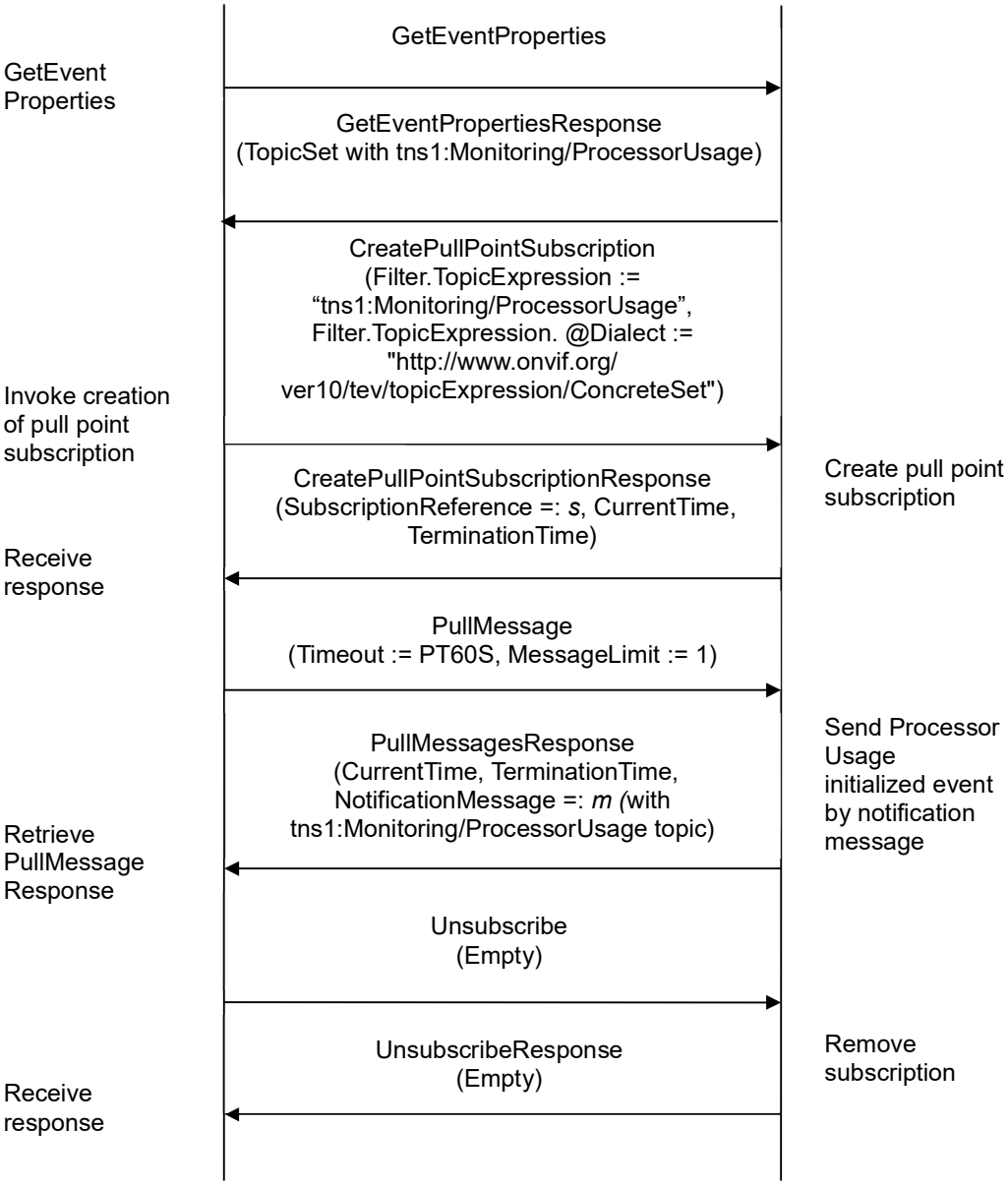
WSDL Reference: devicemgmt.wsdl and event.wsdl

Test Purpose: To verify tns1:Monitoring/ProcessorUsage event generation after subscription and to verify tns1:Monitoring/ProcessorUsage event format.

Pre-requisite: Event Service was received from the DUT. tns1:Monitoring/ProcessorUsage event is supported by the DUT as indicated by the GetEventPropertiesResponse.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list

- FixedTopicSet
- TopicSet =: *topicSet*
- TopicExpressionDialect list
- MessageContentFilterDialect list
- MessageContentSchemaLocation list

5. If *topicSet* does not contain tns1:Monitoring/ProcessorUsage topic, FAIL the test and skip other steps.
6. ONVIF Client verifies tns1:Monitoring/ProcessorUsage topic (*processorUsageTopic*) from *topicSet*:
   6.1. If *processorUsageTopic*.MessageDescription.IsProperty is skipped or equals false, FAIL the test and skip other steps.
   6.2. If *processorUsageTopic* does not contain MessageDescription.Source.SimpleItemDescription item with Name = “Token”, FAIL the test and skip other steps.
   6.3. If *processorUsageTopic.*MessageDescription.Source.SimpleItemDescription with Name = “Token” does not have Type = “tt:ReferenceToken”, FAIL the test and skip other steps.
   6.4. If *processorUsageTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = “Value”, FAIL the test and skip other steps.
   6.5. If *processorUsageTopic*.MessageDescription.Data.SimpleItemDescription item with Name = “Value” does not have Type = “xs:float”, FAIL the test and skip other steps.
7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
   - Filter.TopicExpression := “tns1:Monitoring/ProcessorUsage”
   - Filter.TopicExpression.@Dialect := “http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet”
8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
   - SubscriptionReference =: *s*
   - CurrentTime
   - TerminationTime
9. Until *timeout1* timeout expires, repeat the following steps:
   9.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
      - Timeout := PT60S
      - MessageLimit := 1
   9.2. The DUT responds with **PullMessagesResponse** message with parameters
      - CurrentTime
      - TerminationTime

- NotificationMessage =: *m*

9.3. If *m* is not null and *m*.Message.Message.PropertyOperation="Initialized" ONVIF Client verifies *m:*

9.3.1. If *m*.Topic does not equal to tns1:Monitoring/ProcessorUsage, FAIL the test and go to the step 10.

9.3.2. If *m* does not contain Message.Message.Source.SimpleItem.Token, FAIL the test and go to the step 10.

9.3.3. If *m*.Message.Message.Source.SimpleItem.Token has value type different from tt:ReferenceToken type, FAIL the test and go to the step 10.

9.3.4. If *m* does not contain Message.Message.Data.SimpleItem.Value, FAIL the test and go to the step 10.

9.3.5. If *m*.Message.Message.Data.SimpleItem.Value has value type different from xs:float type, FAIL the test and go to the step 10.

9.3.6. If *m*.Message.Message.Data.SimpleItem.Value value is outside of range 0 to 100, FAIL the test and go to the step 10.

9.3.7. Go to the step 10.

9.4. If *timeout1* timeout expires for step 9 without Notification with PropertyOperation="Initialized", FAIL the test and go to the step 10.

10. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

11. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

The DUT did not send **CreatePullPointSubscriptionResponse** message.

The DUT did not send **PullMessagesResponse** message(s).

The DUT did not send **UnsubscribeResponse** message.

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

6.8.2 Last Reset event

Test Label: Last Reset event

Test Case ID: DEVICE-9-1-2

ONVIF Core Specification Coverage: Monitoring Event LastReset

Command Under Test: GetServices, GetEventProperties, CreatePullPointSubscription,

PullMessages, Monitoring/OperatingTime/LastReset event

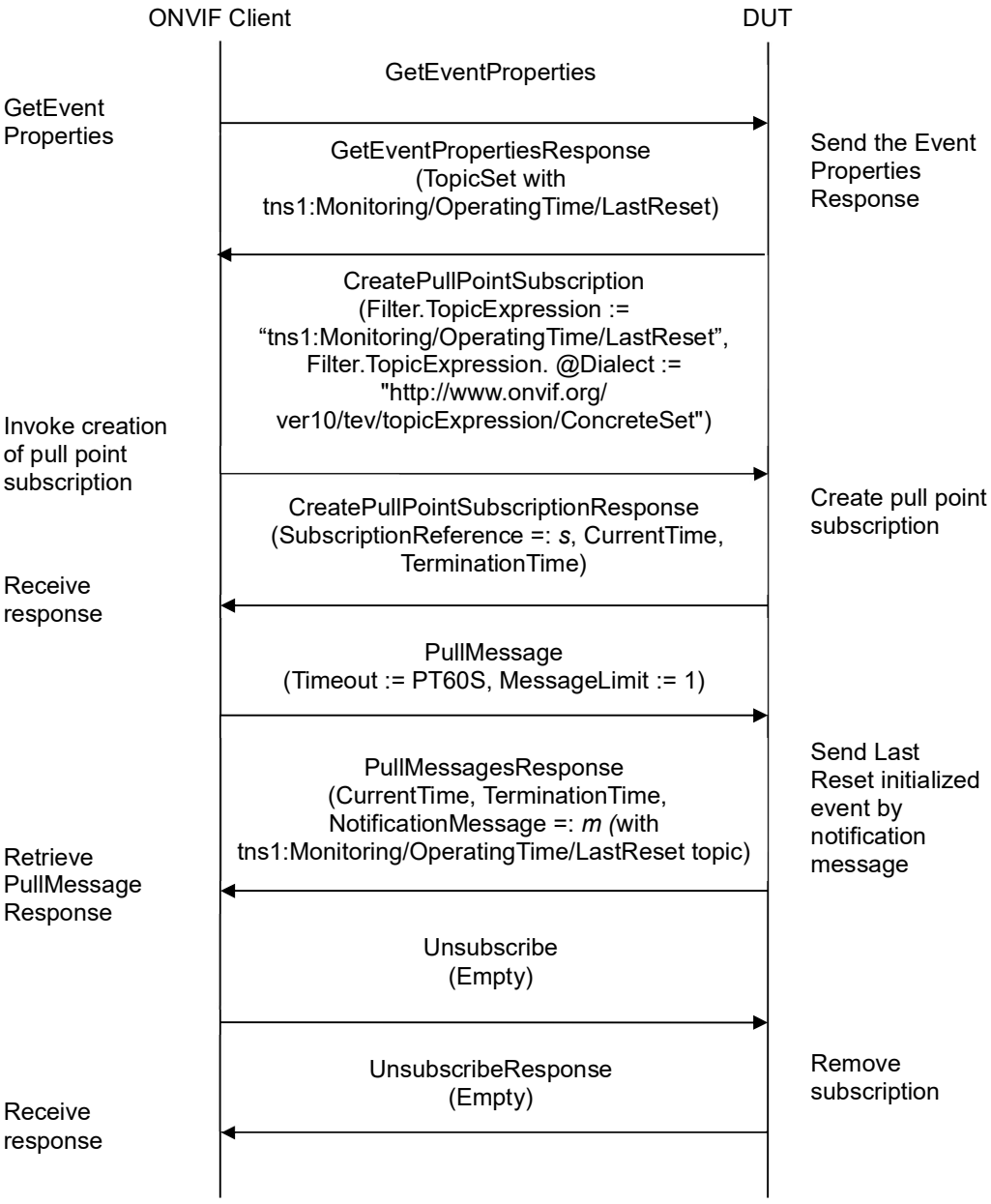
WSDL Reference: devicemgmt.wsdl and event.wsdl

Test Purpose: To verify tns1:Monitoring/OperatingTime/LastReset event generation after subscription and to verify tns1:Monitoring/OperatingTime/LastReset event format.

Pre-requisite: Event Service was received from the DUT. tns1:Monitoring/OperatingTime/LastReset event is supported by the DUT as indicated by the GetEventPropertiesResponse.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters

- TopicNamespaceLocation list
- FixedTopicSet
- TopicSet =: *topicSet*
- TopicExpressionDialect list
- MessageContentFilterDialect list
- MessageContentSchemaLocation list

5. If *topicSet* does not contain tns1:Monitoring/OperatingTime/LastReset topic, FAIL the test and skip other steps.
6. ONVIF Client verifies tns1:Monitoring/OperatingTime/LastReset topic (*lastResetTopic*) from *topicSet*:
   6.1. If *lastResetTopic*.MessageDescription.IsProperty is skipped or equals to false, FAIL the test and skip other steps.
   6.2. If *lastResetTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = “Status”, FAIL the test and skip other steps.
   6.3. If *lastResetTopic*.MessageDescription.Data.SimpleItemDescription with Name = “Status” does not have Type = “xs:dateTime”, FAIL the test and skip other steps.
7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
   - Filter.TopicExpression := “tns1:Monitoring/OperatingTime/LastReset”
   - Filter.TopicExpression.@Dialect := “http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet”
8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
   - SubscriptionReference =: *s*
   - CurrentTime
   - TerminationTime
9. Until *timeout1* timeout expires, repeat the following steps:
   9.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
      - Timeout := PT60S
      - MessageLimit := 1
   9.2. The DUT responds with **PullMessagesResponse** message with parameters
      - CurrentTime
      - TerminationTime
      - NotificationMessage =: *m*
   9.3. If *m* is not null and *m*.Message.Message.PropertyOperation=“Initialized” ONVIF Client verifies

m:

9.3.1. If *m*.Topic does not equal to tns1:Monitoring/OperatingTime/LastReset, FAIL the test and go to the step 10.

9.3.2. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 10.

9.3.3. If *m*.Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 10.

9.3.4. Go to the step 10.

9.4. If *timeout1* timeout expires for step 9 without Notification with PropertyOperation="Initialized", FAIL the test and go to the step 10.

10. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

11. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

The DUT did not send **CreatePullPointSubscriptionResponse** message.

The DUT did not send **PullMessagesResponse** message(s).

The DUT did not send **UnsubscribeResponse** message.

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

6.8.3 Last Reboot event

Test Label: Last Reboot event

Test Case ID: DEVICE-9-1-3

ONVIF Core Specification Coverage: Monitoring Event LastReboot

Command Under Test: GetServices, GetEventProperties, CreatePullPointSubscription, PullMessages, Monitoring/OperatingTime/LastReboot event

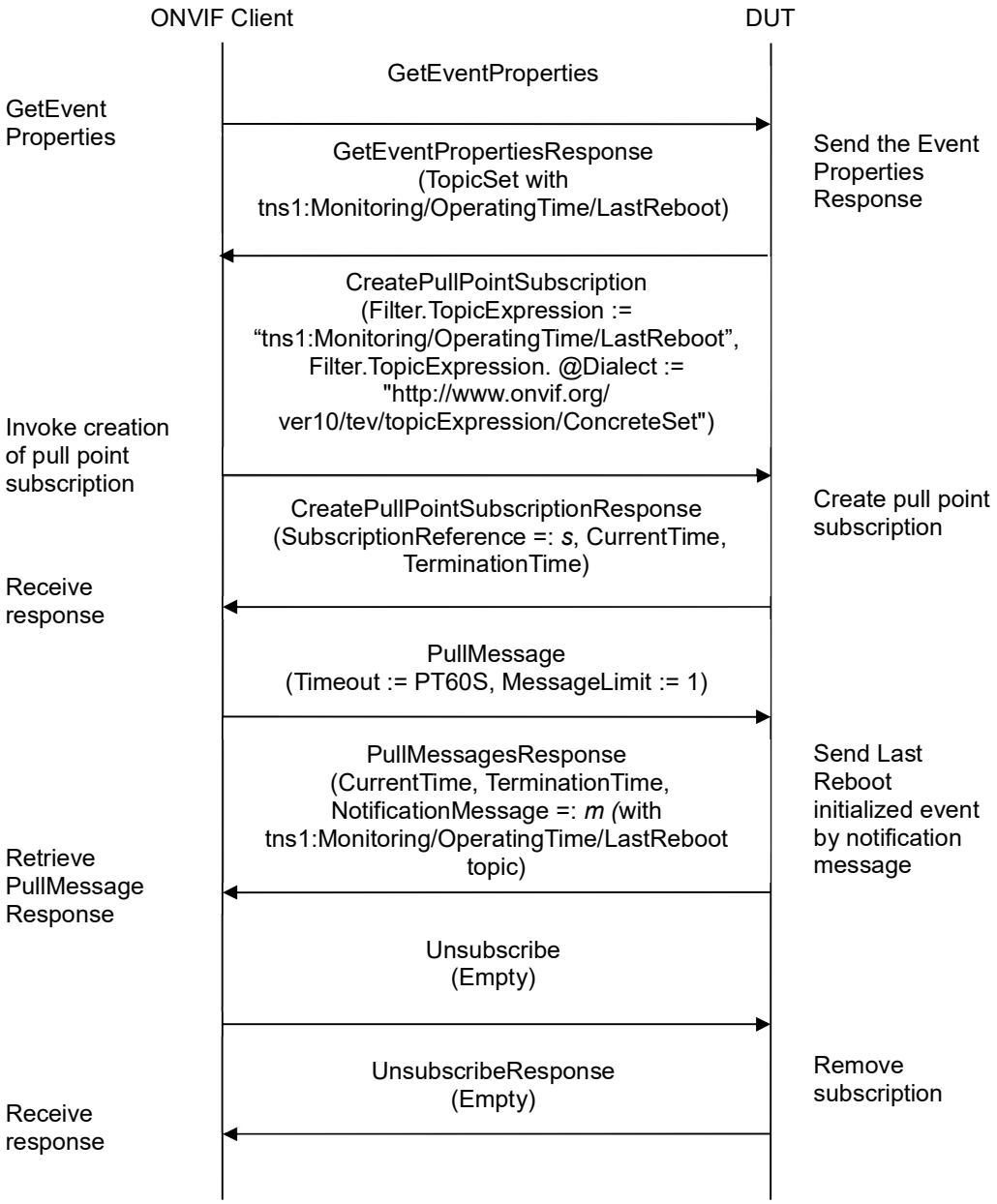
WSDL Reference: devicemgmt.wsdl and event.wsdl

Test Purpose: To verify tns1:Monitoring/OperatingTime/LastReboot event generation after subscription and to verify tns1:Monitoring/OperatingTime/LastReboot event format.

Pre-requisite: Event Service was received from the DUT. tns1:Monitoring/OperatingTime/LastReboot event is supported by the DUT as indicated by the GetEventPropertiesResponse.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters

- TopicNamespaceLocation list
- FixedTopicSet
- TopicSet =: *topicSet*
- TopicExpressionDialect list
- MessageContentFilterDialect list
- MessageContentSchemaLocation list

5. If *topicSet* does not contain tns1:Monitoring/OperatingTime/LastReboot topic, FAIL the test and skip other steps.
6. ONVIF Client verifies tns1:Monitoring/OperatingTime/LastReboot topic (*lastRebootTopic*) from *topicSet*:
   6.1. If *lastRebootTopic*.MessageDescription.IsProperty is skipped or equals to false, FAIL the test and skip other steps.
   6.2. If *lastRebootTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = “Status”, FAIL the test and skip other steps.
   6.3. If *lastRebootTopic.*MessageDescription.Data.SimpleItemDescription with Name = “Status” does not have Type = “xs:dateTime”, FAIL the test and skip other steps.
7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
   - Filter.TopicExpression := “tns1:Monitoring/OperatingTime/LastReboot”
   - Filter.TopicExpression.@Dialect := “http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet”
8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
   - SubscriptionReference =: *s*
   - CurrentTime
   - TerminationTime
9. Until *timeout1* timeout expires, repeat the following steps:
   9.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
      - Timeout := PT60S
      - MessageLimit := 1
   9.2. The DUT responds with **PullMessagesResponse** message with parameters
      - CurrentTime
      - TerminationTime
      - NotificationMessage =: *m*
   9.3. If *m* is not null and *m*.Message.Message.PropertyOperation=“Initialized” ONVIF Client verifies

m:

9.3.1. If *m*.Topic does not equal to tns1:Monitoring/OperatingTime/LastReboot, FAIL the test and go to the step 10.

9.3.2. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 10.

9.3.3. If *m*.Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 10.

9.3.4. Go to the step 10.

9.4. If *timeout1* timeout expires for step 9 without Notification with PropertyOperation="Initialized", FAIL the test and go to the step 10.

10. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

11. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

The DUT did not send **CreatePullPointSubscriptionResponse** message.

The DUT did not send **PullMessagesResponse** message(s).

The DUT did not send **UnsubscribeResponse** message.

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

6.8.4 Last Reboot event (Status change)

Test Label: Last Reboot event (Status change)

Test Case ID: DEVICE-9-1-4

ONVIF Core Specification Coverage: Monitoring Event LastReboot

Command Under Test: CreatePullPointSubscription, PullMessages, SystemReboot, Monitoring/OperatingTime/LastReboot event

WSDL Reference: devicemgmt.wsdl and event.wsdl

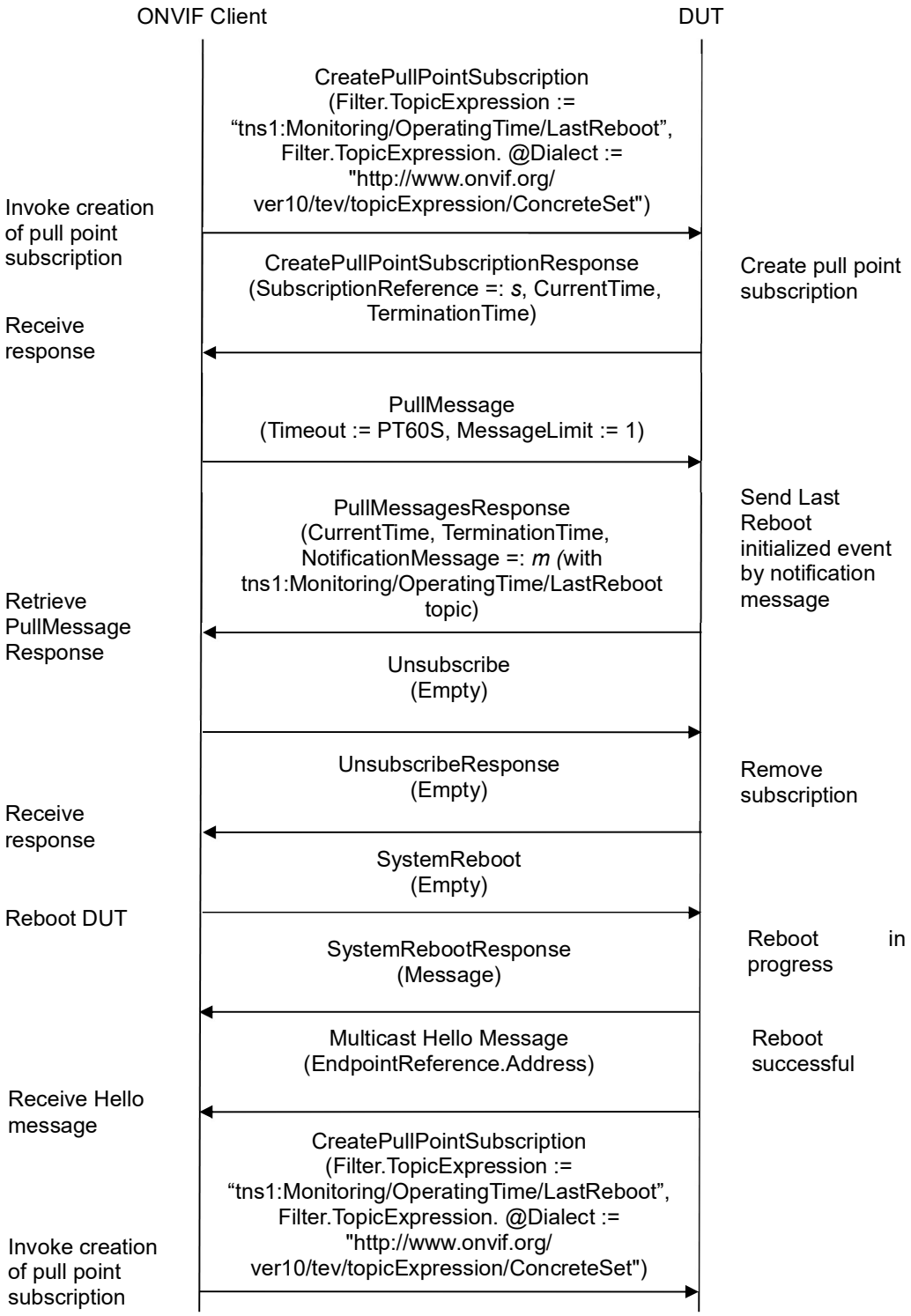
Test Purpose: To verify that the last reboot event signals correct values.

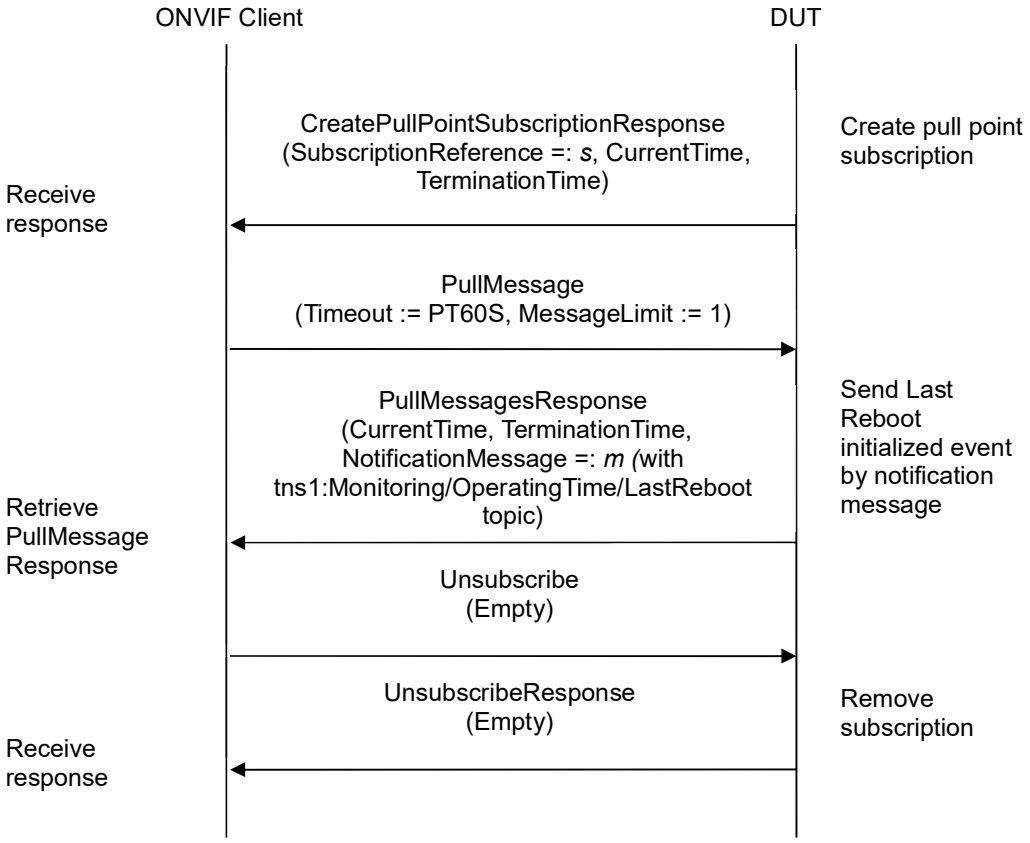
Pre-requisite: Event Service was received from the DUT. tns1:Monitoring/OperatingTime/LastReboot event is supported by the DUT as indicated by the GetEventPropertiesResponse. NTP server in network and configured on the DUT.

Test Configuration: ONVIF Client and DUT

Driving IP-based physical security through global standardization

Test Sequence:





Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **CreatePullPointSubscription** with parameters
   - Filter.TopicExpression := “tns1:Monitoring/OperatingTime/LastReboot”
   - Filter.TopicExpression.@Dialect := “http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet”
4. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
   - SubscriptionReference =: *s*
   - CurrentTime
   - TerminationTime
5. Until *timeout1* timeout expires, repeat the following steps:

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

- Timeout := PT60S
- MessageLimit := 1

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

- CurrentTime
- TerminationTime
- NotificationMessage =: *m*

5.3. If *m* is not null and *m*.Message.Message.PropertyOperation="Initialized"*:*

5.3.1. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 6.

5.3.2. If *m*.Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 17.

5.3.3. Set the following:

- *lastRebootTime* := *m*.Message.Message.Data.SimpleItem.Status

5.3.4. Go to the step 6.

5.4. If *timeout1* timeout expires for step 5 without Notification with PropertyOperation="Initialized", FAIL the test and go to the step 17.

6. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*. The test should not FAIL even if **Unsubscribe** fails.
7. The DUT responds with **UnsubscribeResponse** message. The test should not FAIL even if **UnsubscribeResponse** message is not received.
8. ONVIF Client invokes **SystemReboot**.
9. The DUT responds with **SystemRebootResponse** message with parameters

- Message

10. Until *timeout2* timeout expires repeat the following steps:

10.1. The DUT will send Multicast **Hello** message after it is successfully rebooted with parameters:

- EndpointReference.Address equal to unique endpoint reference of the DUT
- Types list
- Scopes list
- XAddrs list := *xaddrsList*
- MetadataVersion

10.2. If *xaddrsList* contains URI address with not a LinkLocal IPv4 address from ONVIF Client subnet, go to step 12.

11. If *timeout2* timeout expires for step 10 without Hello with URI address with not a LinkLocal IPv4 address from ONVIF Client subnet, FAIL the test and skip other steps.

12. ONVIF client waits for 5 seconds after Hello was received.
13. ONVIF Client invokes **CreatePullPointSubscription** with parameters
    - Filter.TopicExpression := “tns1:Monitoring/OperatingTime/LastReboot”
    - Filter.TopicExpression.@Dialect := “http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet”
14. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
    - SubscriptionReference =: *s*
    - CurrentTime
    - TerminationTime
15. Until *timeout1* timeout expires, repeat the following steps:
    15.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
        - Timeout := PT60S
        - MessageLimit := 1
    15.2. The DUT responds with **PullMessagesResponse** message with parameters
        - CurrentTime
        - TerminationTime
        - NotificationMessage =: *m*
    15.3. If *m* is not null and *m*.Message.Message.PropertyOperation=“Initialized”*:*
        15.3.1. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 16.
        15.3.2. If *m*.Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 17.
        15.3.3. Set the following:
            - *updatedRebootTime* := *m*.Message.Message.Data.SimpleItem.Status
        15.3.4. Go to the step 16.
    15.4. If *timeout1* timeout expires for step 15 without Notification with PropertyOperation=“Initialized”, FAIL the test and go to the step 17.
16. If *updatedRebootTime* <= *lastRebootTime*, FAIL the test and go to the next step.
17. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.
18. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send **CreatePullPointSubscriptionResponse** message.

The DUT did not send **PullMessagesResponse** message(s).

The DUT did not send **SystemRebootResponse** message.

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

Note: *timeout2* will be taken from Reboot Timeout field of ONVIF Device Test Tool.

Note: IPv4 address from Hello shall be used for further test cases. Previous Device service address will be selected if it is present on the XAddrs list of Hello message.

6.8.5 Last Clock Synchronization event

Test Label: Last Clock Synchronization event

Test Case ID: DEVICE-9-1-5

ONVIF Core Specification Coverage: Monitoring Event LastClockSynchronication

Command Under Test: GetServices, GetEventProperties, CreatePullPointSubscription, PullMessages, Monitoring/OperatingTime/LastClockSynchronization event

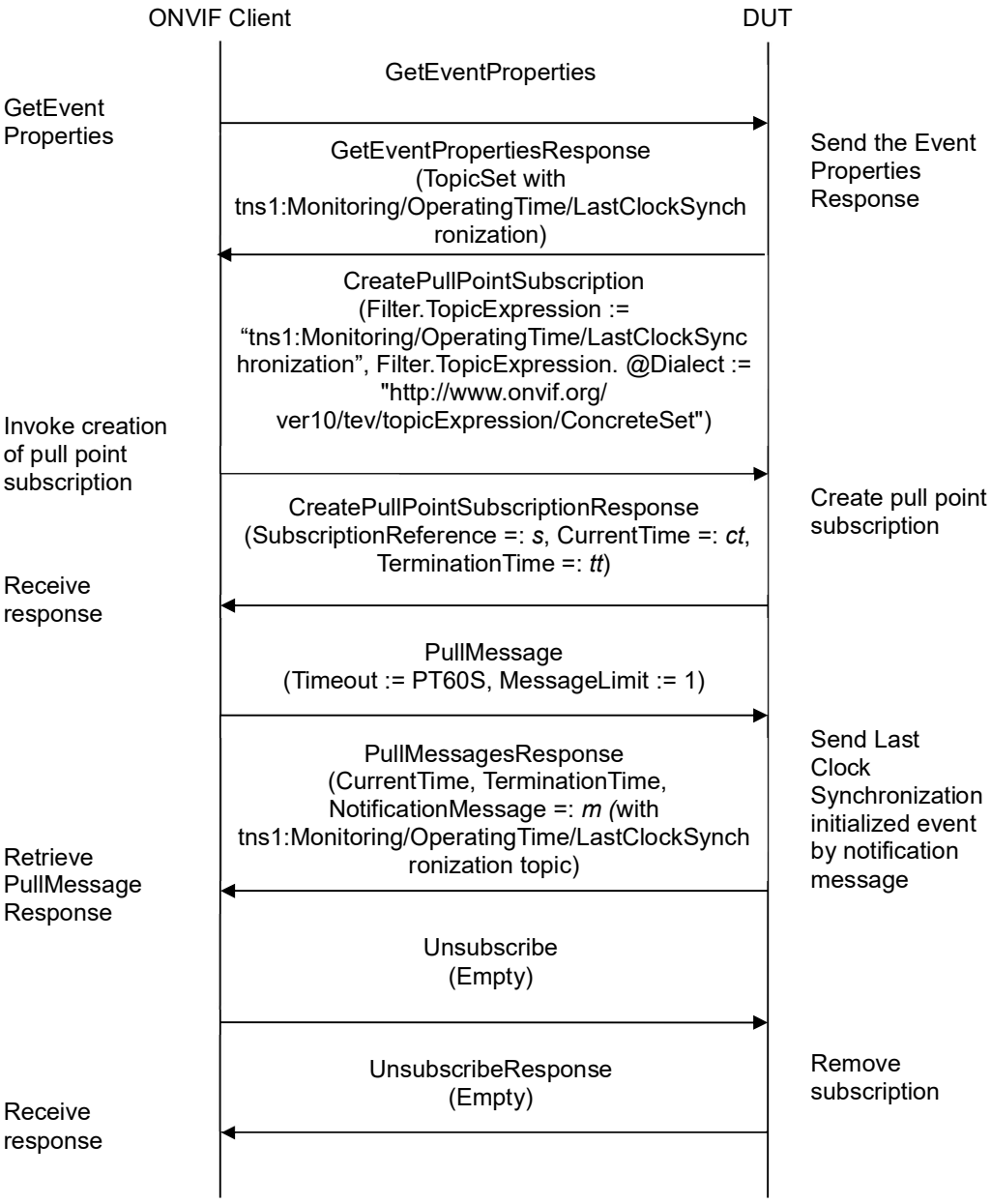
WSDL Reference: devicemgmt.wsdl and event.wsdl

Test Purpose: To verify tns1:Monitoring/OperatingTime/LastClockSynchronization event generation after subscription and to verify tns1:Monitoring/OperatingTime/LastClockSynchronization event format.

Pre-requisite: Event Service was received from the DUT. tns1:Monitoring/OperatingTime/LastClockSynchronization event is supported by the DUT as indicated by the GetEventPropertiesResponse.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list

- FixedTopicSet
- TopicSet =: *topicSet*
- TopicExpressionDialect list
- MessageContentFilterDialect list
- MessageContentSchemaLocation list

5. If *topicSet* does not contain tns1:Monitoring/OperatingTime/LastClockSynchronization topic, FAIL the test and skip other steps.
6. ONVIF Client verifies tns1:Monitoring/OperatingTime/LastClockSynchronization topic (*lastClockSynchronizationTopic*) from *topicSet*:
   6.1. If *lastClockSynchronizationTopic*.MessageDescription.IsProperty is skipped or equals to false, FAIL the test and skip other steps.
   6.2. If *lastClockSynchronizationTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = “Status”, FAIL the test and skip other steps.
   6.3. If *lastClockSynchronizationTopic.*MessageDescription.Data.SimpleItemDescription with Name = “Status” does not have Type = “xs:dateTime”, FAIL the test and skip other steps.
7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
   - Filter.TopicExpression := “tns1:Monitoring/OperatingTime/LastClockSynchronization”
   - Filter.TopicExpression.@Dialect := “http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet”
8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
   - SubscriptionReference =: *s*
   - CurrentTime
   - TerminationTime
9. Until *timeout1* timeout expires, repeat the following steps:
   9.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
      - Timeout := PT60S
      - MessageLimit := 1
   9.2. The DUT responds with **PullMessagesResponse** message with parameters
      - CurrentTime
      - TerminationTime
      - NotificationMessage =: *m*
   9.3. If *m* is not null and *m*.Message.Message.PropertyOperation=“Initialized” ONVIF Client verifies *m:*

9.3.1. If *m*.Topic does not equal to tns1:Monitoring/OperatingTime/LastClockSynchronization, FAIL the test and go to the step 10.

9.3.2. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 10.

9.3.3. If *m*.Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 10.

9.3.4. If *m*.Message.Message.Data.SimpleItem.Status has value time in format different from utc format with including the 'Z' indicator, FAIL the test and go to the step 10.

9.3.5. Go to the step 10.

9.4. If *timeout1* timeout expires for step 9 without Notification with PropertyOperation="Initialized", FAIL the test and go to the step 10.

10. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

11. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

The DUT did not send **CreatePullPointSubscriptionResponse** message.

The DUT did not send **PullMessagesResponse** message(s).

The DUT did not send **UnsubscribeResponse** message.

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

6.8.6 Last Clock Synchronization change event (SetSystemDateAndTime)

Test Label: Last Clock Synchronization change event (SetSystemDateAndTime)

Test Case ID: DEVICE-9-1-6

ONVIF Core Specification Coverage: Monitoring Event LastClockSynchronication

Command Under Test: GetServices, CreatePullPointSubscription, PullMessages, SetSystemDateAndTime, Monitoring/OperatingTime/LastClockSynchronization event

WSDL Reference: devicemgmt.wsdl and event.wsdl

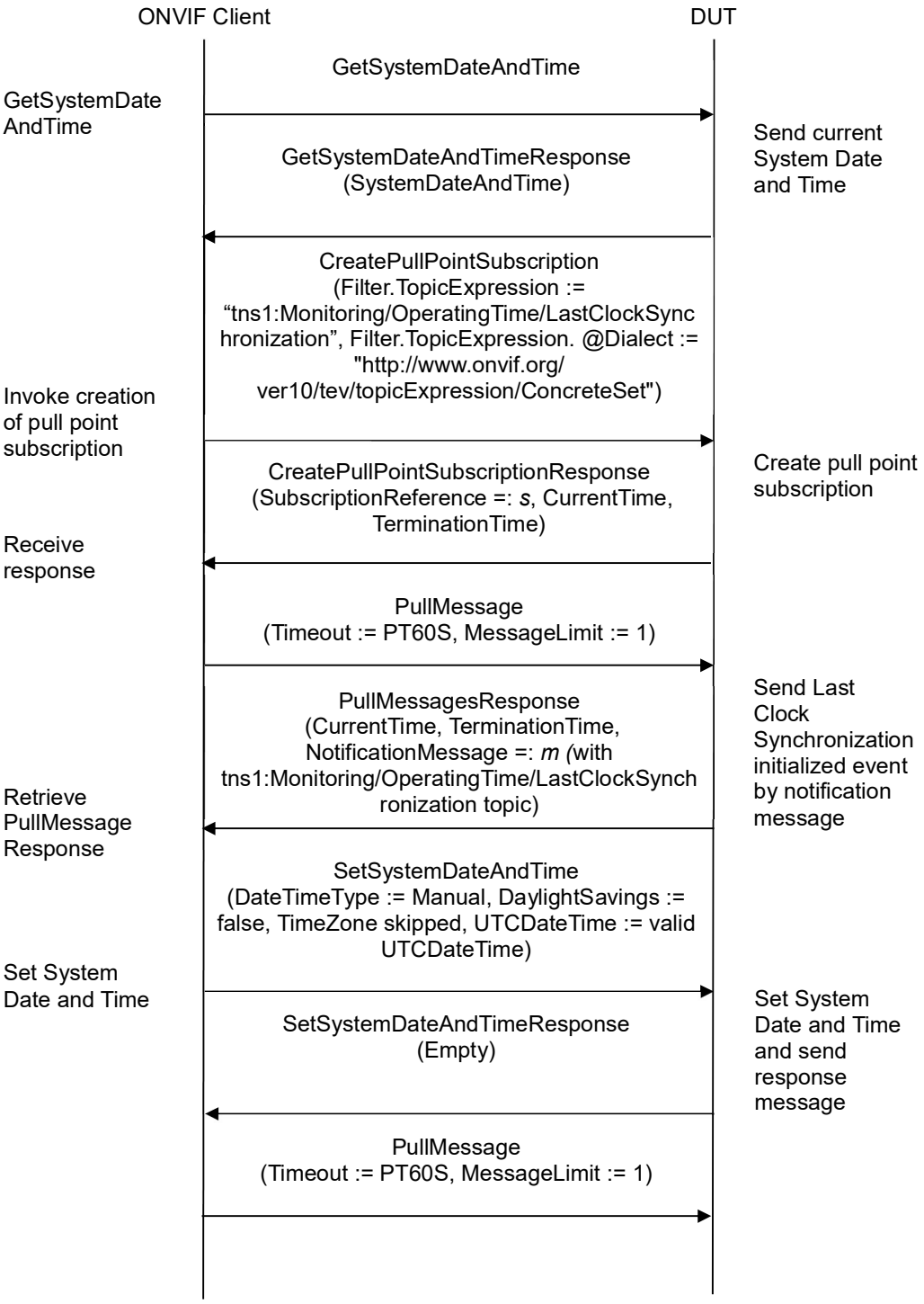
Test Purpose: To verify tns1:Monitoring/OperatingTime/LastClockSynchronization event generation after property was changed via SetSystemDateAndTime call and to verify tns1:Monitoring/OperatingTime/LastClockSynchronization event format. To verify that the clock synchronization event signals correct values.

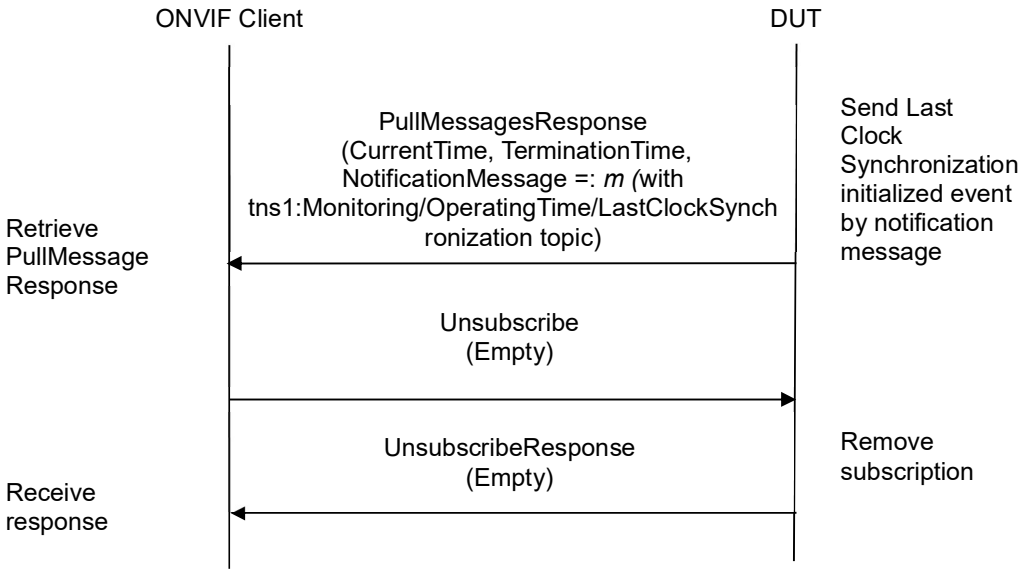
Pre-requisite: Event Service was received from the DUT.

tns1:Monitoring/OperatingTime/LastClockSynchronization event is supported by the DUT as indicated by the GetEventPropertiesResponse.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
GetSystemDateAndTime
GetSystemDate AndTime
Send current System Date and Time
GetSystemDateAndTimeResponse (SystemDateAndTime)
CreatePullPointSubscription (Filter.TopicExpression := “tns1:Monitoring/OperatingTime/LastClockSynchronization”, Filter.TopicExpression. @Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet")
Invoke creation of pull point subscription
Create pull point subscription
CreatePullPointSubscriptionResponse (SubscriptionReference =: s, CurrentTime, TerminationTime)
Receive response
PullMessage (Timeout := PT60S, MessageLimit := 1)
Send Last Clock Synchronization initialized event by notification message
PullMessagesResponse (CurrentTime, TerminationTime, NotificationMessage =: m (with tns1:Monitoring/OperatingTime/LastClockSynchronization topic)
Retrieve PullMessage Response
SetSystemDateAndTime (DateTimeType := Manual, DaylightSavings := false, TimeZone skipped, UTCDateTime := valid UTCDateTime)
Set System Date and Time
Set System Date and Time and send response message
SetSystemDateAndTimeResponse (Empty)
PullMessage (Timeout := PT60S, MessageLimit := 1)



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetSystemDateAndTime**.
4. The DUT responds with a **GetSystemDateAndTimeResponse** message with parameters
   - SystemDateAndTime =: *initialSystemDateAndTime*
5. ONVIF Client invokes **CreatePullPointSubscription** with parameters
   - Filter.TopicExpression := “tns1:Monitoring/OperatingTime/LastClockSynchronization”
   - Filter.TopicExpression.@Dialect := “http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet”
6. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
   - SubscriptionReference =: *s*
   - CurrentTime
   - TerminationTime
7. Until *timeout1* timeout expires, repeat the following steps:

   7.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
   - Timeout := PT60S
   - MessageLimit := 1

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

- CurrentTime
- TerminationTime
- NotificationMessage =: *m*

7.3. If *m* is not null and *m*.Message.Message.PropertyOperation="Initialized" ONVIF Client verifies *m:*

7.3.1. If *m*.TopicDialect does not equal to tns1:Monitoring/OperatingTime/LastClockSynchronization, FAIL the test and go to the step 14.

7.3.2. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 14.

7.3.3. If *m*.Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 14.

7.3.4. Go to the step 8.

7.4. If *timeout1* timeout expires for step 7 without Notification with PropertyOperation="Initialized", FAIL the test and go to the step 14.

8. Set the following:

- *initialClockSynchronization* := *m*.Message.Message.Data.SimpleItem.Status

9. ONVIF Client waits for 1 second.

10. ONVIF Client invokes **SetSystemDateAndTime** with parameters

- DateTimeType := Manual
- DaylightSavings := false
- TimeZone skipped
- UTCDateTime := *initialSystemDateAndTime.UTCDateTime* received from DUT on step 4. If *initialSystemDateAndTime.UTCDateTime* is empty, then system date and time should be used to populate this parameter.

11. The DUT responds with a **SetSystemDateAndTimeResponse** message.

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

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

- Timeout := PT60S
- MessageLimit := 1

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

- CurrentTime
- TerminationTime

- NotificationMessage =: *m*

12.3. If *m* is not null and *m*.Message.Message.PropertyOperation="Changed" ONVIF Client verifies *m:*

12.3.1. If *m*.TopicDialect does not equal to tns1:Monitoring/OperatingTime/LastClockSynchronization, FAIL the test and go to the step 14.

12.3.2. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 14.

12.3.3. If *m*.Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 14.

12.3.4. Set the following:

- *lastClockSynchronization* := m.Message.Message.Data.SimpleItem.Status

12.3.5. Go to the step 13.

12.4. If *timeout1* timeout expires for step 11 without Notification with PropertyOperation="Changed", FAIL the test and go to the step 14.

13. If *lastClockSynchronization* is less or equal to *initialClockSynchronization*, FAIL the test and go to the next step.

14. ONVIF Client restores Default System Date and Time by following the procedure mentioned in Annex A.23.

15. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

16. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send **GetSystemDateAndTimeResponse** message.

The DUT did not send **CreatePullPointSubscriptionResponse** message.

The DUT did not send **PullMessagesResponse** message(s).

The DUT did not send **UnsubscribeResponse** message.

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

6.8.7 Last Clock Synchronization change event (NTP message)

Test Label: Last Clock Synchronization change event (NTP message)

Test Case ID: DEVICE-9-1-7

ONVIF Core Specification Coverage: Monitoring Event LastClockSynchronication

Command Under Test: GetServices, CreatePullPointSubscription, PullMessages, SetSystemDateAndTime, SetNTP, Monitoring/OperatingTime/LastClockSynchronization event

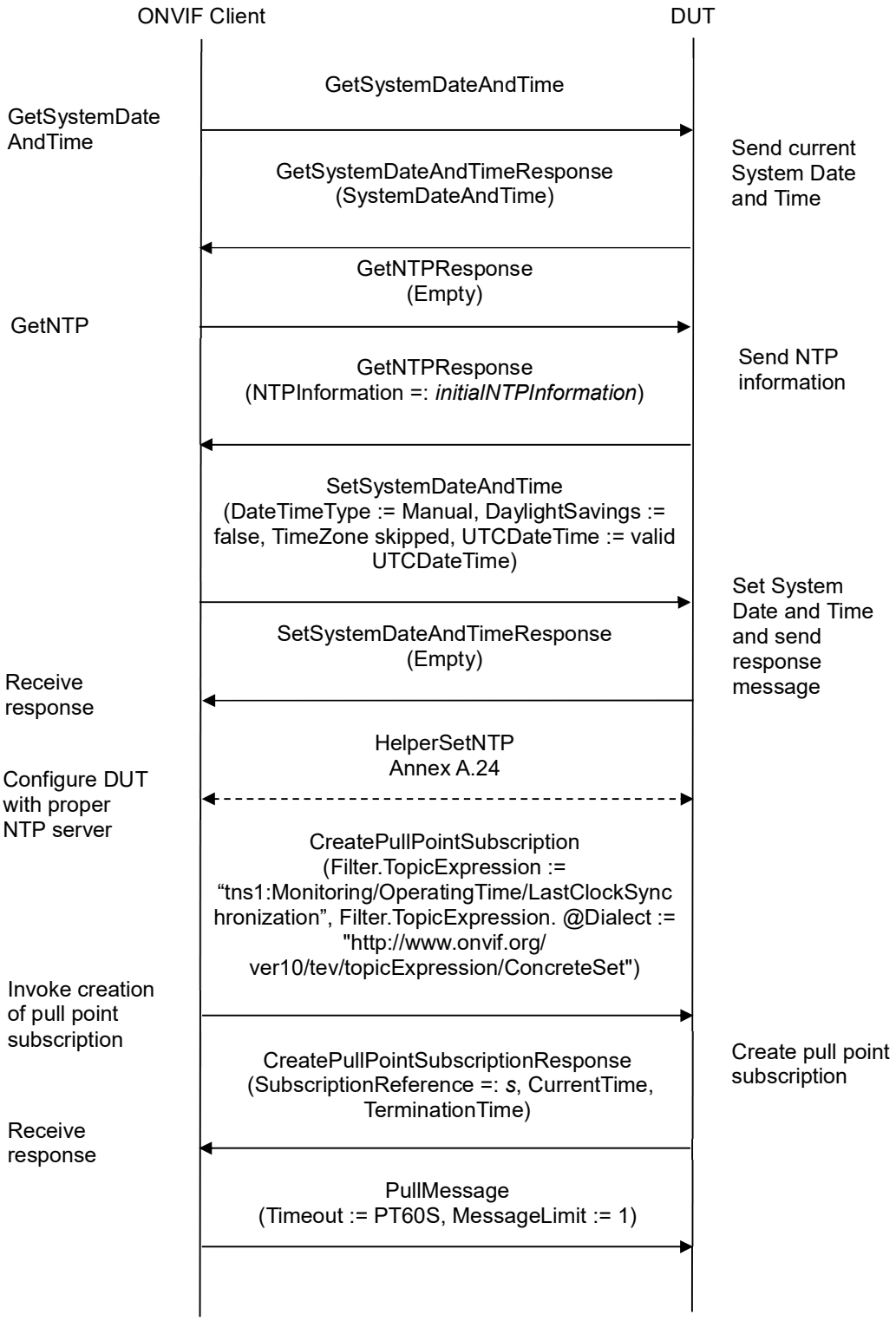
WSDL Reference: devicemgmt.wsdl and event.wsdl

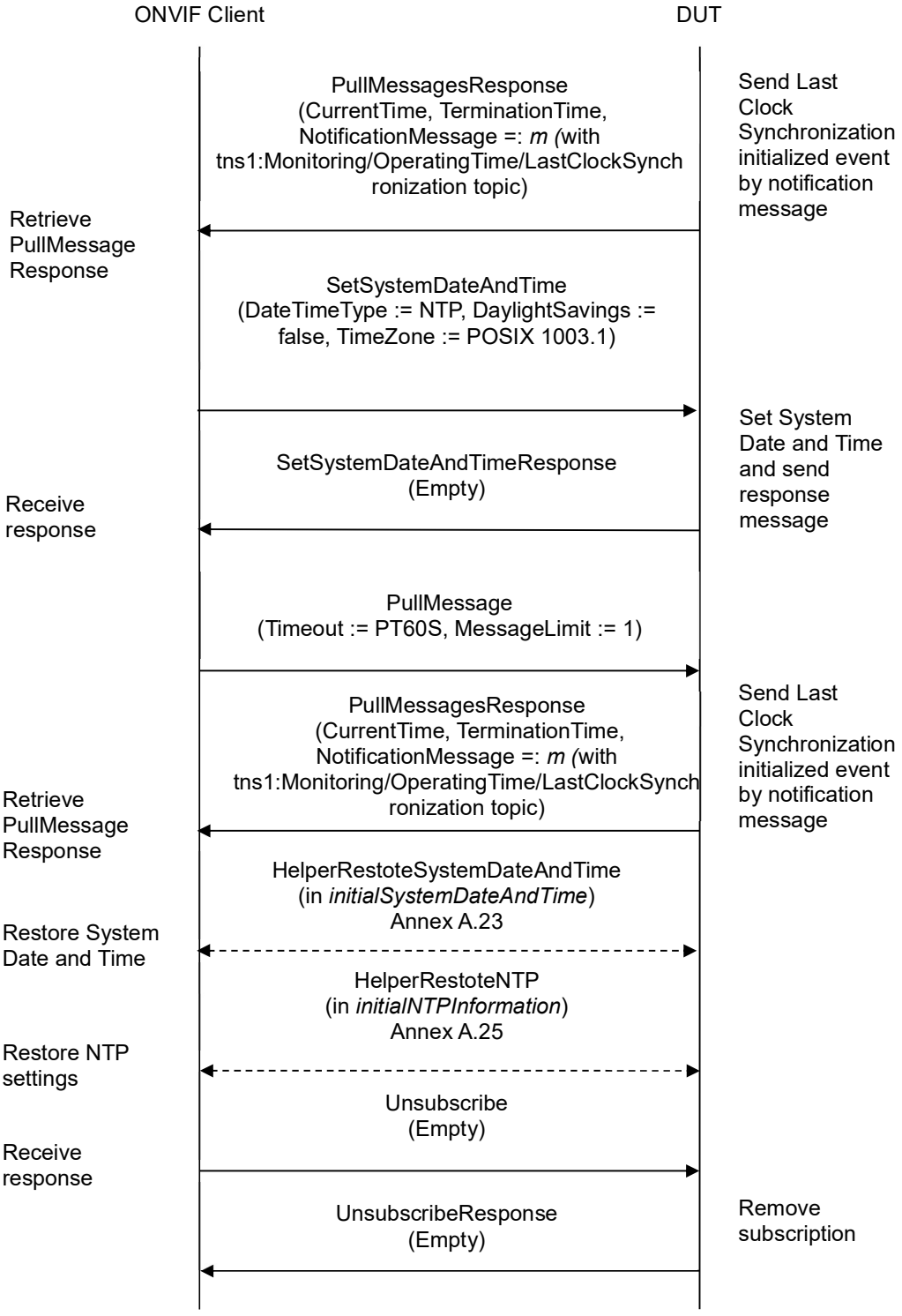
Test Purpose: To verify tns1:Monitoring/OperatingTime/LastClockSynchronization event generation after property was changed via an NTP message and to verify tns1:Monitoring/OperatingTime/LastClockSynchronization event format. To verify that the clock synchronization event signals correct values.

Pre-requisite: Event Service was received from the DUT. tns1:Monitoring/OperatingTime/LastClockSynchronization event is supported by the DUT as indicated by the GetEventPropertiesResponse. NTP is supported by the DUT as indicated by the Network.NTP capability. A valid NTP server address should be configured in the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
GetSystemDateAndTime
GetSystemDate AndTime
Send current System Date and Time
GetSystemDateAndTimeResponse (SystemDateAndTime)
GetNTPResponse (Empty)
GetNTP
Send NTP information
GetNTPResponse (NTPInformation =: initialNTPInformation)
SetSystemDateAndTime (DateTimeType := Manual, DaylightSavings := false, TimeZone skipped, UTCDateTime := valid UTCDateTime)
Set System Date and Time and send response message
SetSystemDateAndTimeResponse (Empty)
Receive response
HelperSetNTP Annex A.24
Configure DUT with proper NTP server
CreatePullPointSubscription (Filter.TopicExpression := “tns1:Monitoring/OperatingTime/LastClockSynchronization”, Filter.TopicExpression. @Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet")
Invoke creation of pull point subscription
Create pull point subscription
CreatePullPointSubscriptionResponse (SubscriptionReference =: s, CurrentTime, TerminationTime)
Receive response
PullMessage (Timeout := PT60S, MessageLimit := 1)



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetSystemDateAndTime**.
4. The DUT responds with a **GetSystemDateAndTimeResponse** message with parameters
   - SystemDateAndTime =: *initialSystemDateAndTime*
5. ONVIF Client invokes **GetNTP**.
6. The DUT responds with a **GetNTPResponse** message with parameters
   - NTPInformation =: *initialNTPInformation*
7. If *initialSystemDateAndTime*.DataTimeType = NTP, ONVIF Client invokes **SetSystemDateAndTime** with parameters
   - DateTimeType := Manual
   - DaylightSavings := false
   - TimeZone skipped
   - UTCDateTime := *initialSystemDateAndTime.UTCDateTime* received from DUT on step 4. If *initialSystemDateAndTime.UTCDateTime* is empty, then system date and time should be used to populate this parameter.
8. The DUT responds with a **SetSystemDateAndTimeResponse** message.
9. ONVIF Client configures DUT with proper NTP server by following the procedure mentioned in Annex A.24.
10. ONVIF Client invokes **CreatePullPointSubscription** with parameters
    - Filter.TopicExpression := “tns1:Monitoring/OperatingTime/LastClockSynchronization”
    - Filter.TopicExpression.@Dialect := “http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet”
11. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
    - SubscriptionReference =: *s*
    - CurrentTime
    - TerminationTime
12. Until *timeout1* timeout expires, repeat the following steps:

    12.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
    - Timeout := PT60S
    - MessageLimit := 1

    12.2. The DUT responds with **PullMessagesResponse** message with parameters
    - CurrentTime

- TerminationTime
- NotificationMessage =: *m*

12.3. If *m* is not null and *m*.Message.Message.PropertyOperation="Initialized" ONVIF Client verifies *m:*

12.3.1. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 18.

12.3.2. If *m*.Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 18.

12.3.3. Go to the step 13.

12.4. If *timeout1* timeout expires for step 12 without Notification with PropertyOperation="Initialized", FAIL the test and go to the step 18.

13. Set the following:
    - *initialClockSynchronization* := *m*.Message.Message.Data.SimpleItem.Status
14. ONVIF Client waits for 1 second.
15. ONVIF Client invokes **SetSystemDateAndTime** with parameters
    - DateTimeType := NTP
    - DaylightSavings := false
    - TimeZone := POSIX 1003.1
    - UTCDateTime skipped
16. The DUT responds with a **SetSystemDateAndTimeResponse** message.
17. Set the following:
    - *timeOfResponse* := time of SetSystemDateAndTimeResponse message receiving
18. Until *timeout1* timeout expires, repeat the following steps:

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

- Timeout := PT60S
- MessageLimit := 1

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

- CurrentTime
- TerminationTime
- NotificationMessage =: *m*

18.3. If *m* is not null and *m*.Message.Message.PropertyOperation="Changed" ONVIF Client verifies *m:*

18.3.1. If *m*.TopicDialect does not equal to

tns1:Monitoring/OperatingTime/LastClockSynchronization, FAIL the test and go to the step 20.

18.3.2. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 20.

18.3.3. If *m*.Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 20.

18.3.4. Set the following:

- *lastClockSynchronization* := m.Message.Message.Data.SimpleItem.Status

18.3.5. Go to the step 19.

18.4. If *timeout1* timeout expires for step 10 without Notification with PropertyOperation="Changed", FAIL the test and go to the step 20.

19. If *lastClockSynchronization* is less or equal to *initialClockSynchronization*, FAIL the test and go to the next step.

20. ONVIF Client restores Default System Date and Time by following the procedure mentioned in Annex A.23.

21. ONVIF Client restores Default NTP settings by following the procedure mentioned in Annex A.25.

22. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

23. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send **GetSystemDateAndTimeResponse** message.

The DUT did not send **GetNTPResponse** message.

The DUT did not send **CreatePullPointSubscriptionResponse** message.

The DUT did not send **SetSystemDateAndTimeResponse** message.

The DUT did not send **PullMessagesResponse** message(s).

The DUT did not send **UnsubscribeResponse** message.

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

6.8.8 Last Backup event

Test Label: Last Backup event

Test Case ID: DEVICE-9-1-8

ONVIF Core Specification Coverage: Monitoring Event Last Backup

Command Under Test: GetServices, GetEventProperties, CreatePullPointSubscription, PullMessages, Monitoring/Backup/Last event

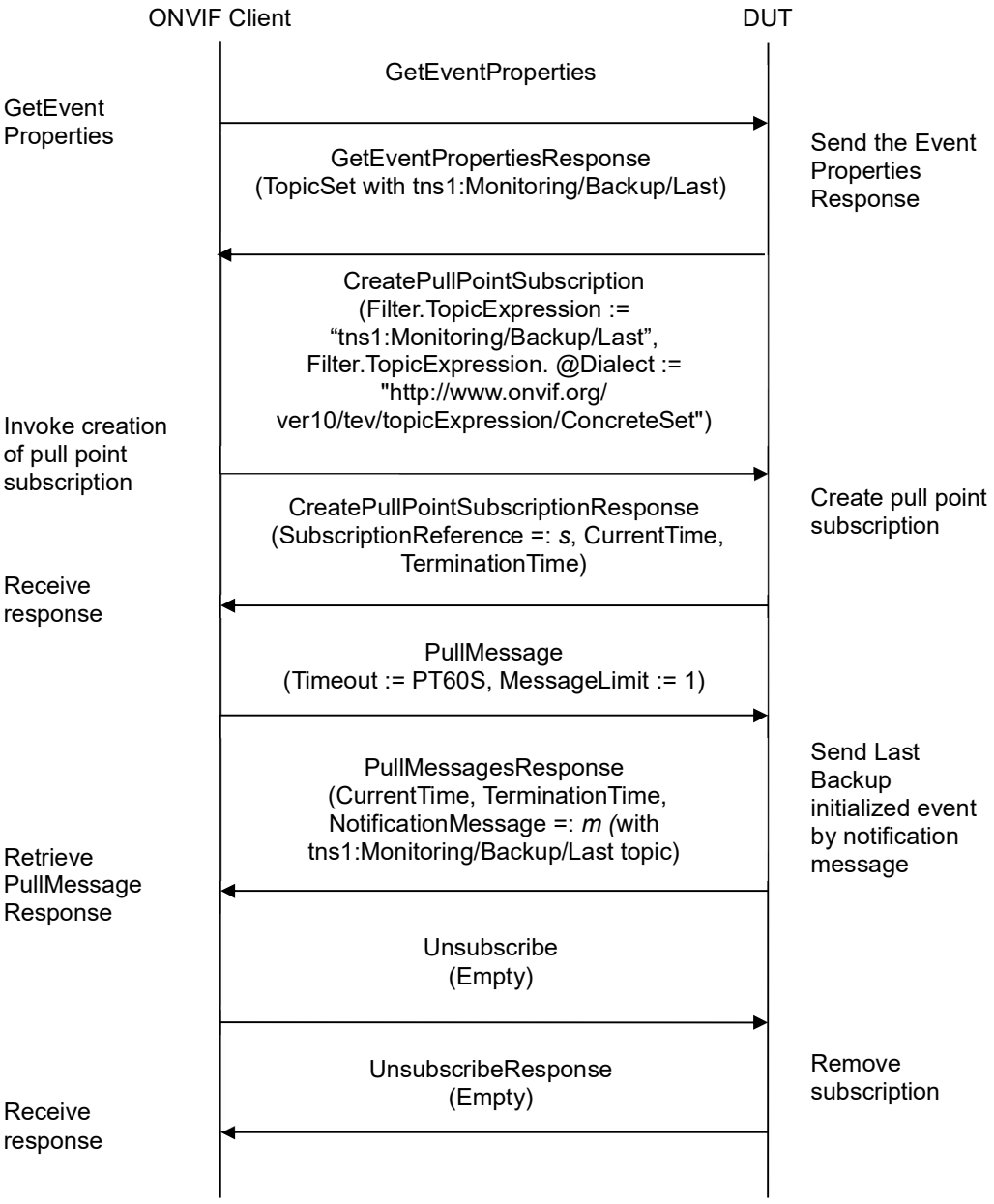
WSDL Reference: devicemgmt.wsdl and event.wsdl

Test Purpose: To verify tns1:Monitoring/Backup/Last intial event generation and to verify tns1:Monitoring/Backup/Last event format.

Pre-requisite: Event Service was received from the DUT. tns1:Monitoring/Backup/Last event is supported by the DUT as indicated by the GetEventPropertiesResponse.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
    - TopicNamespaceLocation list

- FixedTopicSet
- TopicSet =: topicSet
- TopicExpressionDialect list
- MessageContentFilterDialect list
- MessageContentSchemaLocation list

5. If *topicSet* does not contain tns1:Monitoring/Backup/Last topic, FAIL the test and skip other steps.
6. ONVIF Client verifies tns1:Monitoring/Backup/Last topic (*lastBackupTopic*) from *topicSet*:
   6.1. If *lastBackupTopic*.MessageDescription.IsProperty is skipped or equals to false, FAIL the test and skip other steps.
   6.2. If *lastBackupTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = “Status”, FAIL the test and skip other steps.
   6.3. If *lastBackupTopic*.MessageDescription.Data.SimpleItemDescription item with Name = “Status” does not have Type = “xs:dateTime”, FAIL the test and skip other steps.
7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
   - Filter.TopicExpression := “tns1:Monitoring/Backup/Last”
   - Filter.TopicExpression.@Dialect := “http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet”
8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
   - SubscriptionReference =: s
   - CurrentTime
   - TerminationTime
9. Until *timeout1* timeout expires, repeat the following steps:
   9.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
   - Timeout := PT60S
   - MessageLimit := 1

   9.2. The DUT responds with **PullMessagesResponse** message with parameters
   - CurrentTime
   - TerminationTime
   - NotificationMessage =: *m*

   9.3. If *m* is not null and *m*.Message.Message.PropertyOperation=“Initialized” ONVIF Client verifies *m:*
      9.3.1. If *m*.Topic does not equal to tns1:Monitoring/Backup/Last, FAIL the test and go to the step 10.

9.3.2. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 10.

9.3.3. If *m*.Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 10.

9.3.4. Go to the step 10.

9.4. If *timeout1* timeout expires for step 9 without Notification with PropertyOperation="Initialized", FAIL the test and go to the step 10.

10. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

11. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

The DUT did not send **CreatePullPointSubscriptionResponse** message.

The DUT did not send **PullMessagesResponse** message(s).

The DUT did not send **UnsubscribeResponse** message.

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

6.8.9 Fan Failure event

Test Label: Fan Failure event

Test Case ID: DEVICE-9-1-9

ONVIF Core Specification Coverage: Monitoring Event Fan Failure

Command Under Test: GetServices, GetEventProperties, CreatePullPointSubscription, PullMessages, Device/HardwareFailure/FanFailure event format

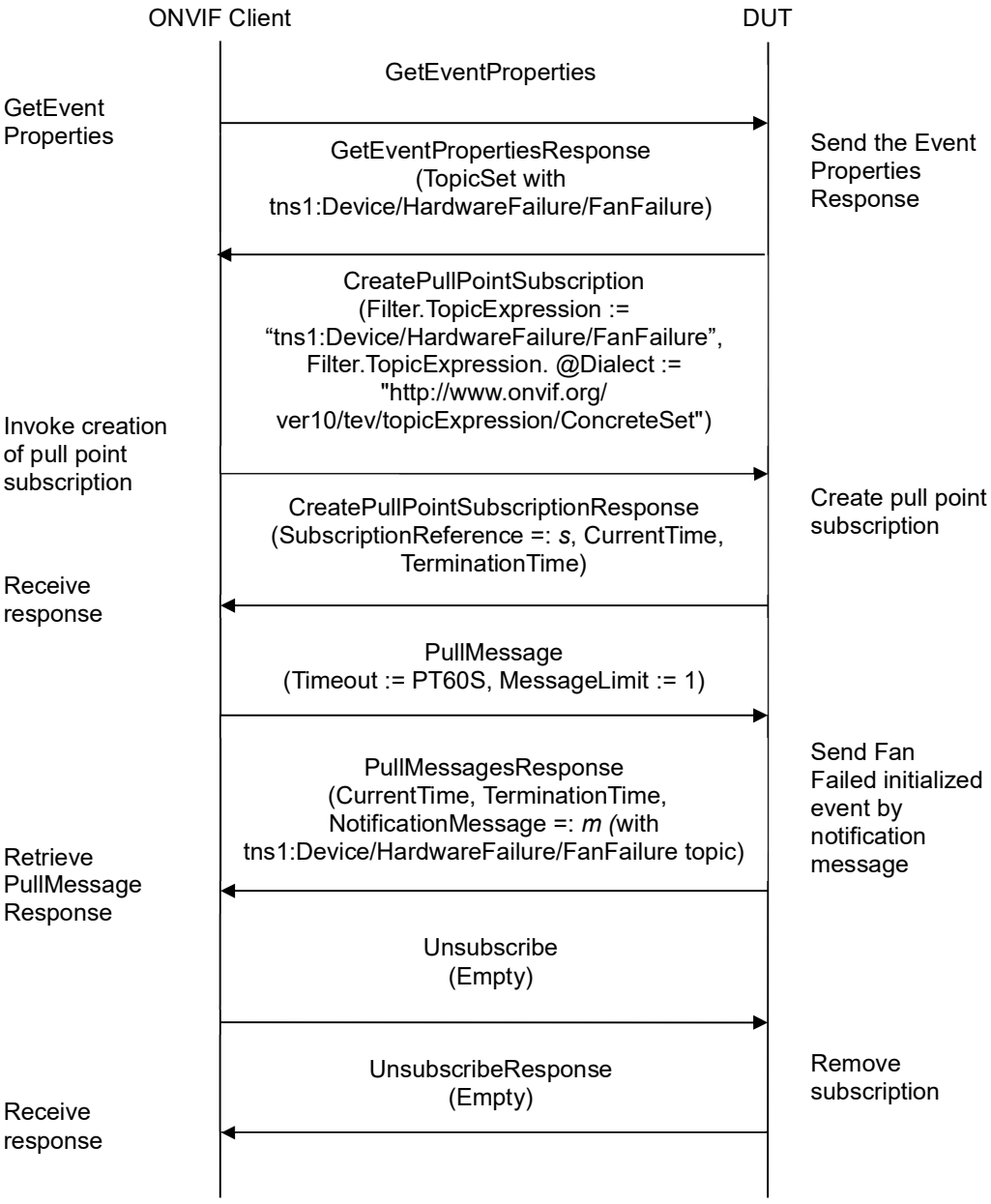
WSDL Reference: devicemgmt.wsdl and event.wsdl

Test Purpose: To verify tns1:Device/HardwareFailure/FanFailure event generation and to verify tns1:Device/HardwareFailure/FanFailure event format.

Pre-requisite: Event Service was received from the DUT. tns1:Device/HardwareFailure/FanFailure event is supported by the DUT as indicated by the GetEventPropertiesResponse.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
    - TopicNamespaceLocation list

- FixedTopicSet
- TopicSet =: topicSet
- TopicExpressionDialect list
- MessageContentFilterDialect list
- MessageContentSchemaLocation list

5. If *topicSet* does not contain tns1:Device/HardwareFailure/FanFailure topic, FAIL the test and skip other steps.
6. ONVIF Client verifies tns1:Device/HardwareFailure/FanFailure topic (*fanFailedTopic*) from *topicSet*:

   6.1. If *fanFailureTopic*.MessageDescription.IsProperty is skipped or equals to false, FAIL the test and skip other steps.

   6.2. If *fanFailureTopic* does not contain MessageDescription.Source.SimpleItemDescription item with Name = “Token”, FAIL the test and skip other steps.

   6.3. If *fanFailureTopic*.MessageDescription.Source.SimpleItemDescription item with Name = “Token” does not have Type = “tt:ReferenceToken”, FAIL the test and skip other steps.

   6.4. If *fanFailureTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = “Failed”, FAIL the test and skip other steps.

   6.5. If *fanFailureTopic*.MessageDescription.Data.SimpleItemDescription item with Name = “Failed” does not have Type = “xs:boolean”, FAIL the test and skip other steps.

7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
   - Filter.TopicExpression := “tns1:Device/HardwareFailure/FanFailure”
   - Filter.TopicExpression.@Dialect := “http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet”
8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
   - SubscriptionReference =: s
   - CurrentTime
   - TerminationTime
9. Until *timeout1* timeout expires, repeat the following steps:

   9.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
   - Timeout := PT60S
   - MessageLimit := 1

   9.2. The DUT responds with **PullMessagesResponse** message with parameters
   - CurrentTime
   - TerminationTime

- NotificationMessage =: *m*

9.3. If *m* is not null and *m*.Message.Message.PropertyOperation="Initialized" ONVIF Client verifies *m:*

9.3.1. If *m*.Topic does not equal to tns1:Device/HardwareFailure/FanFailure, FAIL the test and go to the step 10.

9.3.2. If *m* does not contain Message.Message.Source.SimpleItem.Token, FAIL the test and go to the step 10.

9.3.3. If *m*.Message.Message.Source.SimpleItem.Token has value type different from tt:ReferenceToken type, FAIL the test and go to the step 10.

9.3.4. If *m* does not contain Message.Message.Data.SimpleItem.Failed, FAIL the test and go to the step 10.

9.3.5. If *m*.Message.Message.Data.SimpleItem.Failed has value type different from xs:boolean type, FAIL the test and go to the step 10.

9.3.6. Go to the step 10.

9.4. If *timeout1* timeout expires for step 9 without Notification with PropertyOperation="Initialized", FAIL the test and go to the step 10.

10. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

11. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

The DUT did not send **CreatePullPointSubscriptionResponse** message.

The DUT did not send **PullMessagesResponse** message(s).

The DUT did not send **UnsubscribeResponse** message.

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

6.8.10 Power Supply Failure event

Test Label: Power Supply Failure event

Test Case ID: DEVICE-9-1-10

ONVIF Core Specification Coverage: Monitoring Event Power Supply Failure

Command Under Test: GetServices, GetEventProperties, CreatePullPointSubscription, PullMessages, Device/HardwareFailure/PowerSupplyFailure event

WSDL Reference: devicemgmt.wsdl and event.wsdl

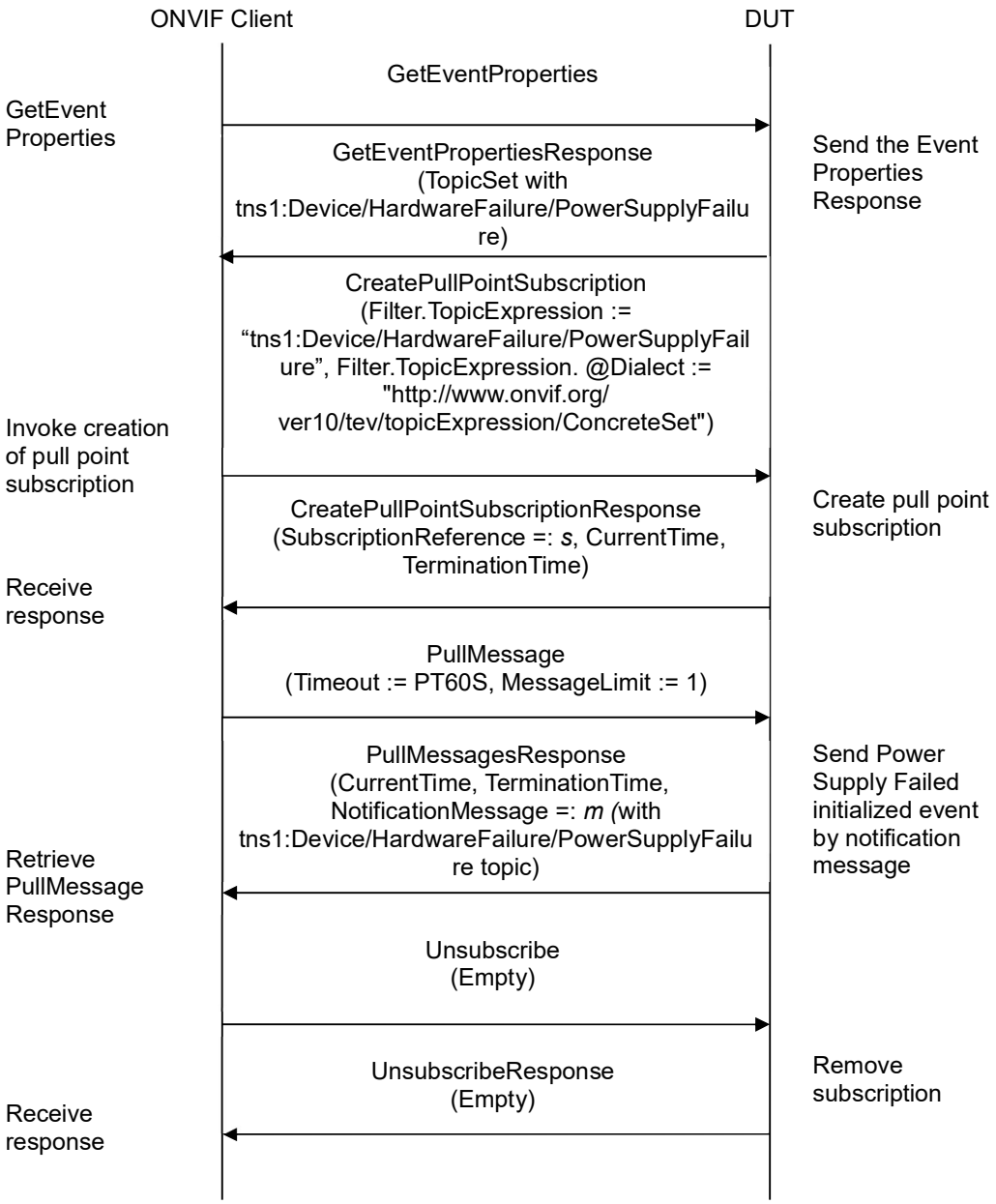
Driving IP-based physical security through global standardization

Test Purpose: To verify tns1:Device/HardwareFailure/PowerSupplyFailure event generation and to verify tns1:Device/HardwareFailure/PowerSupplyFailure event format.

Pre-requisite: Event Service was received from the DUT. tns1:Device/HardwareFailure/PowerSupplyFailure event is supported by the DUT as indicated by the GetEventPropertiesResponse.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
    - TopicNamespaceLocation list
    - FixedTopicSet
    - TopicSet =: topicSet
    - TopicExpressionDialect list
    - MessageContentFilterDialect list
    - MessageContentSchemaLocation list
5. If *topicSet* does not contain tns1:Device/HardwareFailure/PowerSupplyFailure topic, FAIL the test and skip other steps.
6. ONVIF Client verifies tns1:Device/HardwareFailure/PowerSupplyFailure topic (*powerSupplyFailureTopic*) from *topicSet*:
    6.1. If *powerSupplyFailureTopic*.MessageDescription.IsProperty is skipped or equals to false, FAIL the test and skip other steps.
    6.2. If *powerSupplyFailureTopic* does not contain MessageDescription.Source.SimpleItemDescription item with Name = “Token”, FAIL the test and skip other steps.
    6.3. If *powerSupplyFailureTopic*.MessageDescription.Source.SimpleItemDescription item with Name = “Token” does not have Type = “tt:ReferenceToken”, FAIL the test and skip other steps.
    6.4. If *powerSupplyFailedTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = “Failed”, FAIL the test and skip other steps.
    6.5. If *powerSupplyFailureTopic*.MessageDescription.Data.SimpleItemDescription item with Name = “Failed” does not have Type = “xs:boolean”, FAIL the test and skip other steps.
7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
    - Filter.TopicExpression := “tns1:Device/HardwareFailure/PowerSupplyFailure”
    - Filter.TopicExpression.@Dialect := “http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet”
8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
    - SubscriptionReference =: s
    - CurrentTime

- TerminationTime

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

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

   - Timeout := PT60S
   - MessageLimit := 1

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

   - CurrentTime
   - TerminationTime
   - NotificationMessage =: *m*

   9.3. If *m* is not null and *m*.Message.Message.PropertyOperation="Initialized" ONVIF Client verifies *m:*

   9.3.1. If *m*.Topic does not equal to tns1:Device/HardwareFailure/PowerSupplyFailure, FAIL the test and go to the step 10.

   9.3.2. If *m* does not contain Message.Message.Source.SimpleItem.Token, FAIL the test and go to the step 10.

   9.3.3. If *m*.Message.Message.Source.SimpleItem.Token has value type different from tt:ReferenceToken type, FAIL the test and go to the step 10.

   9.3.4. If *m* does not contain Message.Message.Data.SimpleItem.Failed, FAIL the test and go to the step 10.

   9.3.5. If *m*.Message.Message.Data.SimpleItem.Failed has value type different from xs:boolean type, FAIL the test and go to the step 10.

   9.3.6. Go to the step 10.

   9.4. If *timeout1* timeout expires for step 9 without Notification with PropertyOperation="Initialized", FAIL the test and go to the step 10.

10. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

11. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

The DUT did not send **CreatePullPointSubscriptionResponse** message.

The DUT did not send **PullMessagesResponse** message(s).

The DUT did not send **UnsubscribeResponse** message.

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

6.8.11 Storage Failure event

Test Label: Storage Failure event

Test Case ID: DEVICE-9-1-11

ONVIF Core Specification Coverage: Monitoring Event Storage Failure

Command Under Test: GetServices, GetEventProperties, CreatePullPointSubscription, PullMessages, Device/HardwareFailure/StorageFailure event

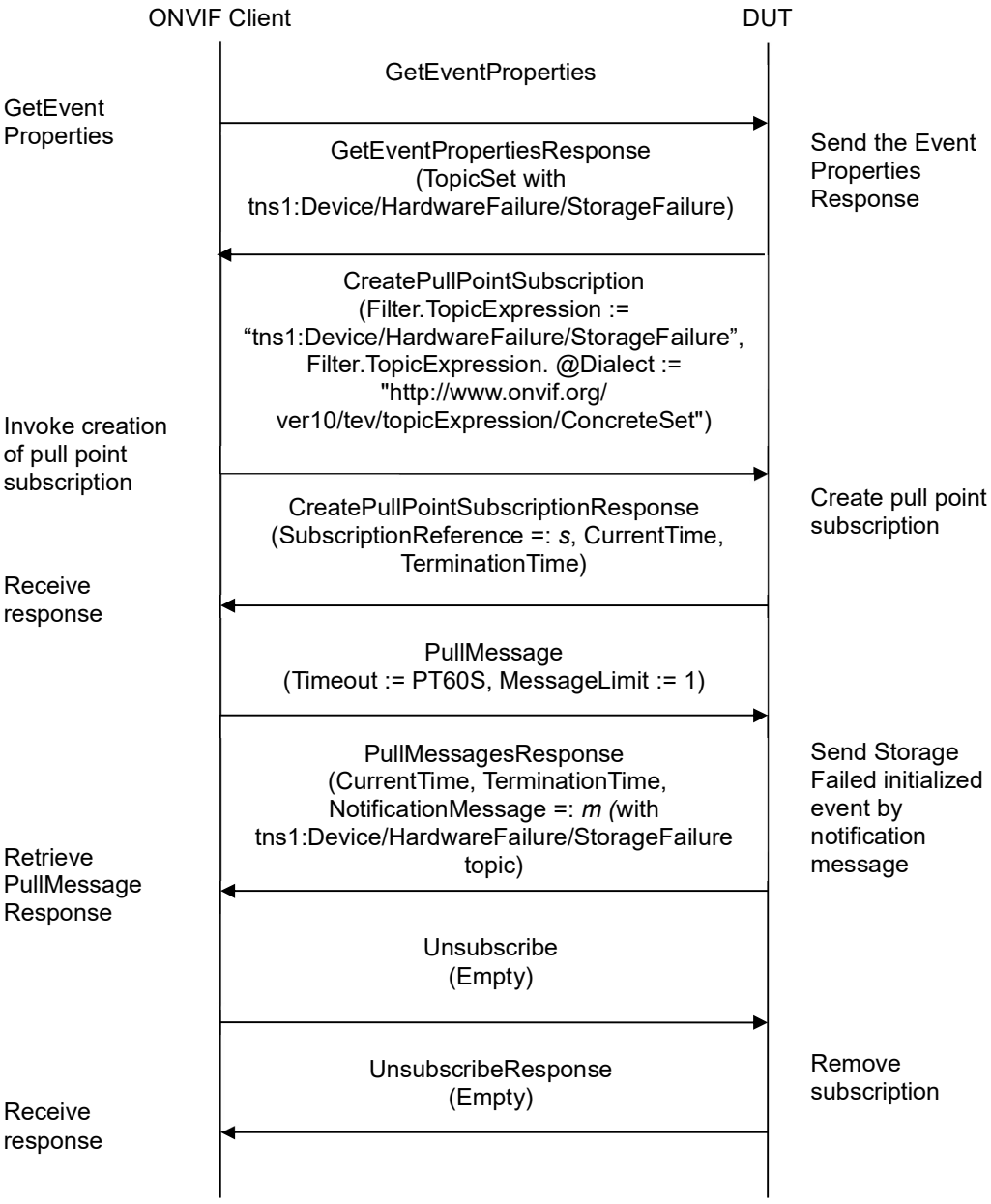
WSDL Reference: devicemgmt.wsdl and event.wsdl

Test Purpose: To verify tns1:Device/HardwareFailure/StorageFailure event generation and to verify tns1:Device/HardwareFailure/StorageFailure event format.

Pre-requisite: Event Service was received from the DUT. tns1:Device/HardwareFailure/StorageFailure event is supported by the DUT as indicated by the GetEventPropertiesResponse.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
    - TopicNamespaceLocation list

- FixedTopicSet
- TopicSet =: topicSet
- TopicExpressionDialect list
- MessageContentFilterDialect list
- MessageContentSchemaLocation list

5. If *topicSet* does not contain tns1:Device/HardwareFailure/StorageFailure topic, FAIL the test and skip other steps.
6. ONVIF Client verifies tns1:Device/HardwareFailure/StorageFailure topic (*storageFailureTopic*) from *topicSet*:
   6.1. If *storageFailureTopic*.MessageDescription.IsProperty is skipped or equals to false, FAIL the test and skip other steps.
   6.2. If *storageFailureTopic* does not contain MessageDescription.Source.SimpleItemDescription item with Name = “Token”, FAIL the test and skip other steps.
   6.3. If *storageFailureTopic*.MessageDescription.Source.SimpleItemDescription item with Name = “Token” does not have Type = “tt:ReferenceToken”, FAIL the test and skip other steps.
   6.4. If *storageFailureTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = “Failed”, FAIL the test and skip other steps.
   6.5. If *storageFailureTopic*.MessageDescription.Data.SimpleItemDescription item with Name = “Failed” does not have Type = “xs:boolean”, FAIL the test and skip other steps.
7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
   - Filter.TopicExpression := “tns1:Device/HardwareFailure/StorageFailure”
   - Filter.TopicExpression.@Dialect := “http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet”
8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
   - SubscriptionReference =: s
   - CurrentTime
   - TerminationTime
9. Until *timeout1* timeout expires, repeat the following steps:
   9.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
   - Timeout := PT60S
   - MessageLimit := 1

   9.2. The DUT responds with **PullMessagesResponse** message with parameters
   - CurrentTime
   - TerminationTime

- NotificationMessage =: *m*

9.3. If *m* is not null and *m*.Message.Message.PropertyOperation="Initialized" ONVIF Client verifies *m:*

9.3.1. If *m*.Topic does not equal to tns1:Device/HardwareFailure/StorageFailure, FAIL the test and go to the step 10.

9.3.2. If *m* does not contain Message.Message.Source.SimpleItem.Token, FAIL the test and go to the step 10.

9.3.3. If *m*.Message.Message.Source.SimpleItem.Token has value type different from tt:ReferenceToken type, FAIL the test and go to the step 10.

9.3.4. If *m* does not contain Message.Message.Data.SimpleItem.Failed, FAIL the test and go to the step 10.

9.3.5. If *m*.Message.Message.Data.SimpleItem.Failed has value type different from xs:boolean type, FAIL the test and go to the step 10.

9.3.6. Go to the step 10.

9.4. If *timeout1* timeout expires for step 9 without Notification with PropertyOperation="Initialized", FAIL the test and go to the step 10.

10. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

11. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

The DUT did not send **CreatePullPointSubscriptionResponse** message.

The DUT did not send **PullMessagesResponse** message(s).

The DUT did not send **UnsubscribeResponse** message.

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

6.8.12 Critical Temperature event

Test Label: Critical Temperature event

Test Case ID: DEVICE-9-1-12

ONVIF Core Specification Coverage: Monitoring Event Critical Temperature

Command Under Test: GetServices, GetEventProperties, CreatePullPointSubscription, PullMessages, Device/HardwareFailure/TemperatureCritical event

WSDL Reference: devicemgmt.wsdl and event.wsdl

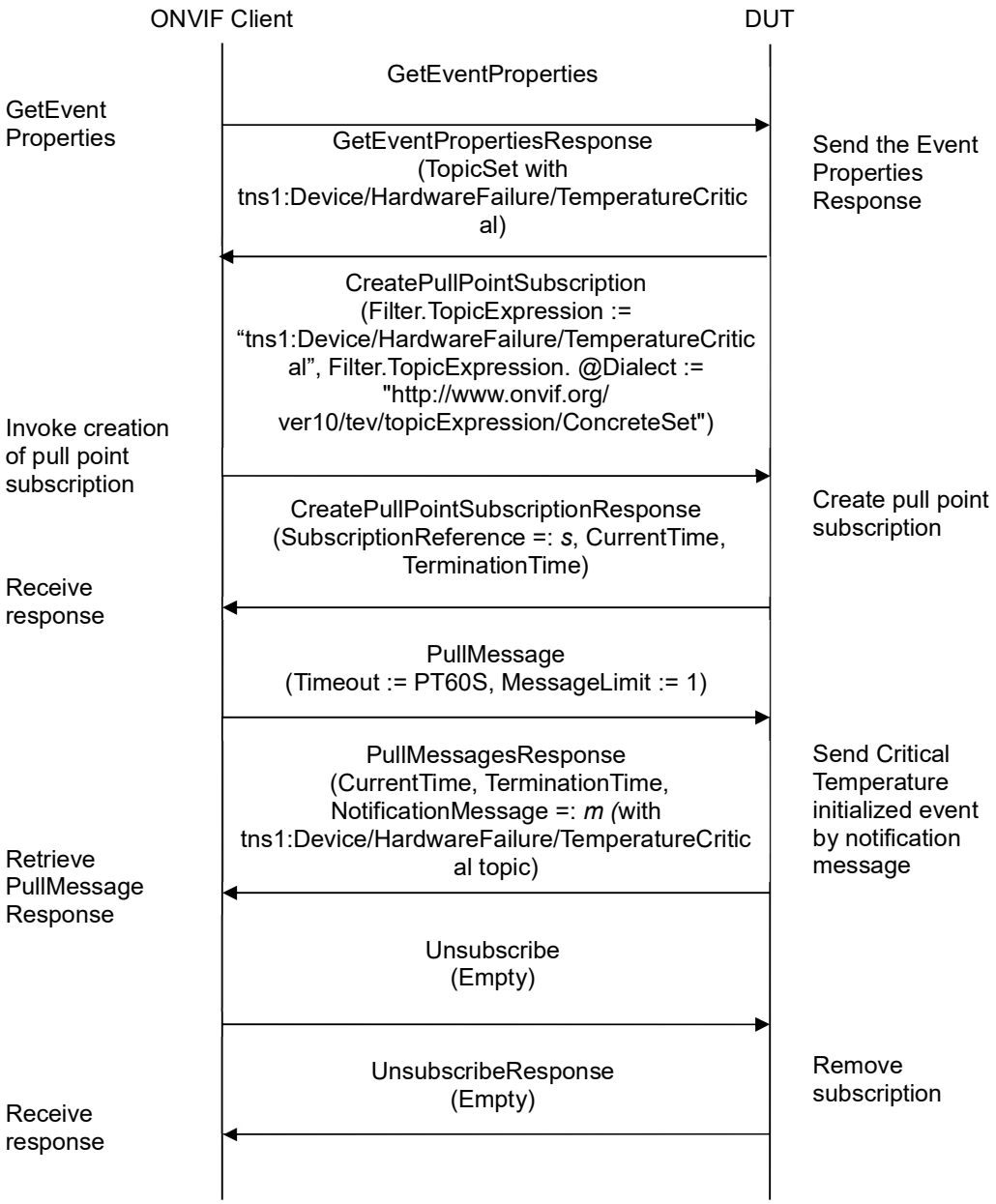
Driving IP-based physical security through global standardization

Test Purpose: To verify tns1:Device/HardwareFailure/TemperatureCritical event generation and to verify tns1:Device/HardwareFailure/TemperatureCritical event format.

Pre-requisite: Event Service was received from the DUT. tns1:Device/HardwareFailure/TemperatureCritical event is supported by the DUT as indicated by the GetEventPropertiesResponse.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
    - TopicNamespaceLocation list
    - FixedTopicSet
    - TopicSet =: topicSet
    - TopicExpressionDialect list
    - MessageContentFilterDialect list
    - MessageContentSchemaLocation list
5. If *topicSet* does not contain tns1:Device/HardwareFailure/TemperatureCritical topic, FAIL the test and skip other steps.
6. ONVIF Client verifies tns1:Device/HardwareFailure/TemperatureCritical topic (*criticalTemperatureTopic*) from *topicSet*:
    6.1. If *criticalTemperatureTopic*.MessageDescription.IsProperty is skipped or equals to false, FAIL the test and skip other steps.
    6.2. If *criticalTemperatureTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = “Critical”, FAIL the test and skip other steps.
    6.3. If *criticalTemperatureTopic*.MessageDescription.Data.SimpleItemDescription item with Name = “Critical” does not have Type = “xs:boolean”, FAIL the test and skip other steps.
7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
    - Filter.TopicExpression := “tns1:Device/HardwareFailure/TemperatureCritical”
    - Filter.TopicExpression.@Dialect := “http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet”
8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
    - SubscriptionReference =: s
    - CurrentTime
    - TerminationTime
9. Until *timeout1* timeout expires, repeat the following steps:
    9.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
    - Timeout := PT60S
    - MessageLimit := 1

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

- CurrentTime
- TerminationTime
- NotificationMessage =: *m*

9.3. If *m* is not null and *m*.Message.Message.PropertyOperation="Initialized" ONVIF Client verifies *m:*

9.3.1. If *m*.Topic does not equal to tns1:Device/HardwareFailure/TemperatureCritical, FAIL the test and go to the step 10.

9.3.2. If *m* does not contain Message.Message.Data.SimpleItem.Critical, FAIL the test and go to the step 10.

9.3.3. If *m*.Message.Message.Data.SimpleItem.Critical has value type different from xs:boolean type, FAIL the test and go to the step 10.

9.3.4. Go to the step 10.

9.4. If *timeout1* timeout expires for step 9 without Notification with PropertyOperation="Initialized", FAIL the test and go to the step 10.

10. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

11. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

The DUT did not send **CreatePullPointSubscriptionResponse** message.

The DUT did not send **PullMessagesResponse** message(s).

The DUT did not send **UnsubscribeResponse** message.

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

7 Event Handling Test Cases

7.1 Event Properties

7.1.1 GET EVENT PROPERTIES

Test Label: Event handling GET EVENT PROPERTIES

Test Case ID: EVENT-1-1-2

ONVIF Core Specification Coverage: GetEventProperties

Command Under Test: GetEventProperties

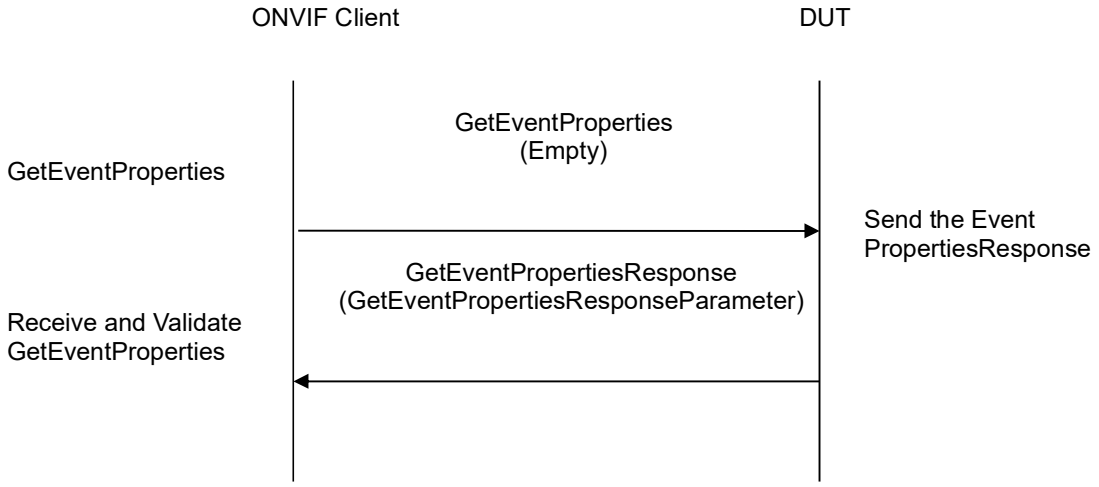
WSDL Reference: event.wsdl

Test Purpose: To verify DUT GetEventProperties command

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetEventPropertiesRequest message to retrieve information about the FilterDialects, schema files and supported topics of the DUT.
4. Verify that DUT sends GetEventPropertiesResponse message
5. Validate that the mandatory TopicExpressionDialects are supported by the DUT (http://docs.oasis-open.org/wsn/t-1/TopicExpression/Concrete and http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet)

6. Validate that the mandatory MessageContentFilterDialects is supported by the DUT (http://www.onvif.org/ver10/tev/messageContentFilter/ItemFilter)
7. Verify that the DUT returns a valid topic namespace
8. Verify that the DUT supports at least one TopicSet, validate that the TopicSet is well formed.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send a GetEventPropertiesResponse message.

The DUT did not support the mandatory TopicExpressionDialects

The DUT did not support the mandatory MessageContentFilterDialects

The DUT did not support a valid topic namespace

The DUT did not support at least one TopicSet or the TopicSet is invalid

The DUT did not send a valid WS-Addressing Action URI in SOAP Header for GetEventPropertiesResponse message (see Annex A.17).

7.2 Basic Notification Interface

7.2.1 BASIC NOTIFICATION INTERFACE - SUBSCRIBE

Test Label: Event handling SUBSCRIBE

Test Case ID: EVENT-2-1-9

ONVIF Core Specification Coverage: Basic Notification Interface

Command Under Test: Subscribe

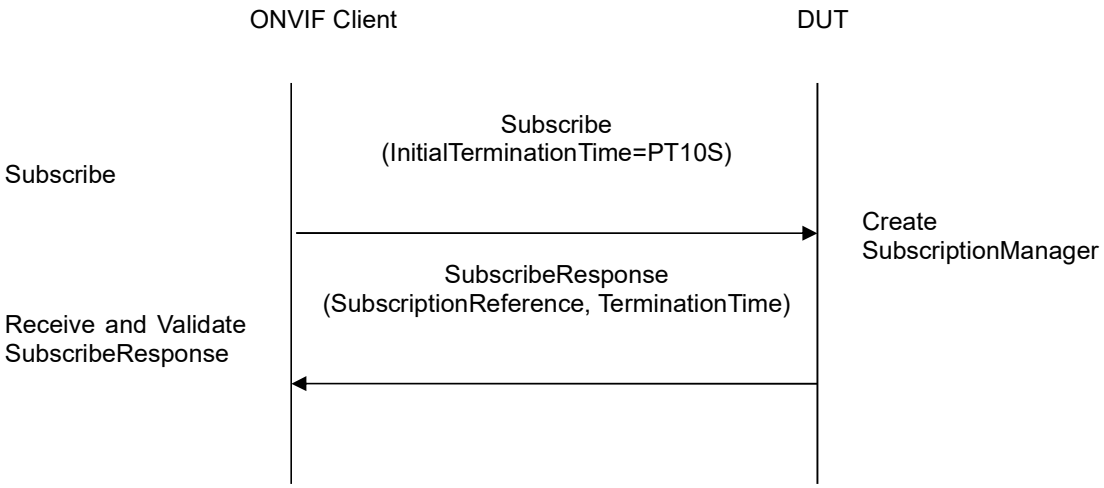
WSDL Reference: event.wsdl

Test Purpose: To verify DUT Subscribe command

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke Subscribe message to instantiate a Subscription Manager. The Subscribe message does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.
4. Verify that the DUT sends SubscribeResponse message. Verify that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SubscribeResponse message.

The DUT did not return a valid SubscriptionReference

The DUT did not return valid values for CurrentTime and TerminationTime.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for SubscribeResponse message (see Annex A.17).

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

If the DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value included in UnacceptableInitialTerminationTimeFault.

7.2.2 BASIC NOTIFICATION INTERFACE - RENEW

Test Label: Event handling Renew

Test Case ID: EVENT-2-1-12

ONVIF Core Specification Coverage: Basic Notification Interface

Command Under Test: Subscribe, Renew

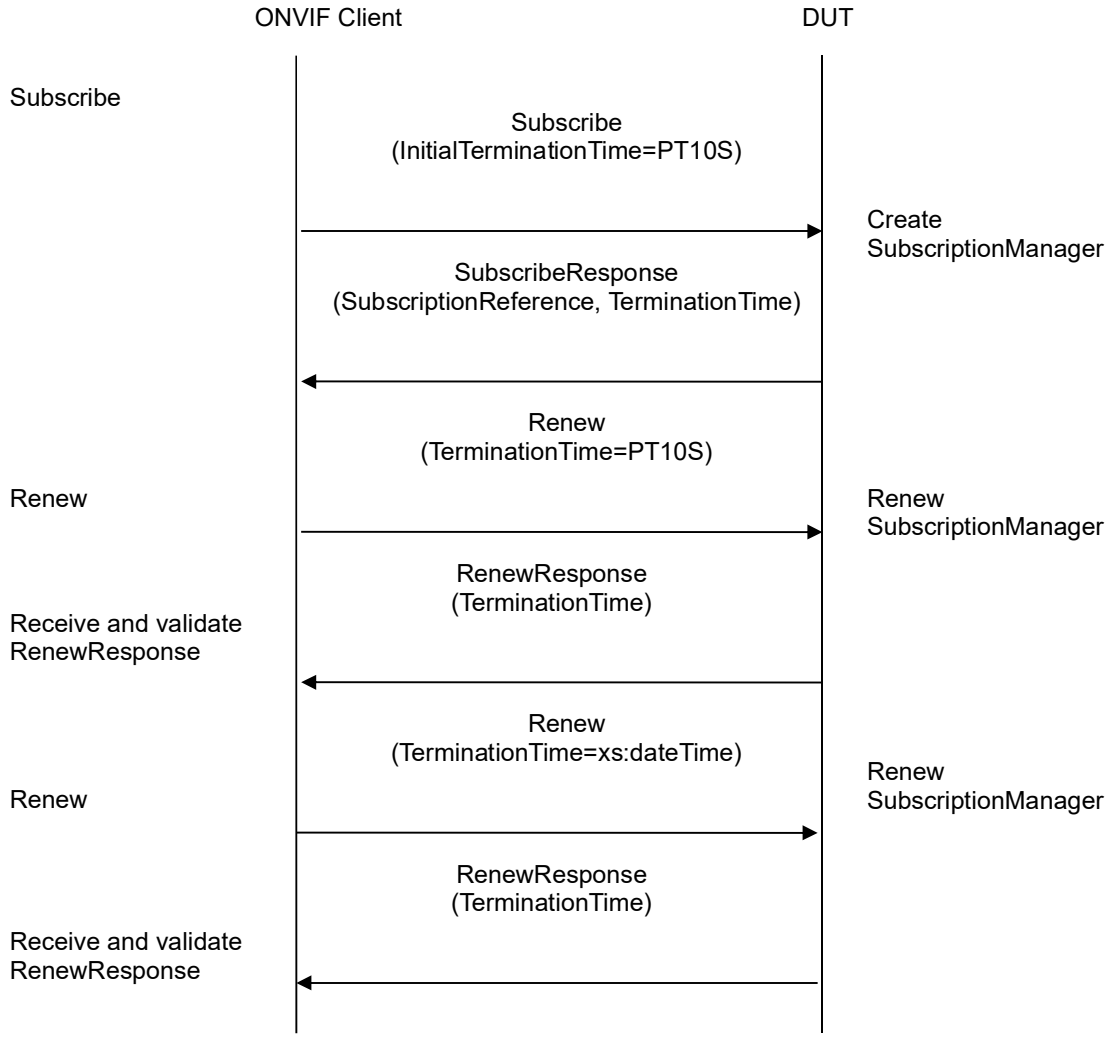
WSDL Reference: event.wsdl

Test Purpose: To verify Renew command

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke Subscribe message with an InitialTerminationTime of 10s.
4. Verify that the DUT sends a SubscribeResponse.
5. Validate CurrentTime and TerminationTime (TerminationTime >= CurrentTime + InitialTerminationTime)
6. ONVIF Client will invoke Renew command with a TerminationTime of 10s to ensure that the Subscription times out after 10s.
7. Verify that the DUT sends a RenewResponse

8. Verify CurrentTime and TerminationTime (TerminationTime>=CurrentTime+ TerminationTime)
9. ONVIF Client will invoke Renew command with a TerminationTime in xs:dateTime format. The TerminationTime shall be current time+ 10s.
10. Verify that the DUT sends a RenewResponse
11. Verify CurrentTime and TerminationTime (TerminationTime(Response)>=TerminationTime(Request) and CurrentTime(Response)<=TerminationTime(Response))

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SubscribeResponse message.

The DUT did not send valid values for CurrentTime and TerminationTime.

The DUT did not send a RenewResponse

The DUT did not send valid values for CurrentTime and TerminationTime

The DUT did not send valid WS-Addressing Action URI in SOAP Header for SubscribeResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for RenewResponse message (see Annex A.17).

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

If DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value which is contained in UnacceptableInitialTerminationTimeFault.

If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the Renew request MinimumTime value included in UnacceptableTerminationTimeFault.

7.2.3 BASIC NOTIFICATION INTERFACE - NOTIFY

Test Label: Event handling NOTIFY

Test Case ID: EVENT-2-1-17

ONVIF Core Specification Coverage: Basic Notification Interface, SetSynchronizationPoint

Command Under Test: Subscribe, Unsubscribe, SetSynchronizationPoint, Notify

WSDL Reference: event.wsdl

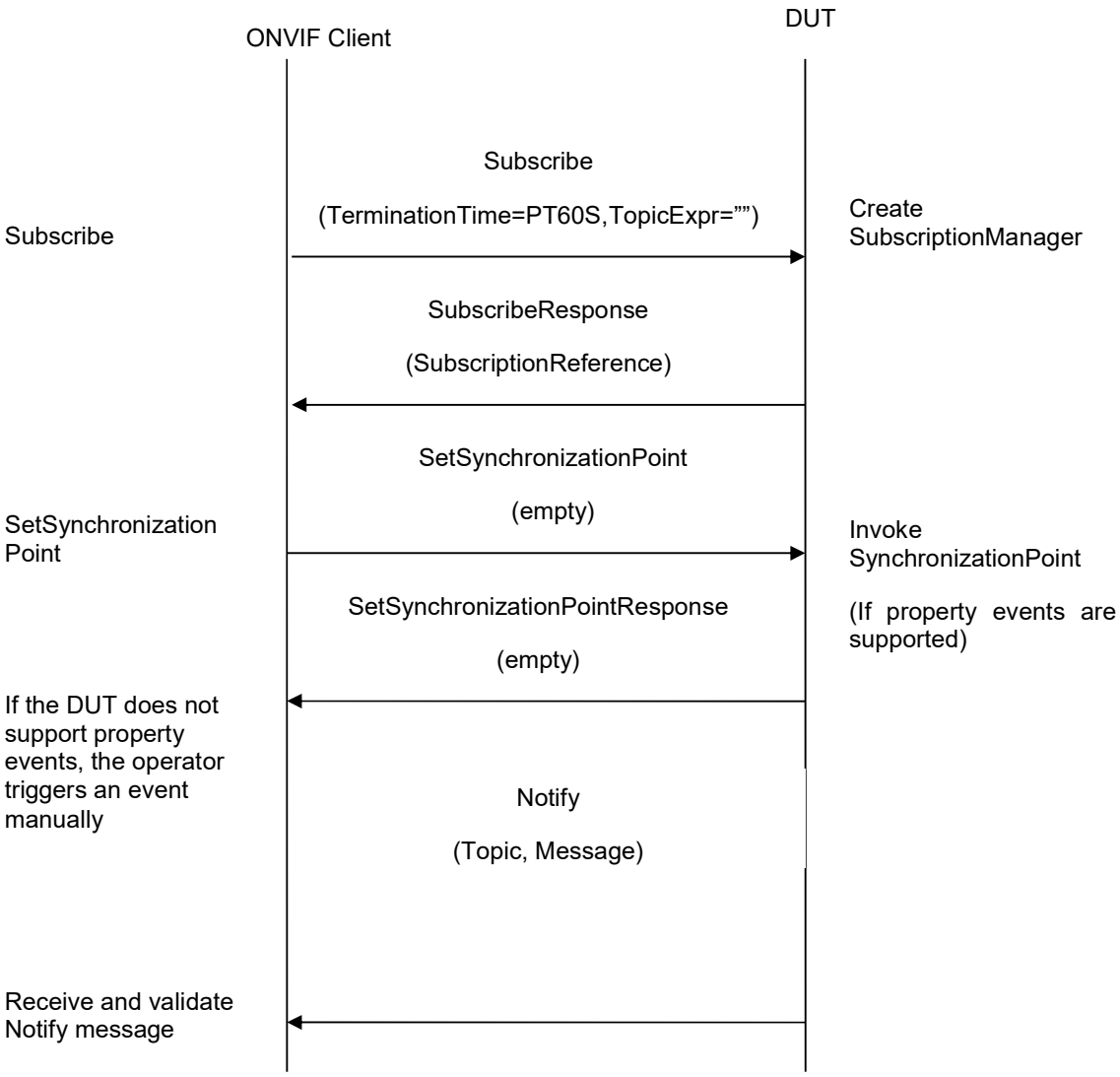
Test Purpose: To verify Notify message

Pre-Requisite: The DUT shall provide at least one event.

The test operator has to ensure that the event is triggered and sent out. ONVIF Client will invoke a SetSynchronizationPoint request. If the DUT does not support property events or if it is not possible to invoke a SetSynchronizationPoint, the test operator has to trigger the event manually.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

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

3. ONVIF Client will invoke Subscribe with an InitialTerminationTime of 60s to ensure that the SubscriptionManager is deleted after one minute.
4. Verify that the DUT sends a SubscribeResponse with valid values for SubscriptionReference, TerminationTime and CurrentTime TerminationTime >= CurrentTime + InitialTerminationTime.
5. Invoke SetSynchronizationPoint command to trigger a property event.
6. Verify that DUT sends SetSynchronizationPointResponse.
7. If the DUT does not support property events, the test operator has to trigger an event manually.
8. Verify that DUT sends Notify message(s).
9. Verify received Notify messages (correct value for UTC time, TopicExpression and wsnt:Message).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SubscribeResponse message.

The DUT did not send valid SubscriptionReference.

The DUT did not send a SetSynchronizationPointResponse

The DUT did not a Notify message

The DUT sends an invalid NOTIFY message

DUT did not send valid WS-Addressing Action URI in SOAP Header for SubscribeResponse message (see Annex A.17).

DUT did not send valid WS-Addressing Action URI in SOAP Header for SetSynchronizationPointResponse message (see Annex A.17).

DUT did not send valid WS-Addressing Action URI in SOAP Header for Notify message (see Annex A.17).

DUT did not send WS-Addressing MessageID SOAP Header if WS-Addressing ReplyTo is included in Notify message.

DUT sent invalid WS-Addressing MessageID.

DUT use wrong HTTP address for Notify message.

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

See Annex A.9 for instructions on how to construct Subscribe when it is required for the ONVIF Client to receive all events supported by the DUT.

7.2.4 BASIC NOTIFICATION INTERFACE - NOTIFY FILTER

Test Label: Event handling NOTIFY FILTER

Test Case ID: EVENT-2-1-18

ONVIF Core Specification Coverage: Basic Notification Interface, SetSynchronizationPoint

Command Under Test: GetEventProperties, Subscribe, Unsubscribe, SetSynchronizationPoint, Notify

WSDL Reference: event.wsdl

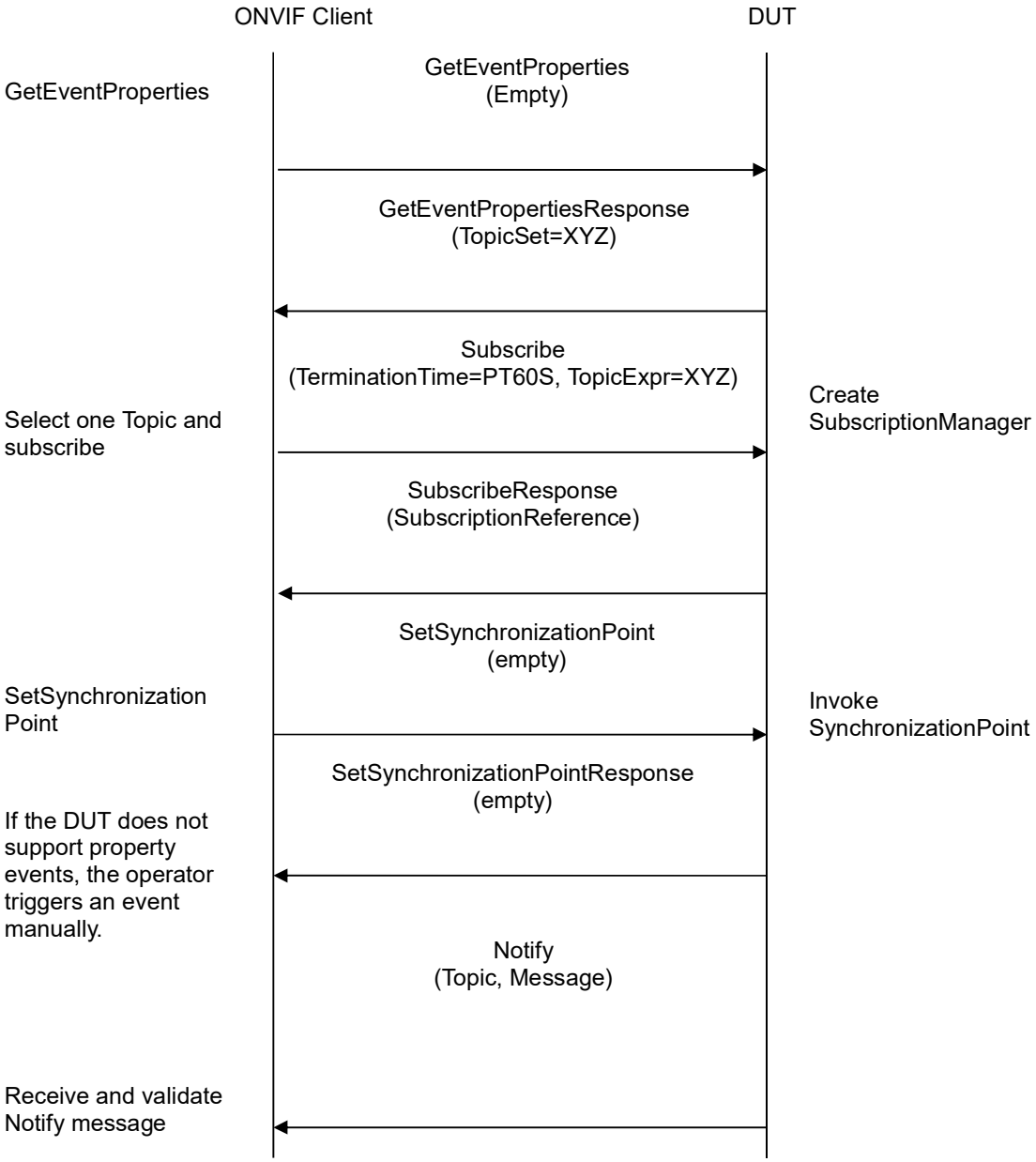
Test Purpose: To verify that the device sends Notification messages; to verify if the DUT handles event filtering in a correct way.

Pre-Requisite: The DUT shall provide at least one event.

The test operator has to ensure that the event is triggered and sent out. ONVIF Client will invoke a SetSynchronizationPoint request. If the DUT does not support property event or if it is not possible to invoke a SetSynchronizationPoint, the test operator has to trigger the event manually.

Test Configuration: ONVIF Client and DUT

Test Sequence:



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, select one Topic.
5. ONVIF Client will invoke Subscribe with this Topic as Filter and an InitialTerminationTime of 60s to ensure that the SubscriptionManager is deleted after one minute.
6. Verify that the DUT sends a SubscribeResponse with valid values for SubscriptionReference, TerminationTime and CurrentTime TerminationTime>=CurrentTime+60S.
7. Invoke SetSynchronizationPoint command to trigger an event.
8. Verify that the DUT sends SetSynchronizationPointResponse.
9. If the DUT does not support property events, the operator has to trigger an event manually.
10. Verify that the DUT sends Notify message(s).
11. Check that at least one Notify message that contains a property event is returned. Verify this Notify messages (correct value for Utc time, TopicExpression and wsnt:Message).
12. Check if Notify message(s) are filtered according to the selected Filter.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send a GetEventPropertiesResponse

The DUT did not send SubscribeResponse message.

The DUT did not send valid SubscriptionReference.

The DUT did not send a SetSynchronizationPointResponse

The DUT did not a Notify message that contains a property event

The DUT send an invalid NOTIFY message

The DUT did not send valid WS-Addressing Action URI in SOAP Header for GetEventPropertiesResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for SubscribeResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for SetSynchronizationPointResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for Notify message (see Annex A.17).

DUT did not send WS-Addressing MessageID in SOAP Header if WS-Addressing ReplyTo is included in Notify message.

DUT sent invalid WS-Addressing MessageID in SOAP Header in Notify message.

DUT used wrong HTTP address for Notify message.

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

7.2.5 BASIC NOTIFICATION INTERFACE - INVALID MESSAGE CONTENT FILTER

Test Label: Event handling SUBSCRIBE INVALID MESSAGE CONTENT FILTER

Test Case ID: EVENT-2-1-19

ONVIF Core Specification Coverage: Basic Notification Interface

Command Under Test: Subscribe

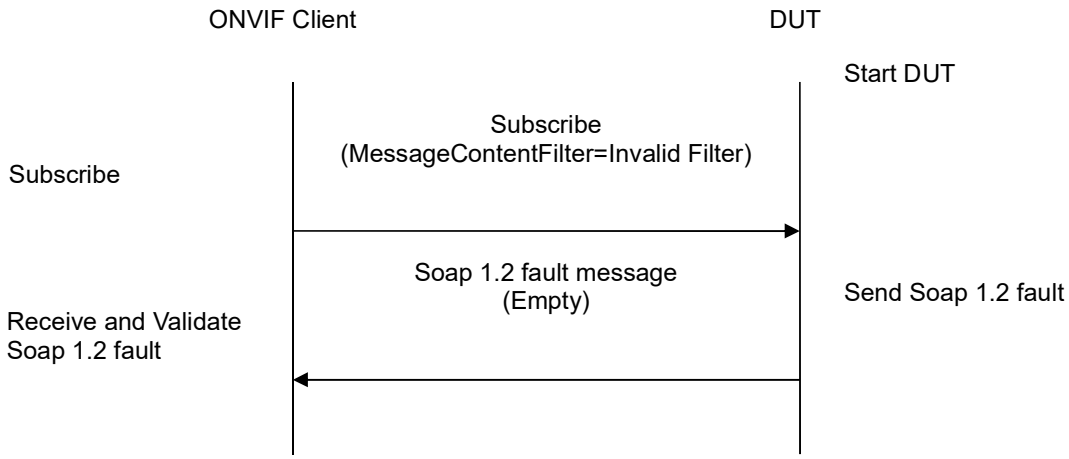
WSDL Reference: event.wsdl

Test Purpose: To verify that a correct error message "InvalidFilterFault" or “InvalidMessageContentExpressionFault” is returned if a Subscribe Request with an invalid MessageContentFilter is invoked.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke Subscribe message with an invalid Filter (boolean (//tt:SimpleItem[@Name=”xyz” and @Value=”xyz”])).
4. Verify that the DUT generates an “InvalidFilterFault” or an “InvalidMessageContentExpressionFault” fault message.
5. Validate the fault message (valid UTC time, valid description).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send an “InvalidFilterFault” or “InvalidMessageContentExpressionFault” fault message.

The DUT did not send a valid fault message.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for Fault message (see Annex A.17).

7.2.6 BASIC NOTIFICATION INTERFACE - UNSUBSCRIBE

Test Label: Event handling UNSUBSCRIBE

Test Case ID: EVENT-2-1-21

ONVIF Core Specification Coverage: Basic Notification Interface

Command Under Test: Subscribe, Unsubscribe

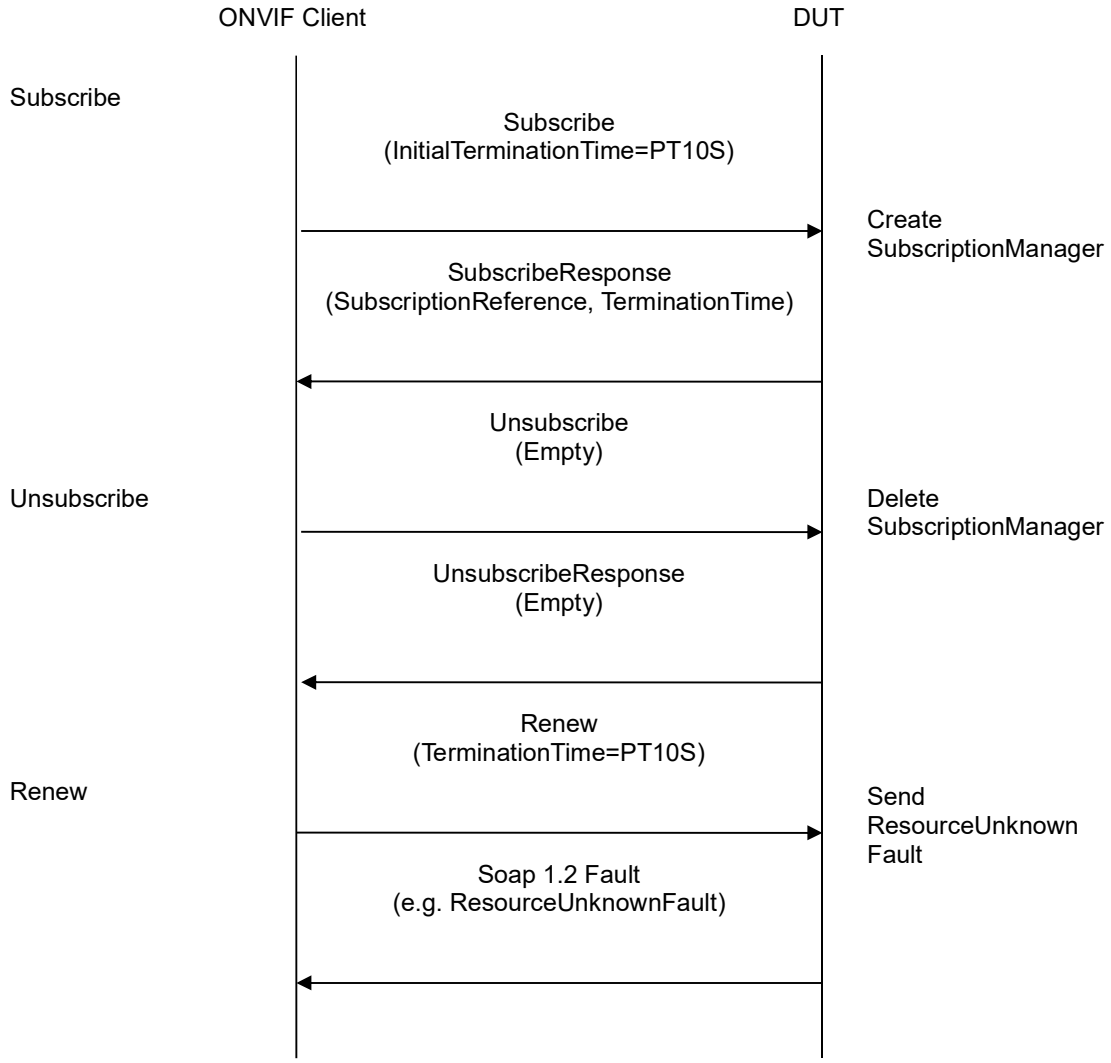
WSDL Reference: event.wsdl

Test Purpose: To verify Unsubscribe command.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke Subscribe message (InitialTerminationTime=PT10S) to instantiate a SubscriptionManager.
4. Verify that the DUT sends a SubscribeResponse with valid values for SubscriptionManager, CurrentTime and TerminationTime.
5. ONVIF Client will invoke Unsubscribe command.
6. Verify that the DUT sends an UnsubscribeResponse.
7. ONVIF Client will invoke a Renew command to verify that the SubscriptionManager is deleted.

8. Verify that the DUT sends a Soap 1.2 Fault (e.g. a “ResourceUnknown” fault message).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SubscribeResponse message.

The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime >= CurrentTime + InitialTerminationTime).

The DUT did not send an UnsubscribeResponse.

The DUT did not send a Soap 1.2 fault message.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for SubscribeResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for UnsubscribeResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for Fault message (see Annex A.17).

Note: If the DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value included in UnacceptableInitialTerminationTimeFault.

7.2.7 BASIC NOTIFICATION INTERFACE - RESOURCE UNKNOWN

Test Label: Event handling RESOURCE UNKNOWN

Test Case ID: EVENT-2-1-22

ONVIF Core Specification Coverage: Basic Notification Interface

Command Under Test: Subscribe, Unsubscribe

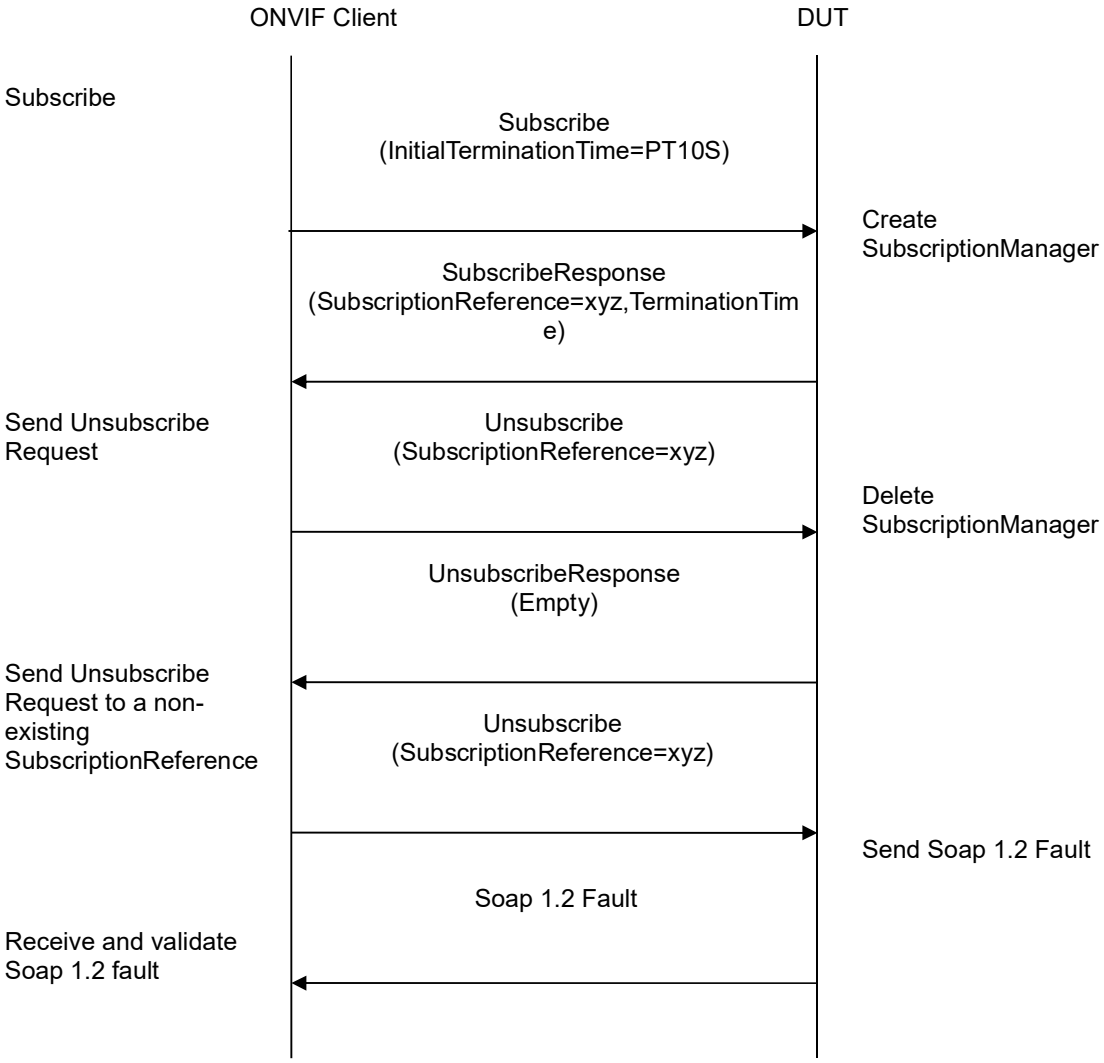
WSDL Reference: event.wsdl

Test Purpose: To verify that a Soap 1.2 Fault Message is returned if an UnsubscribeRequest to a non-existing Subscription is sent.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke Subscribe message with an InitialTerminationTime of 10 s to ensure that the SubscriptionManager will be deleted after testing.
4. Verify that the DUT sends a SubscribeResponse with valid values for SubscriptionReference and TerminationTime).
5. ONVIF Client will invoke Unsubscribe command to delete the SubscriptionManager.
6. Verify that the DUT sends an UnsubscribeResponse.

7. Send the second Unsubscribe Request to the just deleted SubscriptionReference.
8. Verify that the DUT sends a Soap 1.2 Fault (e.g. a “ResourceUnknown” or a “UnableToDestroySubscription” Fault).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SubscribeResponse message.

The DUT did not send valid SubscriptionReference.

The DUT did not send an UnsubscribeResponse.

The DUT did not delete the SubscriptionManager.

The DUT did not send a Soap 1.2 Fault.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for SubscribeResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for UnsubscribeResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for Fault message (see Annex A.17).

Note: If DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value included in UnacceptableInitialTerminationTimeFault.

7.2.8 BASIC NOTIFICATION INTERFACE - INVALID TOPIC EXPRESSION

Test Label: Event handling SUBSCRIBE INVALID FILTER-TOPIC EXPRESSION

Test Case ID: EVENT-2-1-23

ONVIF Core Specification Coverage: Basic Notification Interface

Command Under Test: Subscribe

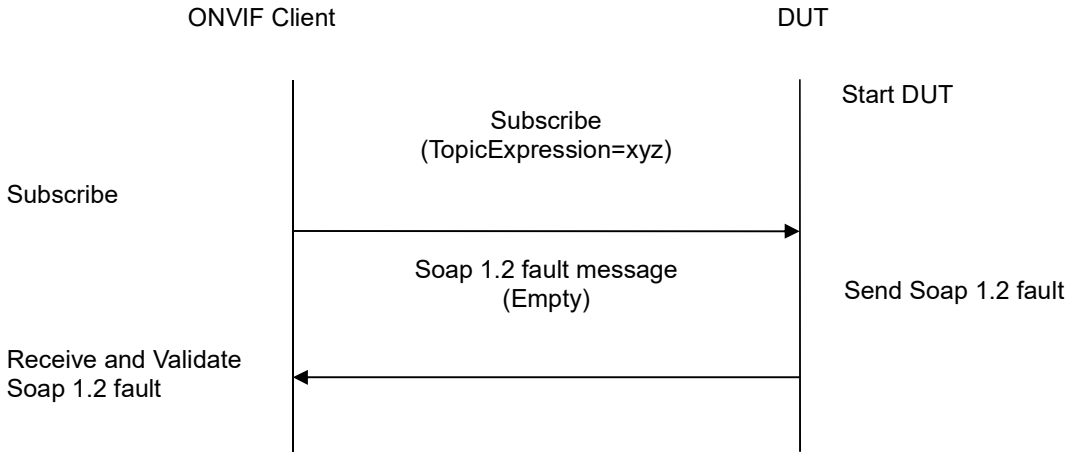
WSDL Reference: event.wsdl

Test Purpose: To verify that a correct error message "InvalidFilterFault" or “TopicNotSupported” or “InvalidTopicExpressionFault” is returned if a Subscribe Request with an invalid Topic Expression is invoked.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke Subscribe message with an invalid Topic Expression (boolean(//tt:SimpleItem[@Name="xyz" and @Value="xyz"])).
4. Verify that the DUT generates an "InvalidFilterFault" or "TopicNotSupported" or "InvalidTopicExpressionFault" fault message.
5. Validate the fault message (valid UTC time, valid description).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send an "InvalidFilterFault" or "TopicNotSupported" or "InvalidTopicExpressionFault" fault message.

The DUT did not send a valid fault message.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for Fault message (see Annex A.17).

7.2.9 BASIC NOTIFICATION INTERFACE - SET SYNCHRONIZATION POINT

Test Label: Event Handling BASIC NOTIFICATION SET SYNCHRONIZATION POINT

Test Case ID: EVENT-2-1-24

ONVIF Core Specification Coverage: Basic Notification Interface, SetSynchronizationPoint

Command Under Test: GetEventProperties, Subscribe, SetSynchronizationPoint, Notify

WSDL Reference: event.wsdl

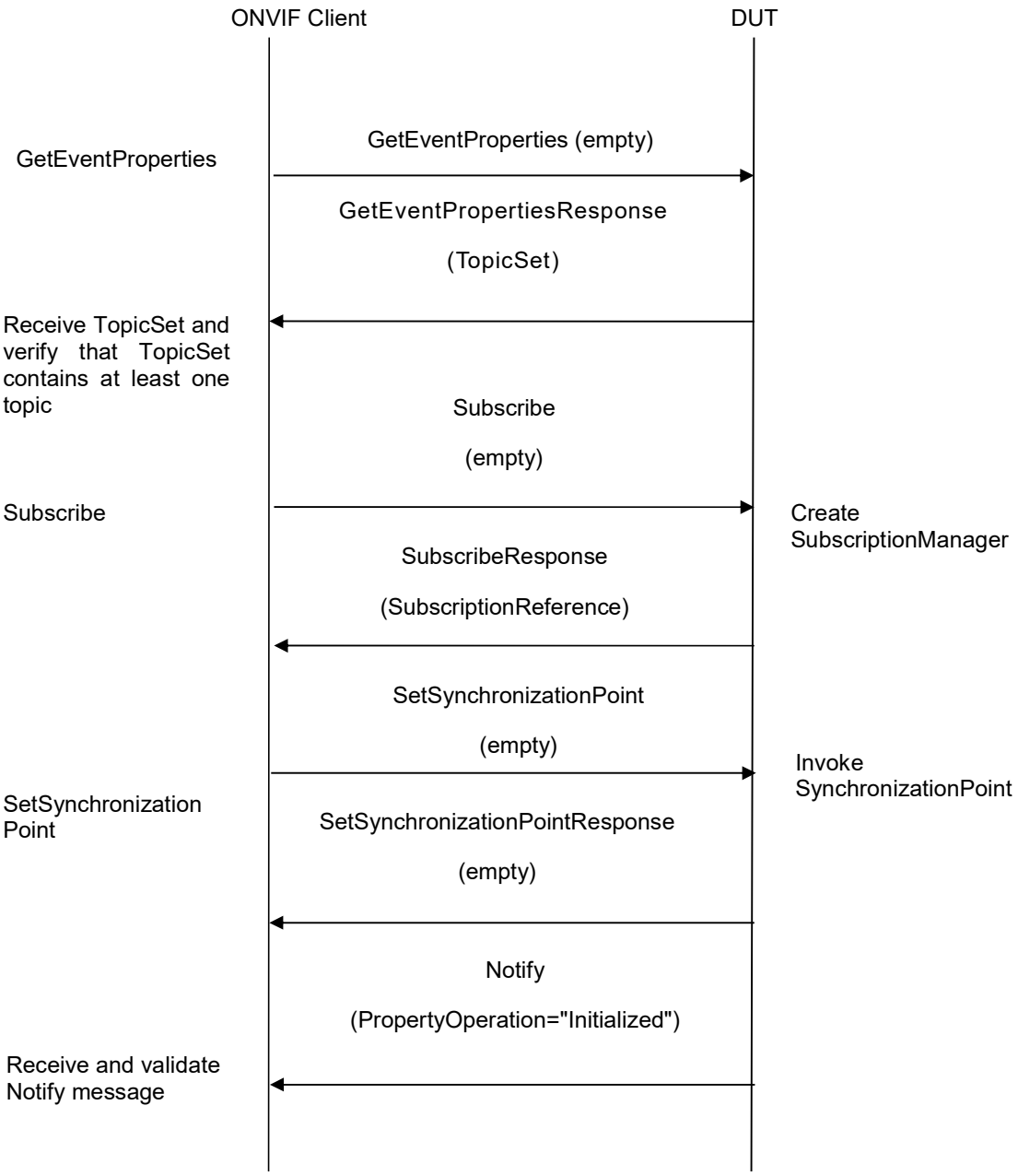
Test Purpose: To verify that the DUT sends all currently alive properties to the Client when SetSynchronizationPoint operation is invoked.

Pre-Requisite:

The test operator has to ensure that the event is triggered and sent out. ONVIF Client will invoke a SetSynchronizationPoint request. The DUT shall produce at least one property notification with the attribute "PropertyOperation" set to the "Initialized" state.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetEventProperties command.
4. Verify that DUT sends GetEventPropertiesResponse with TopicSet and TopicSet contains at least one topic. If TopicSet is empty then skip other steps.

5. ONVIF Client will invoke Subscribe operation.
6. Verify that the DUT sends a SubscribeResponse.
7. Invoke SetSynchronizationPoint command to trigger a property event.
8. Verify that DUT sends SetSynchronizationPointResponse.
9. Verify that DUT sends Notify message(s).
10. Verify received Notify message(s)to ensure that this is correct property notification message with attribute PropertyOperation="Initialized".

Test Result:

PASS –

The DUT passed all assertions.

DUT returned GetEventPropertiesResponse with empty TopicSet.

FAIL –

The DUT did not send SubscribeResponse message.

The DUT did not send valid SubscriptionReference.

The DUT did not send a SetSynchronizationPointResponse

The DUT did not a Notify message

The DUT sends an invalid NOTIFY message (without attribute PropertyOperation="Initialized")

DUT did not send valid WS-Addressing Action URI in SOAP Header for SubscribeResponse message (see Annex A.17).

DUT did not send valid WS-Addressing Action URI in SOAP Header for SetSynchronizationPointResponse message (see AnnexA.17).

DUT did not send valid WS-Addressing Action URI in SOAP Header for Notify message (see AnnexA.17).

DUT did not send WS-Addressing MessageID SOAP Header if WS-Addressing Reply To is included in Notify message.

DUT sent invalid WS-Addressing MessageID.

DUT use wrong HTTP address for Notify message.

7.3 Real-Time Pull-Point Notification Interface

7.3.1 REALTIME PULLPOINT SUBSCRIPTION - CREATE PULL POINT SUBSCRIPTION

Test Label: event handling CREATE PULL POINT SUBSCRIPTION

Test Case ID: EVENT-3-1-9

ONVIF Core Specification Coverage: CreatePullPointSubscription

Command Under Test: CreatePullPointSubscription

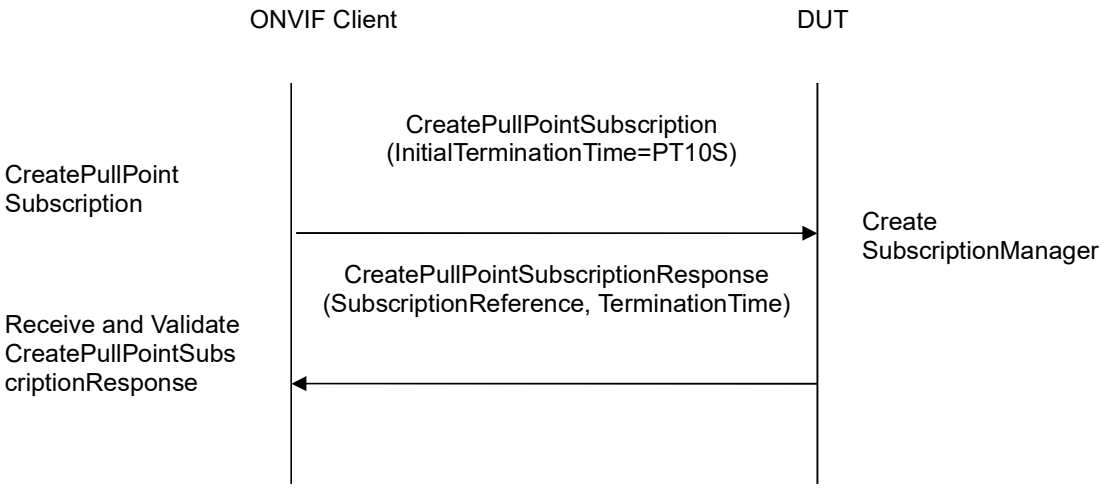
WSDL Reference: event.wsdl

Test Purpose: To verify the DUT CreatePullPointSubscription command

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke CreatePullPointSubscription message to instantiate a Subscription Manager. The CreatePullPointSubscription message does not contain a TopicExpression or Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the SubscriptionManager is terminated after end of this test.
4. Verify that the DUT sends CreatePullPointSubscriptionResponse message
5. Validate that valid values for SubscriptionReference CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime)

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not return a valid SubscriptionReference

The DUT did not return valid values for CurrentTime and TerminationTime.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for CreatePullPointSubscriptionResponse message (see Annex A.17).

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

If DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the CreatePullPointSubscription request with MinimumTime value included in UnacceptableInitialTerminationTime fault.

7.3.2 REALTIME PULLPOINT SUBSCRIPTION - RENEW

Test Label: Event handling RealtimePullPoint Renew

Test Case ID: EVENT-3-1-12

ONVIF Core Specification Coverage: Basic Notification Interface, CreatePullPointSubscription

Command Under Test: CreatePullPointSubscription, Renew

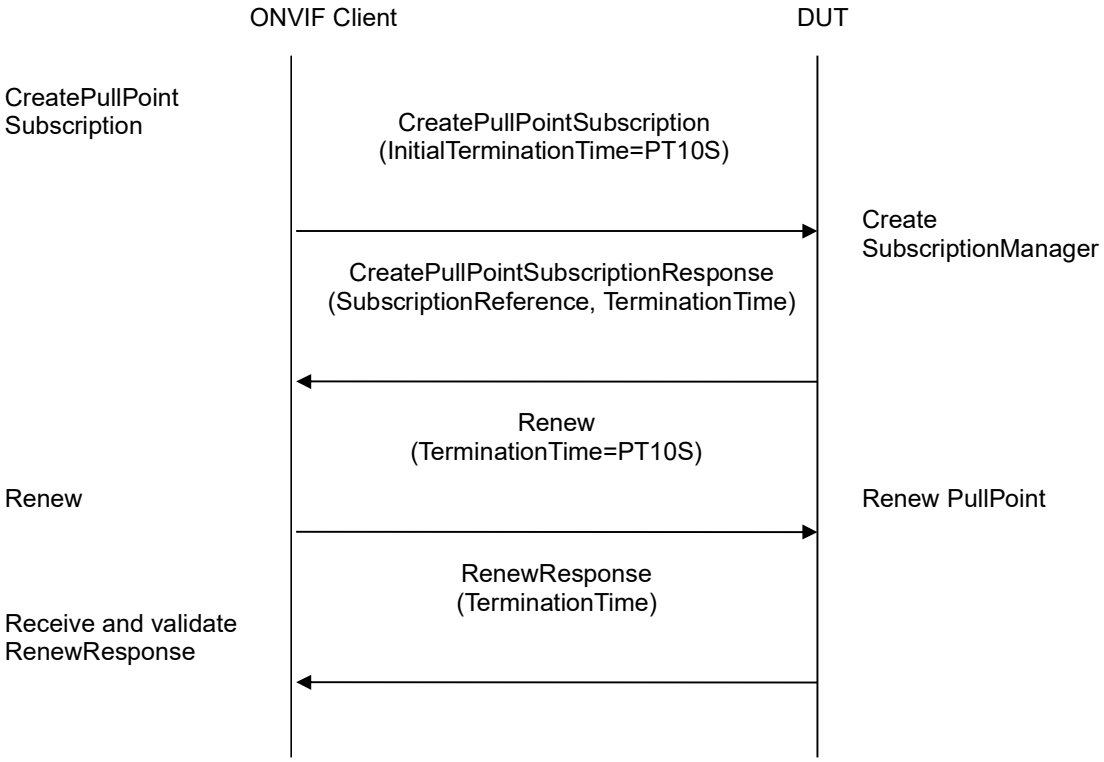
WSDL Reference: event.wsdl

Test Purpose: To verify Renew command

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke CreatePullPointSubscription message with an InitialTerminationTime of 10s
4. Verify that the DUT sends a CreatePullPointSubscriptionResponse.
5. Validate CurrentTime and TerminationTime (TerminationTime>=CurrentTime+10s)
6. ONVIF Client will invoke Renew command with a TerminationTime of 10s to ensure that the Subscription times out after the test
7. Verify that the DUT sends a RenewResponse
8. Verify CurrentTime and TerminationTime (TerminationTime>=CurrentTime+10s)

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SubscribeResponse message.

The DUT did not send valid values for CurrentTime and TerminationTime.

The DUT did not send a RenewResponse

The DUT did not send valid values for CurrentTime and TerminationTime

DUT did not send valid WS-Addressing Action URI in SOAP Header for CreatePullPointSubscriptionResponse message (see Annex A.17).

DUT did not send valid WS-Addressing Action URI in SOAP Header for RenewResponse message (see Annex A.17).

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

If DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the CreatePullPointSubscription request with MinimumTime value included in UnacceptableInitialTerminationTime fault.

7.3.3 REALTIME PULLPOINT SUBSCRIPTION - PULLMESSAGES

Test Label: event handling REALTIME PULL POINT INTERFACE PullMessages

Test Case ID: EVENT-3-1-15

ONVIF Core Specification Coverage: CreatePullPointSubscription, SetSynchronizationPoint, PullMessages

Command Under Test: CreatePullPointSubscription, SetSynchronizationPoint, PullMessages

WSDL Reference: event.wsdl

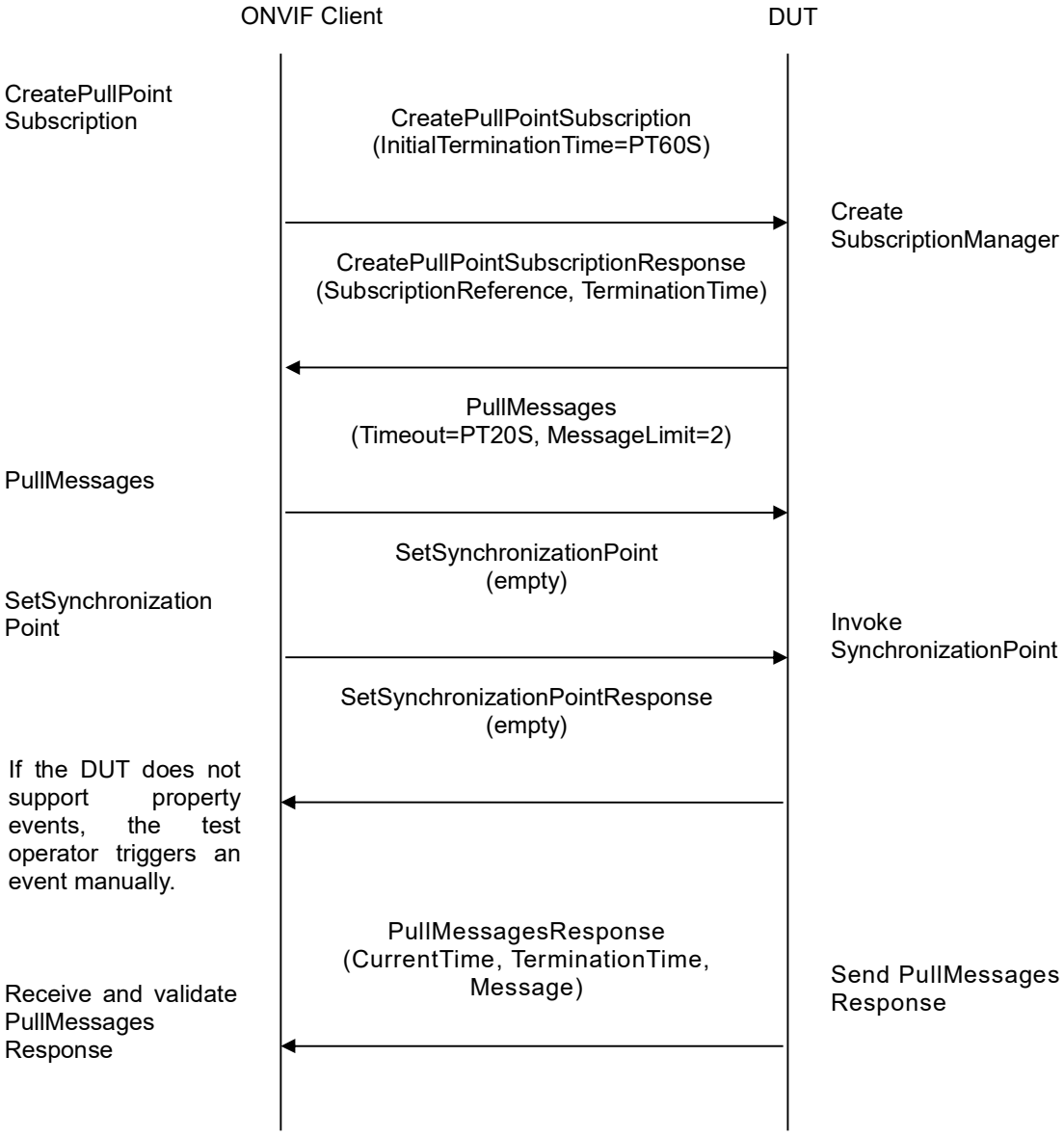
Test Purpose: To verify PullMessages command

Pre-Requisite: The DUT shall provide at least one event.

The test operator has to ensure that the event is triggered and sent out. ONVIF Client will invoke a SetSynchronizationPoint request. If the device does not support property events or if it is not possible to invoke a SetSynchronizationPoint, the test operator has to trigger event manually.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke CreatePullPointSubscription message with a suggested timeout of PT60S.
4. Verify that the DUT sends a CreatePullPointSubscriptionResponse. Validate that correct values for CurrentTime and TerminationTime and SubscriptionReference are returned

5. ONVIF Client will invoke PullMessages command with a PullMessagesTimeout of 20s and a MessageLimit of 2
6. ONVIF Client will invoke SetSynchronizationPoint command to trigger an event
7. Verify that the DUT sends a SetSynchronizationPointResponse
8. If the DUT does not support property events, the operator has to trigger an event manually.
9. Verify that the DUT sends a PullMessagesResponse that contains at least one NotificationMessage that represents a property.
10. Verify NotificationMessage (a maximum number of 2 Notification Messages is included in the PullMessages Response; well formed and valid values for CurrentTime and TerminationTime (TerminationTime>CurrentTime)

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).

The DUT did not send a SetSynchronizationPointResponse.

The DUT did not send a PullMessagesResponse.

The PullMessagesResponse does not contain a NotificationMessage.

The PullMessagesResponse contains more than 2 NotificationMessages.

The NotificationMessages are not well formed.

The PullMessagesResponse contains invalid values for Current or TerminationTime.

The DUT did not send a PullMessagesFaultResponse.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for CreatePullPointSubscriptionResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for SetSynchronizationPointResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for PullMessagesResponse message (see Annex A.17).

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

It may be possible that some other events than the event which is being verified will be sent as PullMessages response during this test case. In such case, ONVIF Client should simply discard such response and they retry PullMessages request for the very event for verification.

During test case execution, it should be guaranteed that the DUT should not delete the property event

which is being used for verification.

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.

See Annex A.9 for instructions on how to construct Subscribe when it is required for the ONVIF Client to receive all events supported by the DUT.

7.3.4 REALTIME PULLPOINT SUBSCRIPTION - PULLMESSAGES FILTER

Test Label: event handling REALTIME PULL POINT INTERFACE PullMessages Filter

Test Case ID: EVENT-3-1-16

ONVIF Core Specification Coverage: CreatePullPointSubscription, SetSynchronizationPoint, PullMessages, MessageFilter

Command Under Test: GetEventProperties, CreatePullPointSubscription, SetSynchronizationPoint, PullMessages

WSDL Reference: event.wsdl

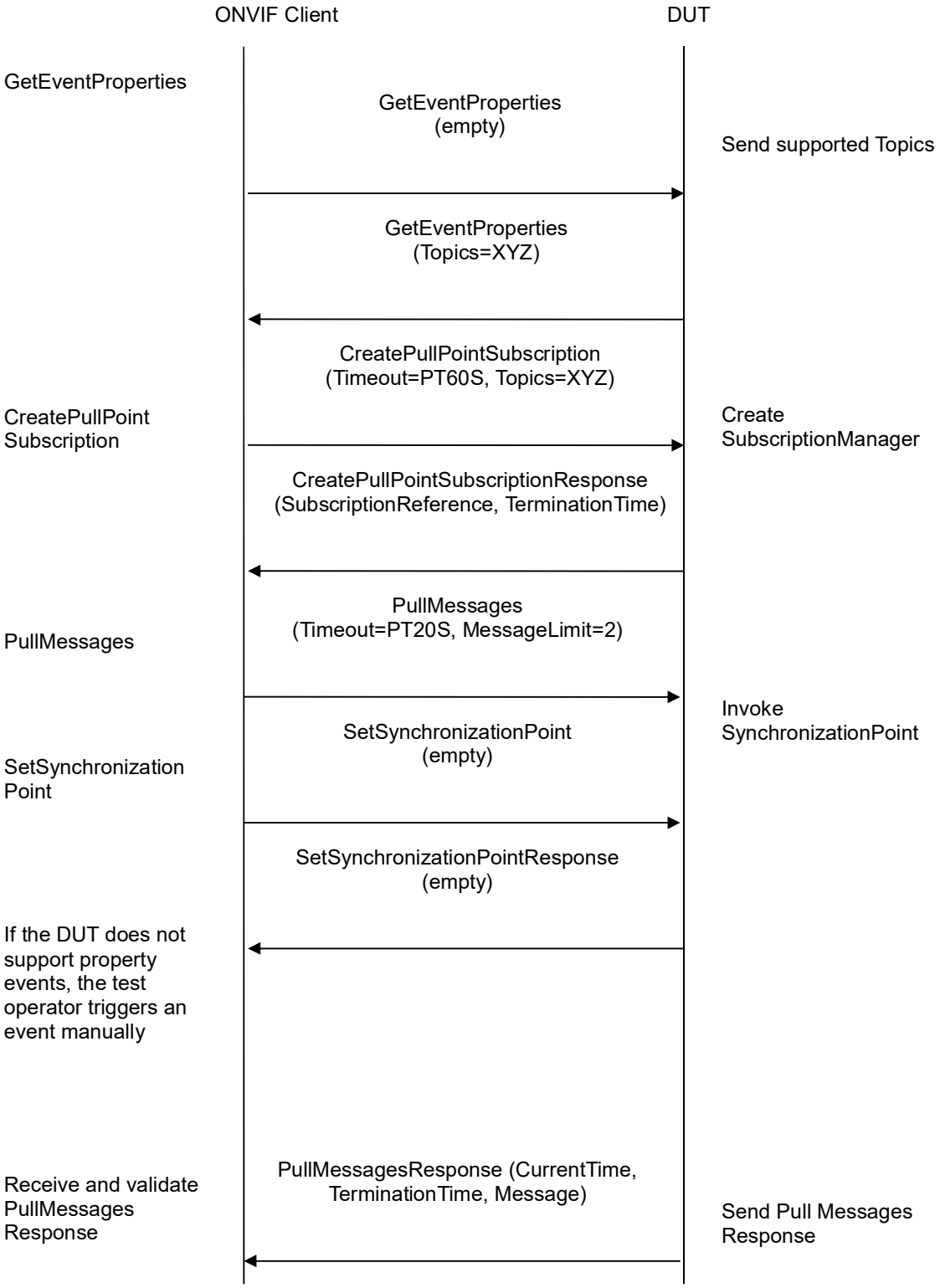
Test Purpose: To verify PullMessages command

Pre-Requisite: The DUT shall provide at least one event.

The test operator has to ensure that the event is triggered and sent out. ONVIF Client will invoke a SetSynchronizationPoint request. If the DUT does not support property events of if it is not possible to invoke a SynchronizationPoint, the test operator has to trigger an event manually.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
GetEventProperties
GetEventProperties
(empty)
Send supported Topics
GetEventProperties
(Topics=XYZ)
CreatePullPointSubscription
(Timeout=PT60S, Topics=XYZ)
CreatePullPoint
Subscription
Create
SubscriptionManager
CreatePullPointSubscriptionResponse
(SubscriptionReference, TerminationTime)
PullMessages
(Timeout=PT20S, MessageLimit=2)
PullMessages
Invoke
SynchronizationPoint
SetSynchronizationPoint
(empty)
SetSynchronization
Point
SetSynchronizationPointResponse
(empty)
If the DUT does not support property events, the test operator triggers an event manually
PullMessagesResponse (CurrentTime, TerminationTime, Message)
Receive and validate PullMessages Response
Send Pull Messages Response

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetEventProperties command to retrieve the supported Topics.
4. Verify that the DUT sends a GetEventPropertiesResponse; select one Topic
5. ONVIF Client will invoke CreatePullPointSubscription message with a suggested timeout of PT60S and a Filter including the selected Topic.
6. Verify that the DUT sends a CreatePullPointSubscriptionResponse.
7. Validate CurrentTime and TerminationTime and SubscriptionReference.
8. ONVIF Client will invoke PullMessages command with a PullMessagesTimeout of 20s and a MessageLimit of 2.
9. ONVIF Client will invoke SetSynchronizationPoint command to trigger a property event.
10. Verify that the DUT sends a SetSynchronizationPointResponse.
11. If the DUT does not support property events, the operator has to trigger an event manually.
12. Verify that the DUT sends a PullMessagesResponse that contains at least one NotificationMessage.
13. Verify that that a maximum number of 2 Notification Messages is included in the PullMessages Response.
14. Verify that at least one property event is returned.
15. Verify that this NotificationMessage is well formed; Verify CurrentTime and TerminationTime (TerminationTime>CurrentTime).
16. Verify that the Topic of the NotificationMessage matches the filter.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send a GetEventPropertiesResponse

The DUT did not send a valid GetEventPropertiesResponse (containing at least one Topic).

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).

The DUT did not send a SetSynchronizationPointResponse.

The DUT did not send a PullMessagesResponse.

The PullMessagesResponse does not contain a NotificationMessage.

The PullMessagesResponse contains more than 2 NotificationMessages.

The PullMessagesResponse does not contain at least one event.

The NotificationMessages are not well formed.

The NotificationMessage contains to a topic that was not requested.

The PullMessagesResponse contains invalid values for Current or TerminationTime.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for GetEventPropertiesResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for CreatePullPointSubscriptionResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for SetSynchronizationPointResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for PullMessagesResponse message (see Annex A.17).

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

It may be possible that some other events than the event which is being verified will be sent as PullMessages response during this test case. In such case, ONVIF Client should simply discard such response and they retry PullMessages request for the very event for verification.

During test case execution, it should be guaranteed that the DUT should not delete the property event which is being used for verification.

If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit included in PullMessagesFaultResponse.

7.3.5 REALTIME PULLPOINT SUBSCRIPTION - INVALID MESSAGE CONTENT FILTER

Test Label: event handling CREATE PULL POINT SUBSCRIPTION – INVALID MESSAGE CONTENT FILTER

Test Case ID: EVENT-3-1-17

ONVIF Core Specification Coverage: CreatePullPointSubscription

Command Under Test: CreatePullPointSubscription

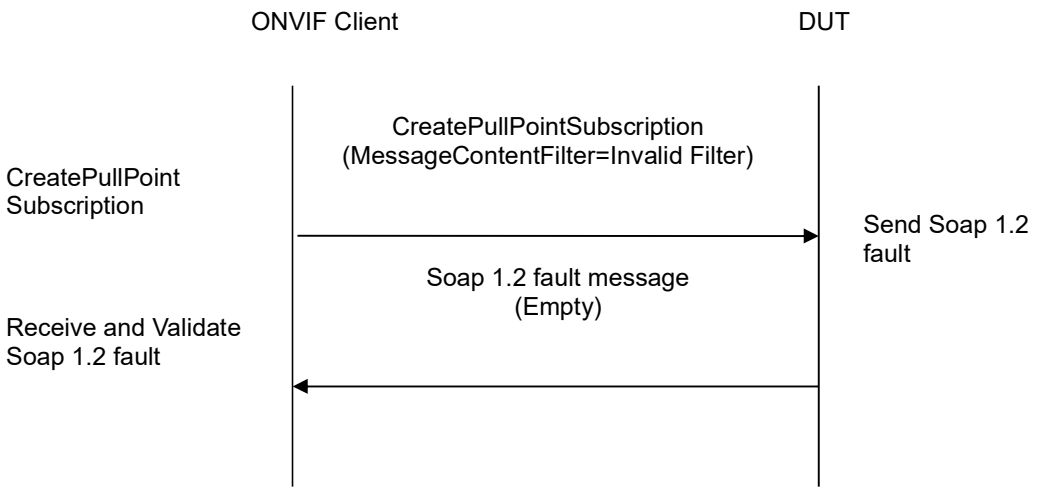
WSDL Reference: event.wsdl

Test Purpose: To verify that a correct error message "InvalidFilterFault" or “InvalidMessageContentExpressionFault” is returned if a CreatePullPointSubscription command with an invalid MessageContentFilter is invoked.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke CreatePullPointSubscription message with an invalid Filter boolean(//tt:SimpleItem[@Name=”xyz” and @Value=”xyz”]).
4. Verify that the DUT generates an “InvalidFilterFault” or an “InvalidMessageContentExpressionFault” fault message. Validate that Utc time and description are correct.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send an “InvalidFilterFault” or “InvalidMessageContentExpressionFault” fault message.

The DUT did not send a valid fault message.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for Fault message (see Annex A.17).

7.3.6 REALTIME PULLPOINT SUBSCRIPTION - UNSUBSCRIBE

Test Label: Event handling RealTime PullPoint UNSUBSCRIBE

Test Case ID: EVENT-3-1-19

Driving IP-based physical security through global standardization

ONVIF Core Specification Coverage: Basic Notification Interface, CreatePullPointSubscription

Command Under Test: CreatePullPointSubscription, Unsubscribe, Renew

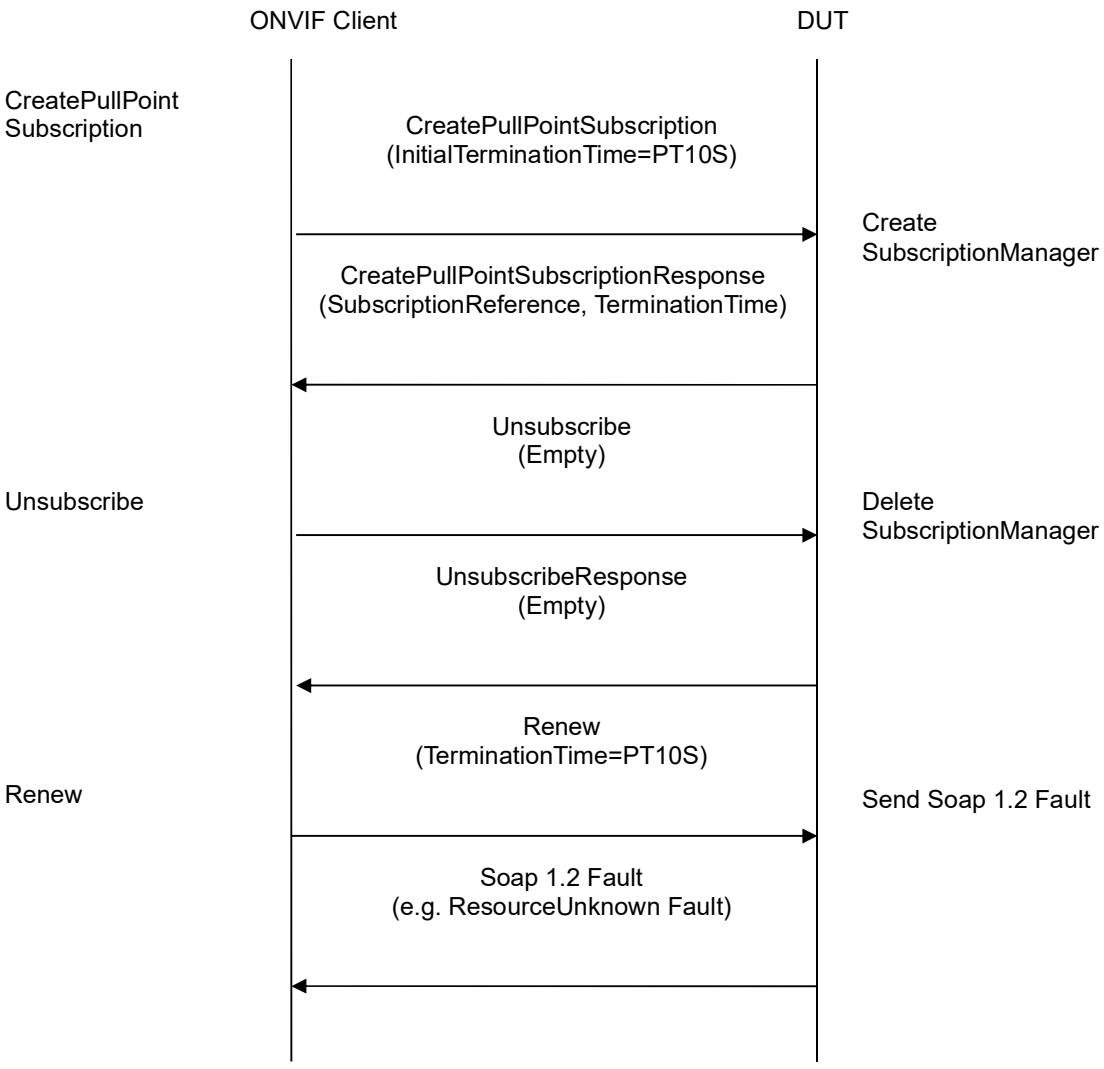
WSDL Reference: event.wsdl

Test Purpose: To verify Unsubscribe command

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client will invoke CreatePullPointSubscription message (InitialTerminationTime=PT10S) to instantiate a SubscriptionManager.
4. Verify that the DUT sends a CreatePullPointSubscriptionResponse. Validate CurrentTime and TerminationTime.
5. ONVIF Client will invoke Unsubscribe command to terminate the SubscriptionManager.
6. Verify that the DUT sends an UnsubscribeResponse.
7. ONVIF Client will invoke a Renew command to verify that the SubscriptionManager is deleted.
8. Verify that the DUT sends a Soap 1.2 Fault (e.g. a “ResourceUnknown” fault message).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime=CurrentTime+10s).

The DUT did not send an UnsubscribeResponse.

The DUT did not send a Soap 1.2.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for CreatePullPointSubscriptionResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for UnsubscribeResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for Fault message (see Annex A.17).

Note: If DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the CreatePullPointSubscription request with MinimumTime value included in UnacceptableInitialTerminationTime fault.

7.3.7 REALTIME PULLPOINT SUBSCRIPTION - TIMEOUT

Test Label: Event handling Realtime Pull Point Timeout

Test Case ID: EVENT-3-1-20

ONVIF Core Specification Coverage: CreatePullPointSubscription, Basic Notification Interface

Command Under Test: CreatePullPointSubscription

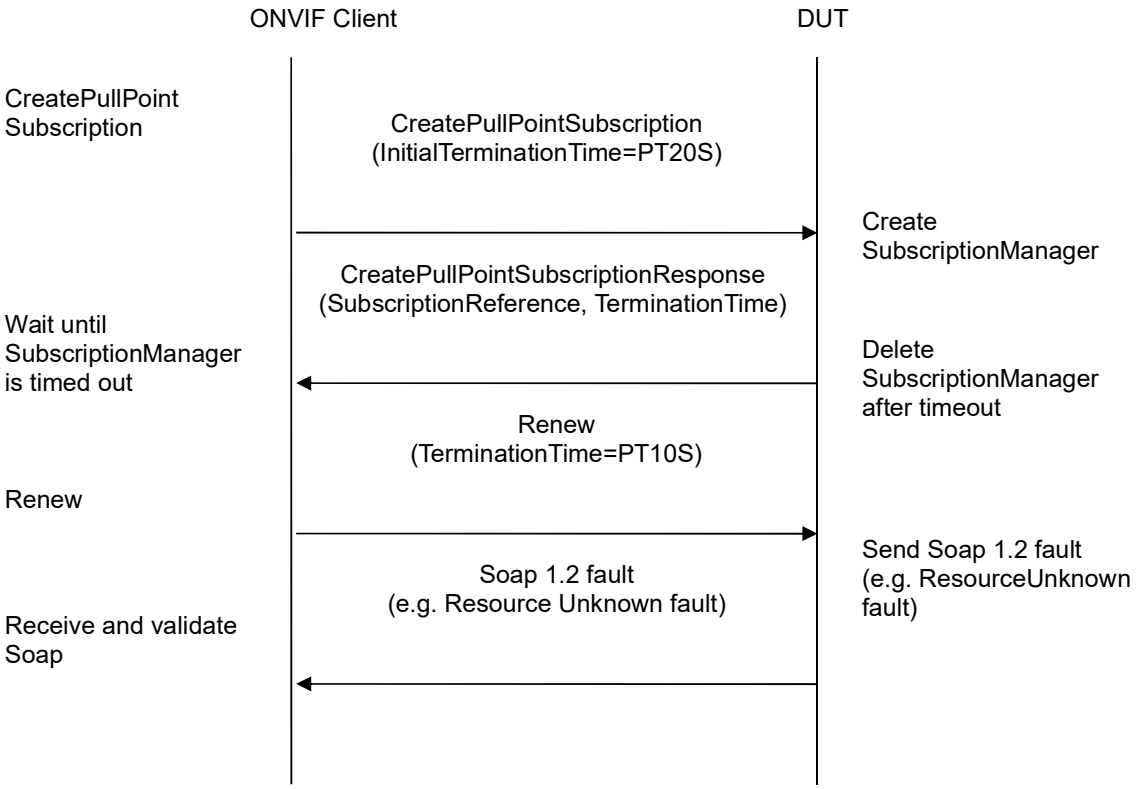
WSDL Reference: event.wsdl

Test Purpose: To verify if a SubscriptionManager times out correctly.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke CreatePullPointSubscription message with a suggested timeout of PT20S.
4. Verify that the DUT sends a CreatePullPointSubscriptionResponse.
5. Validate CurrentTime and TerminationTime and SubscriptionReference.
6. Wait for 20 s (SubscriptionManager timeout).
7. ONVIF Client will invoke Renew command to check if the SubscriptionManager is timed out.
8. Verify that the DUT sends a Soap 1.2 fault (e.g. a “ResourceUnknown” fault).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime >= CurrentTime+10s).

The DUT did not send a Soap 1.2 fault.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for CreatePullPointSubscriptionResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for Fault message (see Annex A.17).

Note: If DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the CreatePullPointSubscription request with MinimumTime value included in UnacceptableInitialTerminationTime fault.

7.3.8 REALTIME PULLPOINT SUBSCRIPTION - INVALID TOPIC EXPRESSION

Test Label: Event handling REALTIME PULLPOINT INVALID FILTER-TOPIC EXPRESSION

Test Case ID: EVENT-3-1-22

ONVIF Core Specification Coverage: CreatePullPointSubscription

Command Under Test: CreatePullPointSubscription

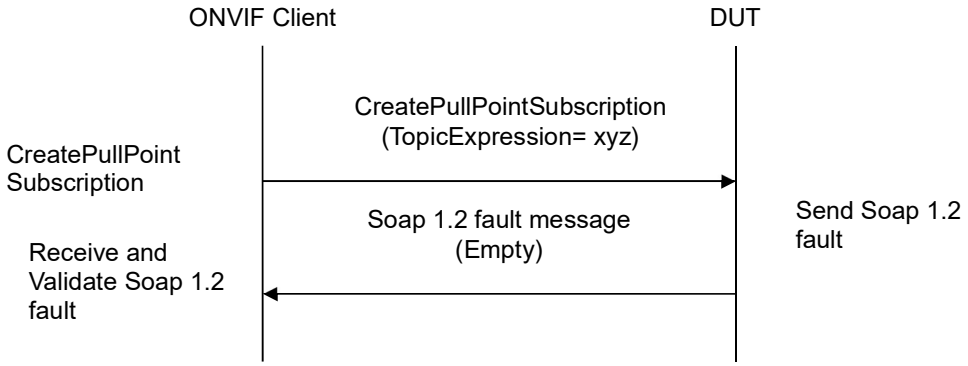
WSDL Reference: event.wsdl

Test Purpose: To verify that a correct error message "InvalidFilterFault" or “TopicNotSupported” or “InvalidTopicExpressionFault” is returned if a CreatePullPointSubscription command with an invalid TopicExpression is invoked.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke CreatePullPointSubscription message with an invalid Topic Expression “xyz”.
4. Verify that the DUT generates an “InvalidFilterFault” or “TopicNotSupported” or “InvalidTopicExpressionFault” fault message.
5. Validate valid the fault message (valid Utc time, valid description).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send an “InvalidFilterFault” or “TopicNotSupported” or “InvalidTopicExpressionFault” fault message.

The DUT did not send a valid fault message.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for Fault message (see Annex A.17).

7.3.9 REALTIME PULLPOINT SUBSCRIPTION – PULLMESSAGES AS KEEP-ALIVE

Test Label: Event Handling Realtime PullPoint Notification Interface PullMessages as Keep-Alive

Test Case ID: DRAFT-EVENT-3-1-24

ONVIF Core Specification Coverage: Realtime PullPoint Notification Interface, CreatePullPointSubscription, PullMessages

Command Under Test: CreatePullPointSubscription, PullMessages

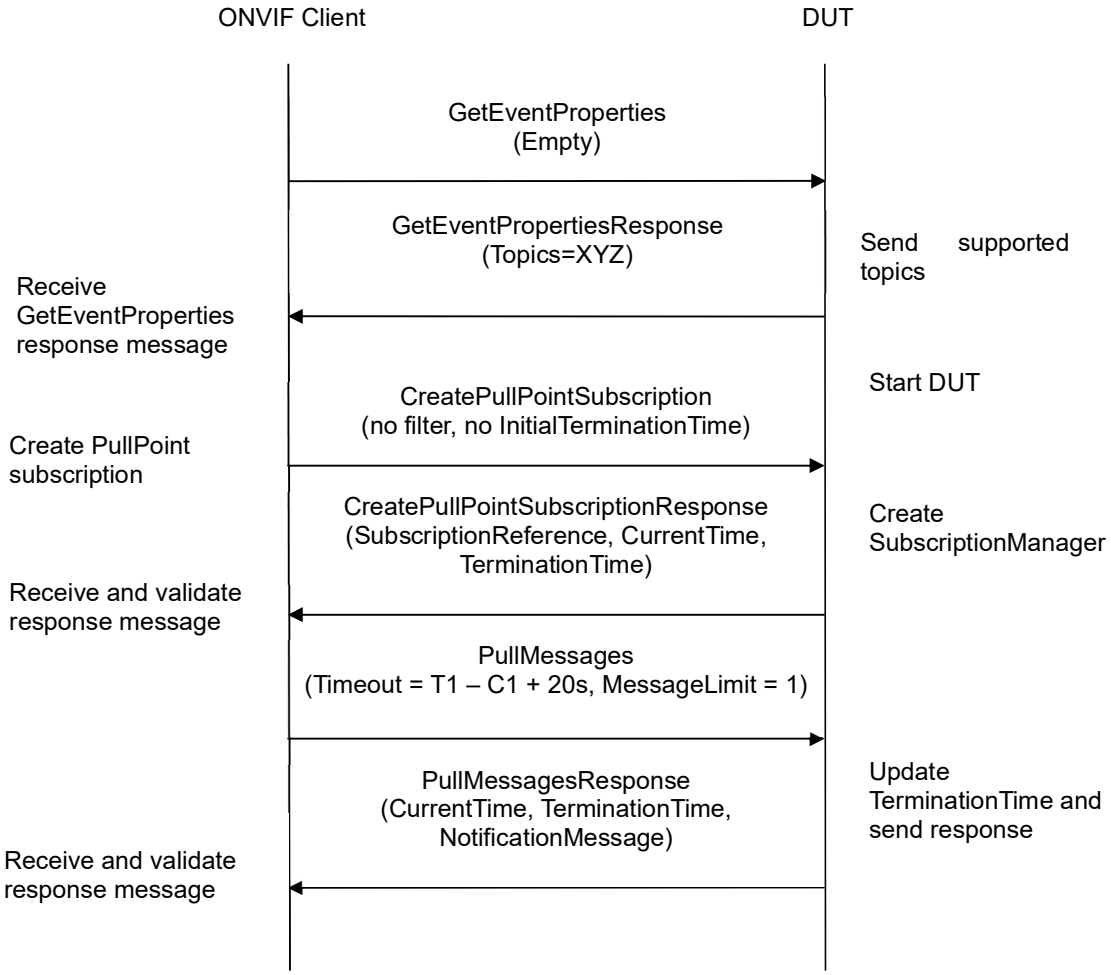
WSDL Reference: event.wsdl

Test Purpose: To verify PullMessages command as the one being kept alive.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties** command to retrieve the supported Topics.
4. Verify that the DUT sends a GetEventPropertiesResponse.
5. ONVIF Client will invoke **CreatePullPointSubscription** message (no filter, no InitialTerminationTime).
6. Verify that the DUT sends a **CreatePullPointSubscriptionResponse**. Validate CurrentTime and TerminationTime. Mark Termination Time as T1. Mark Current Time as C1.
7. ONVIF Client waits for 1 s.
8. ONVIF Client will invoke **PullMessagesRequest** message with a PullMessages Timeout of [T1 – C1 + 20s] and a MessageLimit of 1 to update termination time of subscription.
9. Verify **PullMessagesResponse** message or **PullMessagesFaultResponse** from the DUT.
10. If PullMessagesFaultResponse is received from the DUT then ONVIF Client will invoke PullMessagesRequest message with a PullMessages Timeout equal to MaxTimeout from PullMessagesFaultResponse and a MessageLimit of 1 to update termination time of subscription. Otherwise, go to step 11.
11. Verify **PullMessagesResponse** message from the DUT.
12. Validate Current Time and Termination Time. Check that TerminationTime > CurrentTime. Check that TerminationTime > T1.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not send valid values for CurrentTime and TerminationTime.

The DUT did not send a PullMessagesResponse or PullMessagesFaultResponse at step 6.

The DUT did not send a **PullMessagesResponse** at step 8.

The **PullMessagesResponse** contained invalid values for Current or TerminationTime.

The **PullMessagesResponse** to the **PullMessagesRequest** message with a PullMessages Timeout of [T1 – C1 + 20s] contained TerminationTime value less or equal to T1.

Note: If the DUT does not update termination time after PullMessagesRequest, then the ONVIF Client passes the test with a warning.

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

Note: If the DUT does not support property events, then test operator will send event manually (user interaction window will be used).

Note: To avoid long time of test execution, the test tool waits PullMessagesResponse during Operation delay timeout time. If the DUT does not send PullMessagesResponse during Operation delay timeout time, the test tool terminates subscription and fails the test.

7.3.10 REALTIME PULLPOINT SUBSCRIPTION - SET SYNCHRONIZATION POINT

Test Label: Event Handling REALTIME PULLPOINT SUBSCRIPTION SET SYNCHRONIZATION POINT

Test Case ID: EVENT-3-1-25

ONVIF Core Specification Coverage: CreatePullPointSubscription, SetSynchronizationPoint, PullMessages

Command Under Test: GetEventProperties, CreatePullPointSubscription, SetSynchronizationPoint, PullMessages

WSDL Reference: event.wsdl

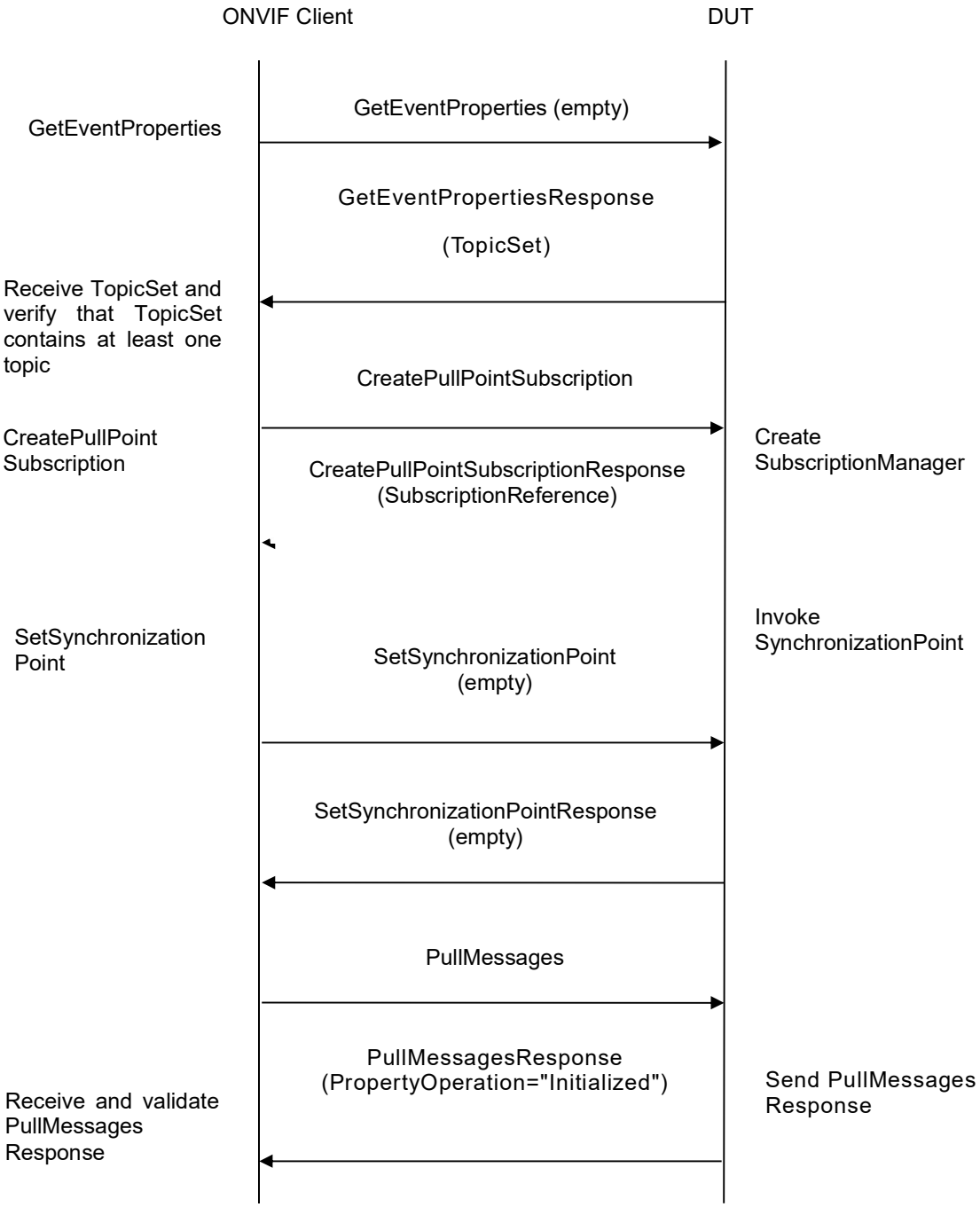
Test Purpose: To verify that the DUT sends all currently alive properties to the Client when SetSynchronizationPoint operation is invoked.

Pre-Requisite: The DUT shall provide at least one property event.

The test operator has to ensure that the event is triggered and sent out. ONVIF Client will invoke a SetSynchronizationPoint request. The DUT shall produce at least one property pull point notification with the attribute "PropertyOperation" set to the "Initialized" state.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client will invoke GetEventProperties command.
4. Verify that the DUT sends GetEventPropertiesResponse with TopicSet and TopicSet contains at least one topic. If TopicSet is empty then skip other steps.
5. ONVIF Client will invoke CreatePullPointSubscription message.
6. Verify that the DUT sends a CreatePullPointSubscriptionResponse.
7. ONVIF Client will invoke SetSynchronizationPoint command to trigger an event.
8. Verify that the DUT sends a SetSynchronizationPointResponse.
9. ONVIF Client will invoke PullMessages command.
10. Verify that the DUT sends a PullMessagesResponse that contains at least one NotificationMessage that represents a property.
11. Verify NotificationMessage to ensure that this is correct property notification message with attribute PropertyOperation="Initialized"

Test Result:

PASS –

The DUT passed all assertions.

The DUT returned GetEventPropertiesResponse with empty TopicSet.

FAIL –

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not send a SetSynchronizationPointResponse.

The DUT did not send a PullMessagesResponse.

The PullMessagesResponse does not contain a NotificationMessage.

The NotificationMessages are not well formed.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for CreatePullPointSubscriptionResponse message (see AnnexA.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for SetSynchronizationPointResponse message (see AnnexA.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for PullMessagesResponse message (see AnnexA.17).

7.3.11 REALTIME PULLPOINT SUBSCRIPTION – RELAY EVENT

Test Label: event handling REALTIME PULL POINT INTERFACE - RELAY EVENT

Test Case ID: EVENT-3-1-27

ONVIF Core Specification Coverage: CreatePullPointSubscription, PullMessages, MessageFilter

Command Under Test: GetEventProperties, CreatePullPointSubscription, PullMessages

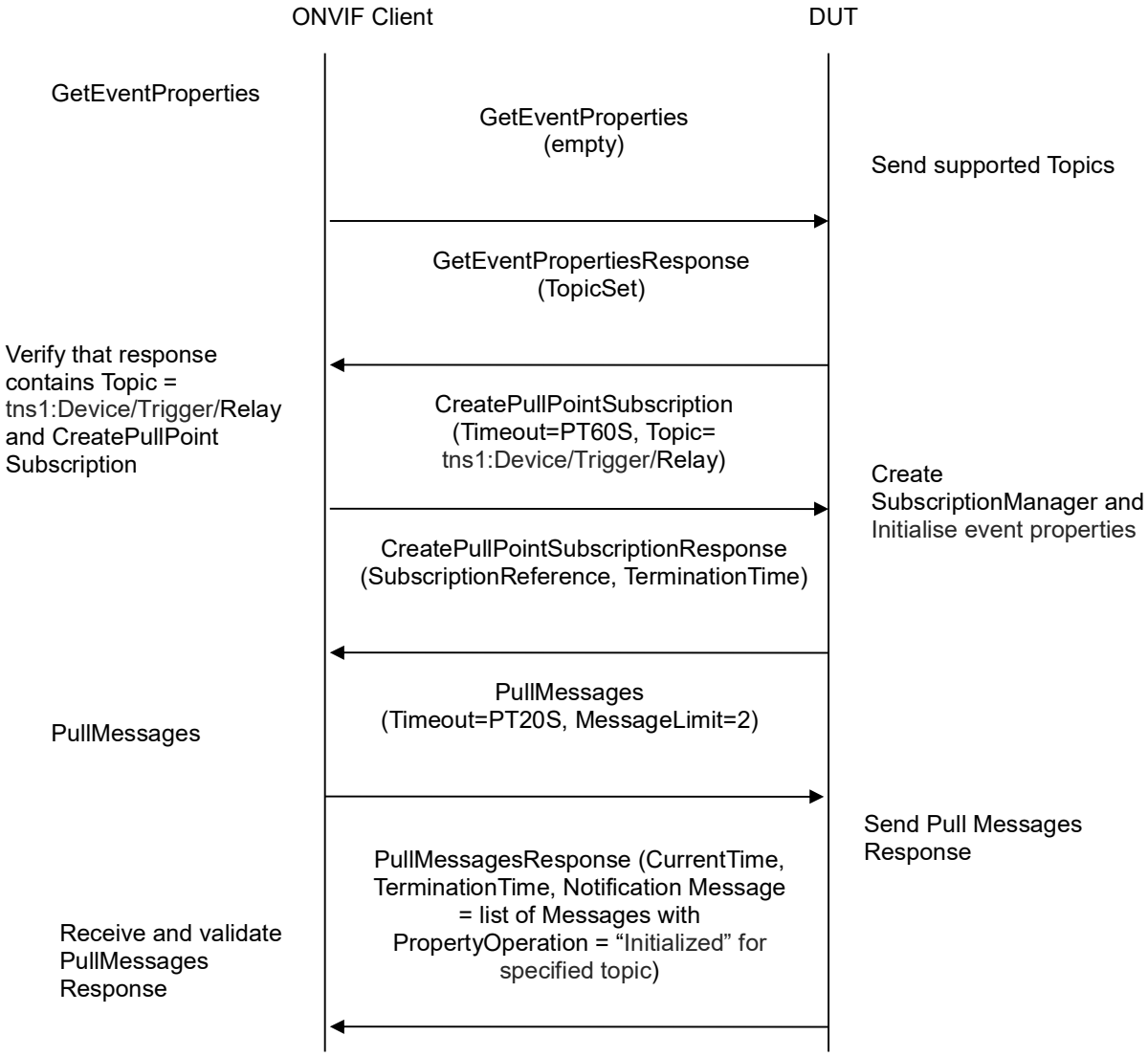
WSDL Reference: event.wsdl

Test Purpose: To verify that the device sends Notification messages for the Topic="tns1:Device/Trigger/Relay".

Pre-Requisite: Device supports Relay Outputs feature.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
GetEventProperties
GetEventProperties
(empty)
Send supported Topics
GetEventPropertiesResponse
(TopicSet)
Verify that response contains Topic = tns1:Device/Trigger/Relay and CreatePullPoint Subscription
CreatePullPointSubscription
(Timeout=PT60S, Topic= tns1:Device/Trigger/Relay)
Create SubscriptionManager and Initialise event properties
CreatePullPointSubscriptionResponse
(SubscriptionReference, TerminationTime)
PullMessages
(Timeout=PT20S, MessageLimit=2)
PullMessages
Send Pull Messages Response
PullMessagesResponse (CurrentTime, TerminationTime, Notification Message = list of Messages with PropertyOperation = “Initialized” for specified topic)
Receive and validate PullMessages Response

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. See the definition below (Note: LogicalState can be either set at "true" or "false"):

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="RelayToken" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="LogicalState" Type="tt:RelayLogicalState"/>
  </tt:Data>
</tt:MessageDescription>
```

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. Validate CurrentTime and TerminationTime and SubscriptionReference.
8. ONVIF Client will invoke PullMessages command with a PullMessagesTimeout of 20s and a MessageLimit of 2.
9. 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.
10. Verify that this NotificationMessage is well formed; Verify CurrentTime and TerminationTime (TerminationTime>CurrentTime) and PropertyOperation (PropertyOperation = "Initialized").

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send a GetEventPropertiesResponse

The GetEventPropertiesResponse does not contain Topic="tns1:Device/Trigger/Relay" or MessageDescription is wrong or the topic does not contain MessageDescription.

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).

The DUT did not send a PullMessagesResponse.

The PullMessagesResponse does not contain a NotificationMessage with Topic="tns1:Device/Trigger/Relay" or the PropertyOperation of event is not "Initialized".

The PullMessagesResponse contains more than 2 NotificationMessages.

The NotificationMessages are not well formed or the structure of NotificationMessage does not meet Property NotificationMessage standard described in "Property example, continued" section of ONVIF-Core-Spec document.

The NotificationMessage contains to a topic that was not requested.

The PullMessagesResponse contains invalid values for Current or TerminationTime.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for GetEventPropertiesResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for CreatePullPointSubscriptionResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for PullMessagesResponse message (see Annex A.17).

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

If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit included in PullMessagesFaultResponse.

7.3.12 REALTIME PULLPOINT SUBSCRIPTION – IMAGE TOO BLURRY

Test Label: event handling REALTIME PULL POINT INTERFACE – IMAGE TOO BLURRY

Test Case ID: EVENT-3-1-28

ONVIF Core Specification Coverage: CreatePullPointSubscription, PullMessages, MessageFilter

Command Under Test: GetEventProperties, CreatePullPointSubscription, PullMessages

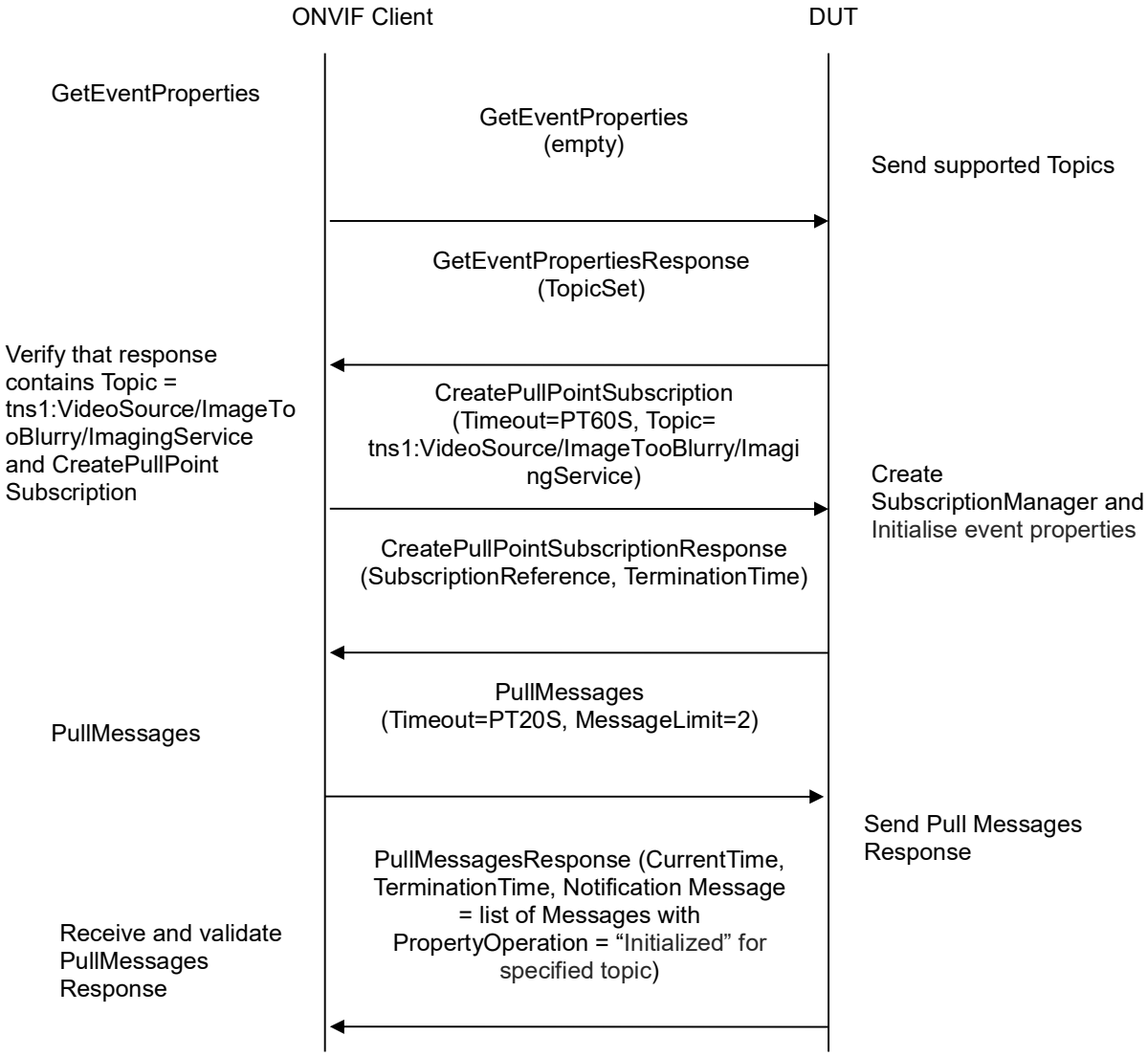
WSDL Reference: event.wsdl

Test Purpose: To verify that the device sends Notification messages for the Topic="tns1:VideoSource/ImageTooBlurry/ImagingService".

Pre-Requisite: Device supports Imaging Service.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
GetEventProperties
GetEventProperties
(empty)
Send supported Topics
GetEventPropertiesResponse
(TopicSet)
Verify that response contains Topic = tns1:VideoSource/ImageTooBlurry/ImagingService and CreatePullPoint Subscription
CreatePullPointSubscription
(Timeout=PT60S, Topic= tns1:VideoSource/ImageTooBlurry/ImagingService)
Create SubscriptionManager and Initialise event properties
CreatePullPointSubscriptionResponse
(SubscriptionReference, TerminationTime)
PullMessages
(Timeout=PT20S, MessageLimit=2)
PullMessages
Send Pull Messages Response
PullMessagesResponse (CurrentTime, TerminationTime, Notification Message = list of Messages with PropertyOperation = “Initialized” for specified topic)
Receive and validate PullMessages Response

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:VideoSource/ImageTooBlurry/ImagingService" and this topic contains MessageDescription item. This MessageDescription is defined in "Event" section of ONVIF-Imaging-Service-Spec document. See the definition below (Note: State can be either set at "true" or "false"):

```
<tt:MessageDescription IsProperty="true">
    <tt:Source>
        <tt:SimpleItemDescription Name="Source" Type="tt:ReferenceToken"/>
    </tt:Source>
    <tt:Data>
        <tt:SimpleItemDescription Name="State" Type="xs:boolean"/>
    </tt:Data>
</tt:MessageDescription>
```

5. ONVIF Client will invoke CreatePullPointSubscription message with a suggested timeout of PT60S and a Filter including the Topic=" tns1:VideoSource/ImageTooBlurry/ImagingService"
6. Verify that the DUT sends a CreatePullPointSubscriptionResponse.
7. Validate CurrentTime and TerminationTime and SubscriptionReference.
8. ONVIF Client will invoke PullMessages command with a PullMessagesTimeout of 20s and a MessageLimit of 2.
9. Verify that the DUT sends a PullMessagesResponse that contains at least one NotificationMessage with Topic="tns1:VideoSource/ImageTooBlurry/ImagingService" and this message contains a property event.
10. Verify that this NotificationMessage is well formed; Verify CurrentTime and TerminationTime (TerminationTime>CurrentTime) and PropertyOperation (PropertyOperation = "Initialized").

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send a GetEventPropertiesResponse

The GetEventPropertiesResponse does not contain Topic="tns1:VideoSource/ImageTooBlurry/ImagingService" or MessageDescription is wrong or the topic does not contain MessageDescription.

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).

The DUT did not send a PullMessagesResponse.

The PullMessagesResponse does not contain a NotificationMessage with Topic="tns1:VideoSource/ImageTooBlurry/ImagingService" or the PropertyOperation of event is not "Initialized".

The PullMessagesResponse contains more than 2 NotificationMessages.

The NotificationMessages are not well formed or the structure of NotificationMessage does not meet Property NotificationMessage standard described in "Property example, continued" section of ONVIF-Core-Spec document.

The NotificationMessage contains to a topic that was not requested.

The PullMessagesResponse contains invalid values for Current or TerminationTime.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for GetEventPropertiesResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for CreatePullPointSubscriptionResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for PullMessagesResponse message (see Annex A.17).

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

If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit included in PullMessagesFaultResponse.

7.3.13 REALTIME PULLPOINT SUBSCRIPTION – IMAGE TOO DARK

Test Label: event handling REALTIME PULL POINT INTERFACE – IMAGE TOO DARK

Test Case ID: EVENT-3-1-29

ONVIF Core Specification Coverage: CreatePullPointSubscription, PullMessages, MessageFilter

Command Under Test: GetEventProperties, CreatePullPointSubscription, PullMessages

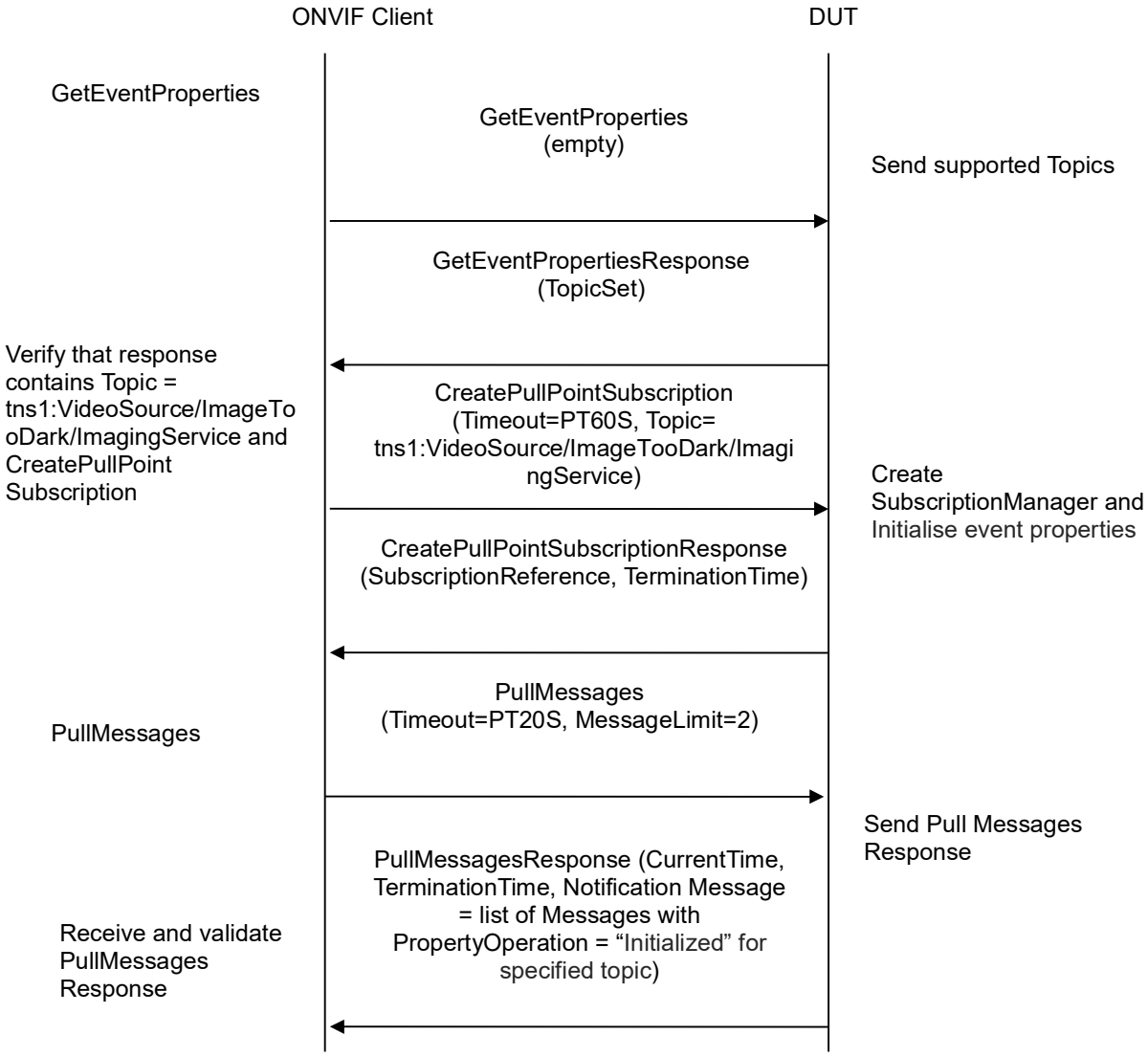
WSDL Reference: event.wsdl

Test Purpose: To verify that the device sends Notification messages for the Topic="tns1:VideoSource/ImageTooDark/ImagingService".

Pre-Requisite: Device supports Imaging Service.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
GetEventProperties
GetEventProperties
(empty)
Send supported Topics
GetEventPropertiesResponse
(TopicSet)
Verify that response contains Topic = tns1:VideoSource/ImageTooDark/ImagingService and CreatePullPoint Subscription
CreatePullPointSubscription
(Timeout=PT60S, Topic= tns1:VideoSource/ImageTooDark/ImagingService)
Create SubscriptionManager and Initialise event properties
CreatePullPointSubscriptionResponse
(SubscriptionReference, TerminationTime)
PullMessages
(Timeout=PT20S, MessageLimit=2)
PullMessages
Send Pull Messages Response
PullMessagesResponse (CurrentTime, TerminationTime, Notification Message = list of Messages with PropertyOperation = “Initialized” for specified topic)
Receive and validate PullMessages Response

Test Procedure:

11. Start an ONVIF Client.
12. Start the DUT.
13. ONVIF Client will invoke GetEventProperties command.
14. Verify that the DUT sends a GetEventPropertiesResponse, and response contains Topic="tns1:VideoSource/ImageTooDark/ImagingService" and this topic contains MessageDescription item. This MessageDescription is defined in "Event" section of ONVIF-Imaging-Service-Spec document. See the definition below (Note: State can be either set at "true" or "false"):

```
<tt:MessageDescription IsProperty="true">
    <tt:Source>
        <tt:SimpleItemDescription Name="Source" Type="tt:ReferenceToken"/>
    </tt:Source>
    <tt:Data>
        <tt:SimpleItemDescription Name="State" Type="xs:boolean"/>
    </tt:Data>
</tt:MessageDescription>
```

15. ONVIF Client will invoke CreatePullPointSubscription message with a suggested timeout of PT60S and a Filter including the Topic=" tns1:VideoSource/ImageTooDark/ImagingService"
16. Verify that the DUT sends a CreatePullPointSubscriptionResponse.
17. Validate CurrentTime and TerminationTime and SubscriptionReference.
18. ONVIF Client will invoke PullMessages command with a PullMessagesTimeout of 20s and a MessageLimit of 2.
19. Verify that the DUT sends a PullMessagesResponse that contains at least one NotificationMessage with Topic="tns1:VideoSource/ImageTooDark/ImagingService" and this message contains a property event.
20. Verify that this NotificationMessage is well formed; Verify CurrentTime and TerminationTime (TerminationTime>CurrentTime) and PropertyOperation (PropertyOperation = "Initialized").

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send a GetEventPropertiesResponse

The GetEventPropertiesResponse does not contain Topic="tns1:VideoSource/ImageTooDark/ImagingService" or MessageDescription is wrong or the topic does not contain MessageDescription.

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).

The DUT did not send a PullMessagesResponse.

The PullMessagesResponse does not contain a NotificationMessage with Topic="tns1:VideoSource/ImageTooDark/ImagingService" or the PropertyOperation of event is not "Initialized".

The PullMessagesResponse contains more than 2 NotificationMessages.

The NotificationMessages are not well formed or the structure of NotificationMessage does not meet Property NotificationMessage standard described in "Property example, continued" section of ONVIF-Core-Spec document.

The NotificationMessage contains to a topic that was not requested.

The PullMessagesResponse contains invalid values for Current or TerminationTime.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for GetEventPropertiesResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for CreatePullPointSubscriptionResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for PullMessagesResponse message (see Annex A.17).

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

If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit included in PullMessagesFaultResponse.

7.3.14 REALTIME PULLPOINT SUBSCRIPTION – IMAGE TOO BRIGHT

Test Label: event handling REALTIME PULL POINT INTERFACE – IMAGE TOO BRIGHT

Test Case ID: EVENT-3-1-30

ONVIF Core Specification Coverage: CreatePullPointSubscription, PullMessages, MessageFilter

Command Under Test: GetEventProperties, CreatePullPointSubscription, PullMessages

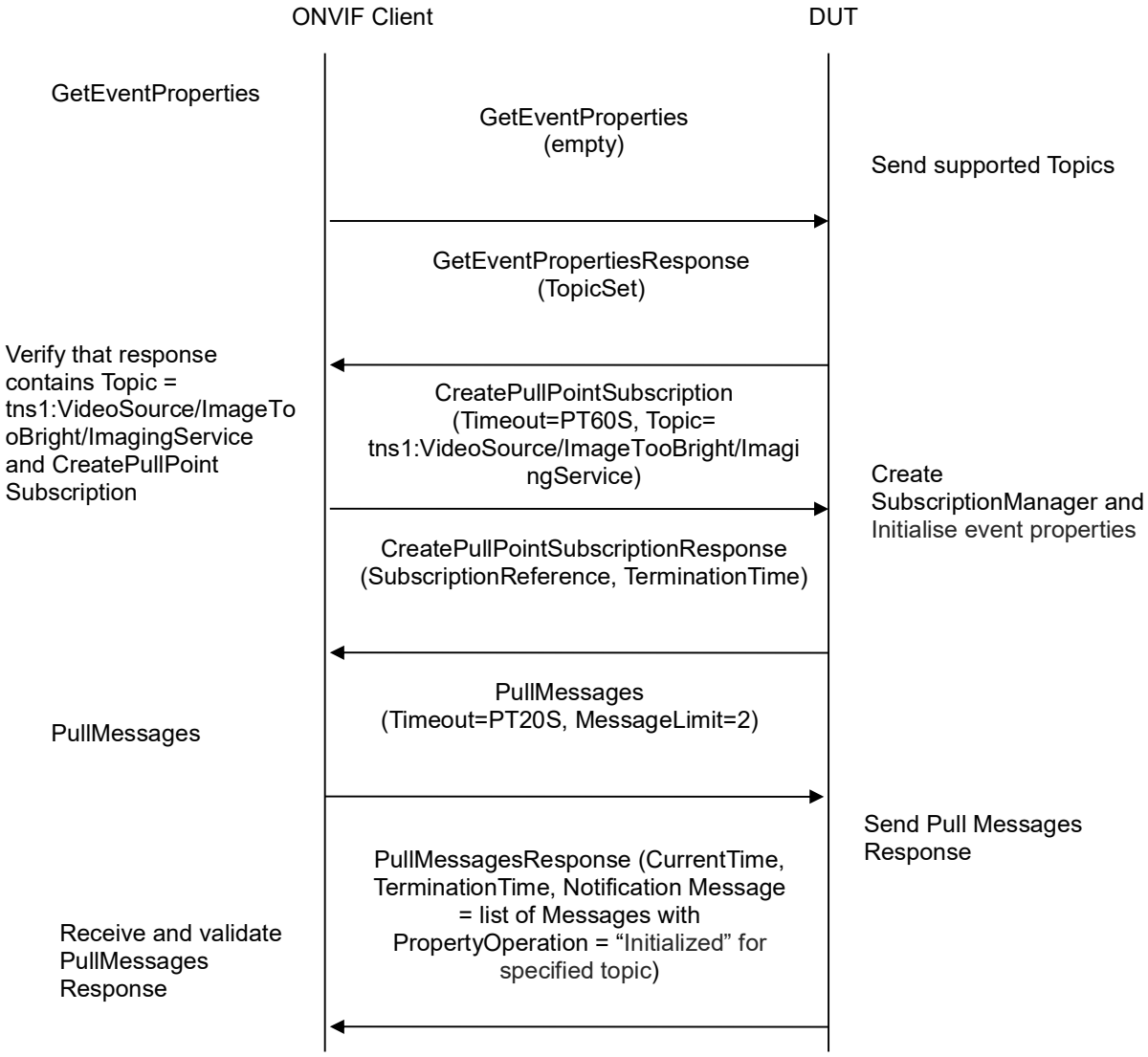
WSDL Reference: event.wsdl

Test Purpose: To verify that the device sends Notification messages for the Topic="tns1:VideoSource/ImageTooBright/ImagingService".

Pre-Requisite: Device supports Imaging Service.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
GetEventProperties
GetEventProperties
(empty)
Send supported Topics
GetEventPropertiesResponse
(TopicSet)
Verify that response contains Topic = tns1:VideoSource/ImageTooBright/ImagingService and CreatePullPoint Subscription
CreatePullPointSubscription
(Timeout=PT60S, Topic= tns1:VideoSource/ImageTooBright/ImagingService)
Create SubscriptionManager and Initialise event properties
CreatePullPointSubscriptionResponse
(SubscriptionReference, TerminationTime)
PullMessages
(Timeout=PT20S, MessageLimit=2)
PullMessages
Send Pull Messages Response
PullMessagesResponse (CurrentTime, TerminationTime, Notification Message = list of Messages with PropertyOperation = “Initialized” for specified topic)
Receive and validate PullMessages Response

Test Procedure:

21. Start an ONVIF Client.
22. Start the DUT.
23. ONVIF Client will invoke GetEventProperties command.
24. Verify that the DUT sends a GetEventPropertiesResponse, and response contains Topic="tns1:VideoSource/ImageTooBright/ImagingService" and this topic contains MessageDescription item. This MessageDescription is defined in "Event" section of ONVIF-Imaging-Service-Spec document. See the definition below (Note: State can be either set at "true" or "false"):

```
<tt:MessageDescription IsProperty="true">
    <tt:Source>
        <tt:SimpleItemDescription Name="Source" Type="tt:ReferenceToken"/>
    </tt:Source>
    <tt:Data>
        <tt:SimpleItemDescription Name="State" Type="xs:boolean"/>
    </tt:Data>
</tt:MessageDescription>
```

25. ONVIF Client will invoke CreatePullPointSubscription message with a suggested timeout of PT60S and a Filter including the Topic=" tns1:VideoSource/ImageTooBright/ImagingService"
26. Verify that the DUT sends a CreatePullPointSubscriptionResponse.
27. Validate CurrentTime and TerminationTime and SubscriptionReference.
28. ONVIF Client will invoke PullMessages command with a PullMessagesTimeout of 20s and a MessageLimit of 2.
29. Verify that the DUT sends a PullMessagesResponse that contains at least one NotificationMessage with Topic="tns1:VideoSource/ImageTooBright/ImagingService" and this message contains a property event.
30. Verify that this NotificationMessage is well formed; Verify CurrentTime and TerminationTime (TerminationTime>CurrentTime) and PropertyOperation (PropertyOperation = "Initialized").

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send a GetEventPropertiesResponse

The GetEventPropertiesResponse does not contain Topic="tns1:VideoSource/ImageTooBright/ImagingService" or MessageDescription is wrong or the topic does not contain MessageDescription.

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).

The DUT did not send a PullMessagesResponse.

The PullMessagesResponse does not contain a NotificationMessage with Topic="tns1:VideoSource/ImageTooBright/ImagingService" or the PropertyOperation of event is not "Initialized".

The PullMessagesResponse contains more than 2 NotificationMessages.

The NotificationMessages are not well formed or the structure of NotificationMessage does not meet Property NotificationMessage standard described in "Property example, continued" section of ONVIF-Core-Spec document.

The NotificationMessage contains to a topic that was not requested.

The PullMessagesResponse contains invalid values for Current or TerminationTime.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for GetEventPropertiesResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for CreatePullPointSubscriptionResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for PullMessagesResponse message (see Annex A.17).

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

If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit included in PullMessagesFaultResponse.

7.3.15 REALTIME PULLPOINT SUBSCRIPTION – GLOBAL SCENE CHANGE

Test Label: event handling REALTIME PULL POINT INTERFACE – GLOBAL SCENE CHANGE

Test Case ID: EVENT-3-1-31

ONVIF Core Specification Coverage: CreatePullPointSubscription, PullMessages, MessageFilter

Command Under Test: GetEventProperties, CreatePullPointSubscription, PullMessages

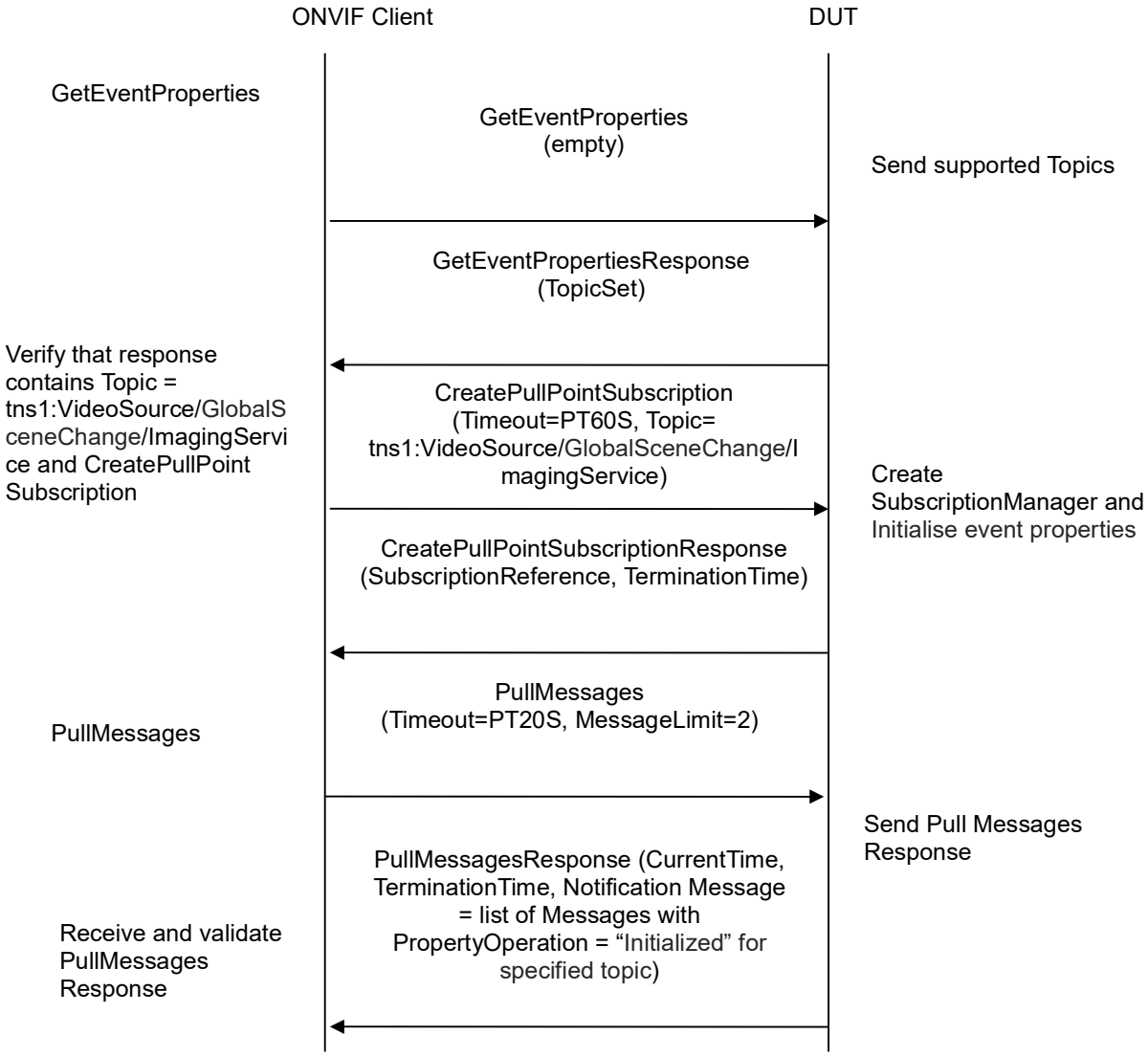
WSDL Reference: event.wsdl

Test Purpose: To verify that the device sends Notification messages for the Topic="tns1:VideoSource/GlobalSceneChange/ImagingService".

Pre-Requisite: Device supports Imaging Service.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
GetEventProperties
GetEventProperties
(empty)
Send supported Topics
GetEventPropertiesResponse
(TopicSet)
Verify that response contains Topic = tns1:VideoSource/GlobalSceneChange/ImagingService and CreatePullPoint Subscription
CreatePullPointSubscription
(Timeout=PT60S, Topic= tns1:VideoSource/GlobalSceneChange/ImagingService)
Create SubscriptionManager and Initialise event properties
CreatePullPointSubscriptionResponse
(SubscriptionReference, TerminationTime)
PullMessages
(Timeout=PT20S, MessageLimit=2)
PullMessages
Send Pull Messages Response
PullMessagesResponse (CurrentTime, TerminationTime, Notification Message = list of Messages with PropertyOperation = “Initialized” for specified topic)
Receive and validate PullMessages Response

Test Procedure:

31. Start an ONVIF Client.
32. Start the DUT.
33. ONVIF Client will invoke GetEventProperties command.
34. Verify that the DUT sends a GetEventPropertiesResponse, and response contains Topic="tns1:VideoSource/GlobalSceneChange/ImagingService" and this topic contains MessageDescription item. This MessageDescription is defined in "Event" section of ONVIF-Imaging-Service-Spec document. See the definition below (Note: State can be either set at "true" or "false"):

```
<tt:MessageDescription IsProperty="true">
    <tt:Source>
        <tt:SimpleItemDescription Name="Source" Type="tt:ReferenceToken"/>
    </tt:Source>
    <tt:Data>
        <tt:SimpleItemDescription Name="State" Type="xs:boolean"/>
    </tt:Data>
</tt:MessageDescription>
```

35. ONVIF Client will invoke CreatePullPointSubscription message with a suggested timeout of PT60S and a Filter including the Topic=" tns1:VideoSource/GlobalSceneChange/ImagingService"
36. Verify that the DUT sends a CreatePullPointSubscriptionResponse.
37. Validate CurrentTime and TerminationTime and SubscriptionReference.
38. ONVIF Client will invoke PullMessages command with a PullMessagesTimeout of 20s and a MessageLimit of 2.
39. Verify that the DUT sends a PullMessagesResponse that contains at least one NotificationMessage with Topic="tns1:VideoSource/GlobalSceneChange/ImagingService" and this message contains a property event.
40. Verify that this NotificationMessage is well formed; Verify CurrentTime and TerminationTime (TerminationTime>CurrentTime) and PropertyOperation (PropertyOperation = "Initialized").

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send a GetEventPropertiesResponse

The GetEventPropertiesResponse does not contain Topic="tns1:VideoSource/GlobalSceneChange/ImagingService" or MessageDescription is wrong or the topic does not contain MessageDescription.

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).

The DUT did not send a PullMessagesResponse.

The PullMessagesResponse does not contain a NotificationMessage with Topic="tns1:VideoSource/GlobalSceneChange/ImagingService" or the PropertyOperation of event is not "Initialized".

The PullMessagesResponse contains more than 2 NotificationMessages.

The NotificationMessages are not well formed or the structure of NotificationMessage does not meet Property NotificationMessage standard described in "Property example, continued" section of ONVIF-Core-Spec document.

The NotificationMessage contains to a topic that was not requested.

The PullMessagesResponse contains invalid values for Current or TerminationTime.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for GetEventPropertiesResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for CreatePullPointSubscriptionResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for PullMessagesResponse message (see Annex A.17).

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

If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit included in PullMessagesFaultResponse.

7.4 Namespace Handling

7.4.1 EVENT - NAMESPACES (DEFAULT NAMESPACES FOR EACH TAG)

Test Label: Event Service Different Namespaces Definition Test (Default Namespaces for Each Tag).

Test Case ID: EVENT-4-1-6

ONVIF Core Specification Coverage: None

Command Under Test: None

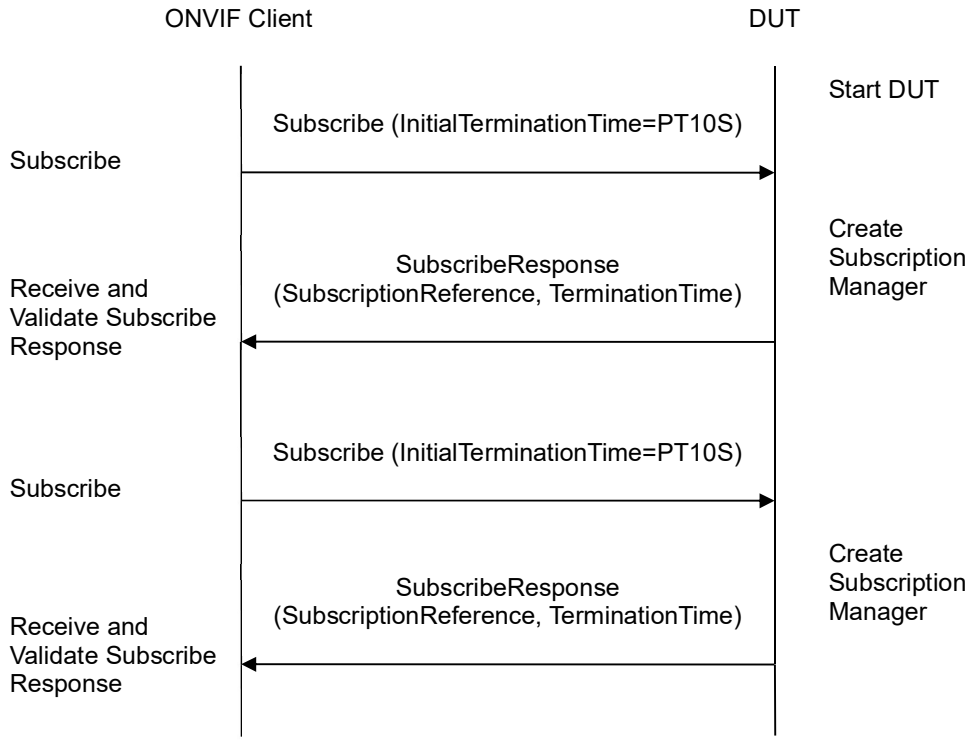
WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT accepts requests for Event Service with different namespaces definition.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

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

3. ONVIF Client will invoke Subscribe message to instantiate a Subscription Manager. The Subscribe message does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.
4. Verify that the device sends SubscribeResponse message. Validate that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).
5. ONVIF Client will invoke Subscribe message to instantiate a Subscription Manager. The Subscribe message does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.
6. Verify that the DUT sends SubscribeResponse message. Validate that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SubscribeResponse message.

The DUT did not return a valid SubscriptionReference.

The DUT did not return valid values for CurrentTime and TerminationTime.

The DUT returned different results at step 4 and step 6 (EndpointReference, TerminationTime, CurrentTime fields could be different, only types of response shall be the same).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for SubscribeResponse message (see Annex A.17).

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

If the DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value included in UnacceptableInitialTerminationTimeFault.

Everything that happens at step 4 shall happen at step 6 as well. Otherwise, it indicates that the DUT processes namespaces in a wrong way and the test shall be failed.

Note: All requests for steps 5-6 to the DUT shall have default namespaces definition in each tag (see examples in Annex A.11).

7.4.2 EVENT - NAMESPACES (DEFAULT NAMESPACES FOR PARENT TAG)

Test Label: Event Service Different Namespaces Definition Test (Default Namespaces for Parent Tag).

Test Case ID: EVENT-4-1-7

ONVIF Core Specification Coverage: None

Command Under Test: None

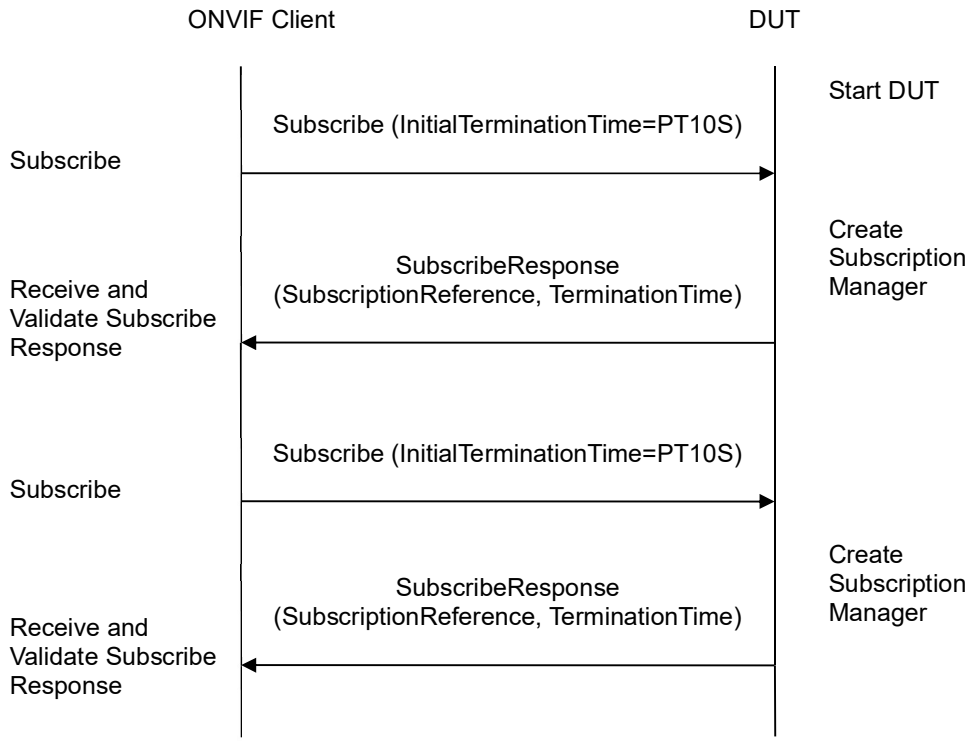
WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT accepts requests for Event Service with different namespaces definition.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke Subscribe message to instantiate a Subscription Manager. The Subscribe message does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.

4. Verify that the DUT sends SubscribeResponse message. Validate that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).

5. ONVIF Client will invoke Subscribe message to instantiate a Subscription Manager. The Subscribe message does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.

6. Verify that the DUT sends SubscribeResponse message. Validate that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SubscribeResponse message.

The DUT did not return a valid SubscriptionReference.

The DUT did not return valid values for CurrentTime and TerminationTime.

The DUT returned different results at step 4 and step 6 (EndpointReference, TerminationTime, CurrentTime fields could be different, only type of response shall be the same).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for SubscribeResponse message (see Annex A.17).

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

If the DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value included in UnacceptableInitialTerminationTimeFault.

Everything that happens at step 4 shall happen at step 6 as well. Otherwise, it indicates that the DUT processes namespaces in a wrong way and the test shall be failed.

Note: All requests for steps 5-6 to the DUT shall have default namespaces definition in parent tag (see examples in Annex A.11).

7.4.3 EVENT - NAMESPACES (NOT STANDARD PREFIXES)

Test Label: Event Service Different Namespaces Definition Test (Not Standard Prefixes).

Test Case ID: EVENT-4-1-8

ONVIF Core Specification Coverage: None

Command Under Test: None

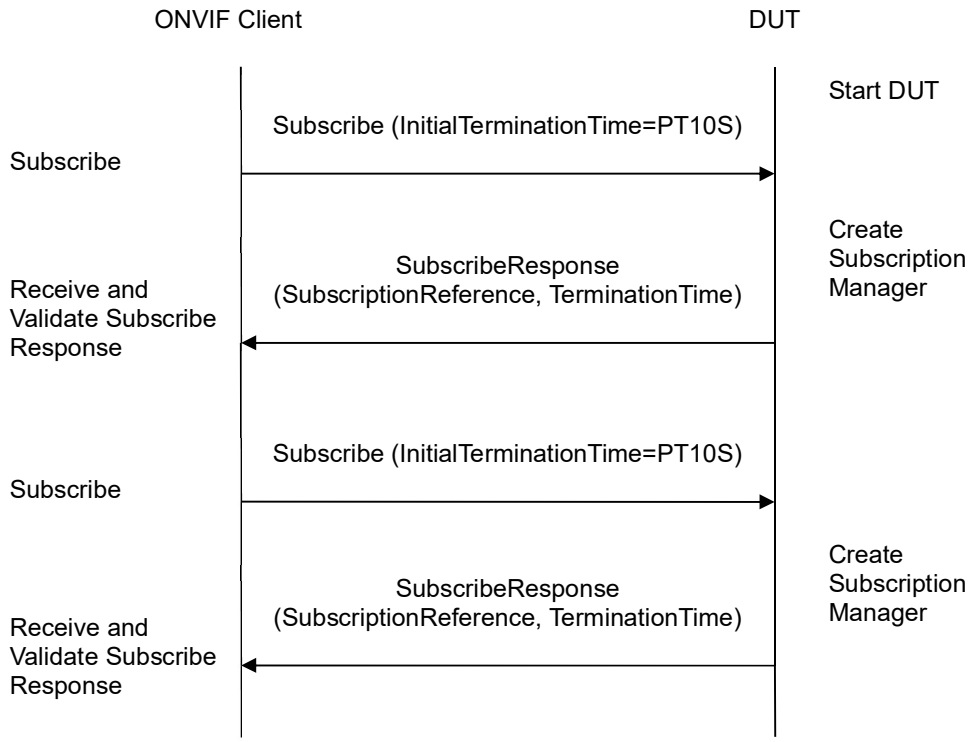
WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT accepts requests for Event Service with different namespaces definition.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke Subscribe message to instantiate a Subscription Manager. The Subscribe message does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.
4. Verify that the DUT sends SubscribeResponse message. Validate that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).
5. ONVIF Client will invoke Subscribe message to instantiate a Subscription Manager. The Subscribe message does not contain a TopicExpression or a Message Content Filter. The

Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.

6. Verify that the DUT sends SubscribeResponse message. Validate that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SubscribeResponse message.

The DUT did not return a valid SubscriptionReference

The DUT did not return valid values for CurrentTime and TerminationTime.

The DUT returned different results at step 4 and step 6 (EndpointReference, TerminationTime, CurrentTime fields could be different, only type of response shall be the same).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for SubscribeResponse message (see Annex A.17).

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

If the DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value included in UnacceptableInitialTerminationTimeFault.

Everything that happens at step 4 shall happen at step 6 as well. Otherwise, it indicates that the DUT processes namespaces in a wrong way and the test shall be failed.

Note: All requests for steps 5-6 to the DUT shall have namespaces definition with not standard prefixes (see examples in Annex A.11).

7.4.4 EVENT - NAMESPACES (DIFFERENT PREFIXES FOR THE SAME NAMESPACE)

Test Label: Event Service Different Namespaces Definition Test (Different Prefixes for the Same Namespace).

Test Case ID: EVENT-4-1-9

ONVIF Core Specification Coverage: None

Command Under Test: None

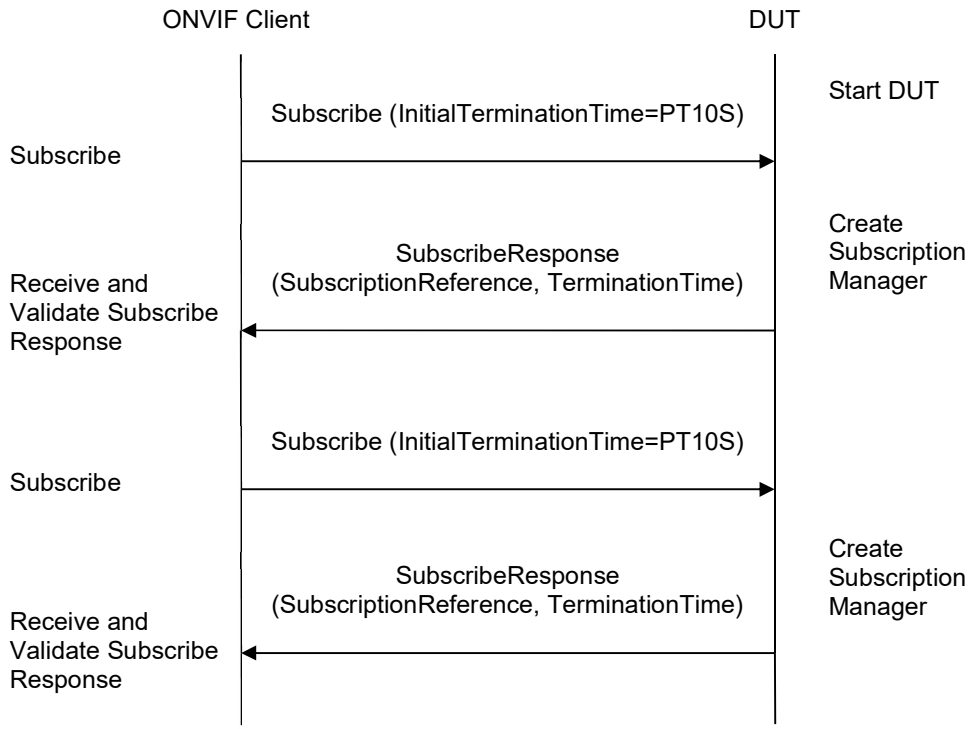
WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT accepts requests for Event Service with different namespaces definition.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke Subscribe message to instantiate a Subscription Manager. The Subscribe message does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.
4. Verify that ONVIF Client sends SubscribeResponse message. Validate that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).
5. ONVIF Client will invoke Subscribe message to instantiate a Subscription Manager. The Subscribe message does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.
6. Verify that the DUT sends SubscribeResponse message. Validate that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SubscribeResponse message.

The DUT did not return a valid SubscriptionReference

The DUT did not return valid values for CurrentTime and TerminationTime.

The DUT returned different results at step 4 and step 6 (EndpointReference, TerminationTime, CurrentTime fields could be different, only types of response shall be the same).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for SubscribeResponse message (see Annex A.17).

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

If the DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value which is contained in UnacceptableInitialTerminationTimeFault.

Everything that happens at step 4 shall happen at step 6 as well. Otherwise, it indicates that the DUT processes namespaces in a wrong way and the test shall be failed.

Note: All requests for steps 5-6 to the DUT shall have namespaces definition with different prefixes for the same namespace (see examples in Annex A.11).

7.4.5 EVENT - NAMESPACES (THE SAME PREFIX FOR DIFFERENT NAMESPACES)

Test Label: Event Service Different Namespaces Definition Test (the Same Prefix for Different Namespaces).

Test Case ID: EVENT-4-1-10

ONVIF Core Specification Coverage: None

Command Under Test: None

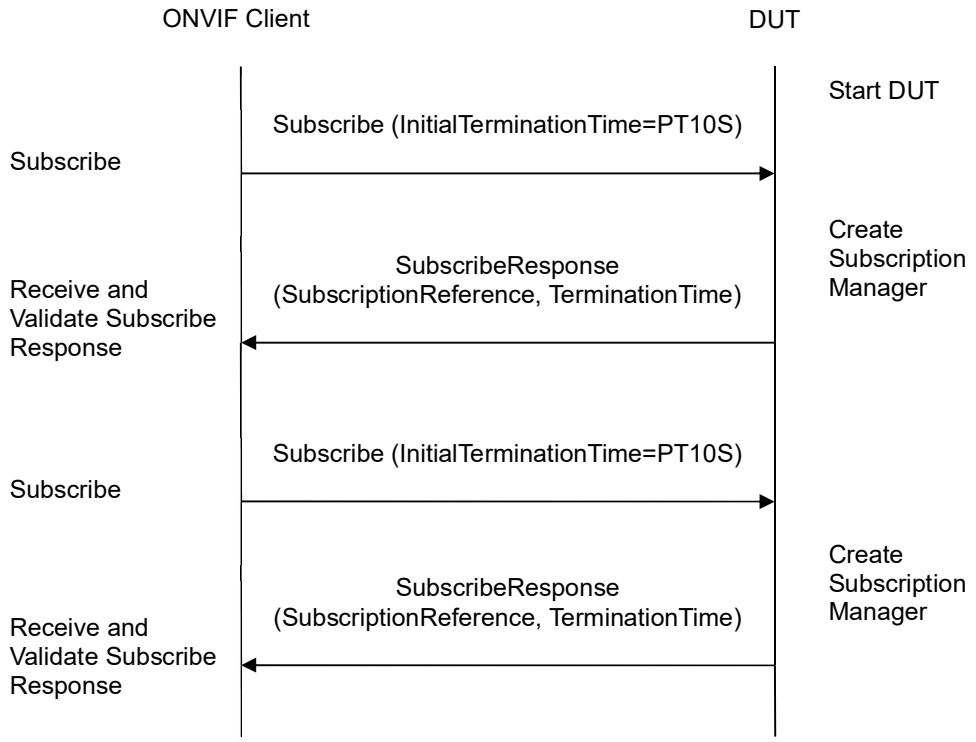
WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT accepts requests for Event Service with different namespaces definition.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke Subscribe message to instantiate a Subscription Manager. The Subscribe message does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.
4. Verify that the DUT sends SubscribeResponse message. Verify that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).
5. ONVIF Client will invoke Subscribe message to instantiate a Subscription Manager. The Subscribe message does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.
6. Verify that the DUT sends SubscribeResponse message. Validate that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send SubscribeResponse message.

The DUT did not return a valid SubscriptionReference

The DUT did not return valid values for CurrentTime and TerminationTime.

The DUT returned different results at step 4 and step 6 (EndpointReference, TerminationTime, CurrentTime fields could be different, only type of response shall be the same).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for SubscribeResponse message (see Annex A.17).

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

If the DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value included in UnacceptableInitialTerminationTimeFault.

Everything that happens at step 4 shall happen at step 6 as well. Otherwise, it indicates that the DUT processes namespaces in a wrong way and the test shall be failed.

Note: All requests for steps 5-6 to the DUT shall have namespaces definition with the same prefixes for different namespaces (see examples in Annex A.11).

7.5 Capabilities

7.5.1 EVENT SERVICE CAPABILITIES

Test Label: Event Service Capabilities Verification.

Test Case ID: EVENT-5-1-1

ONVIF Core Specification Coverage: Capability exchange

Command under test: GetServiceCapabilities (for Event Service)

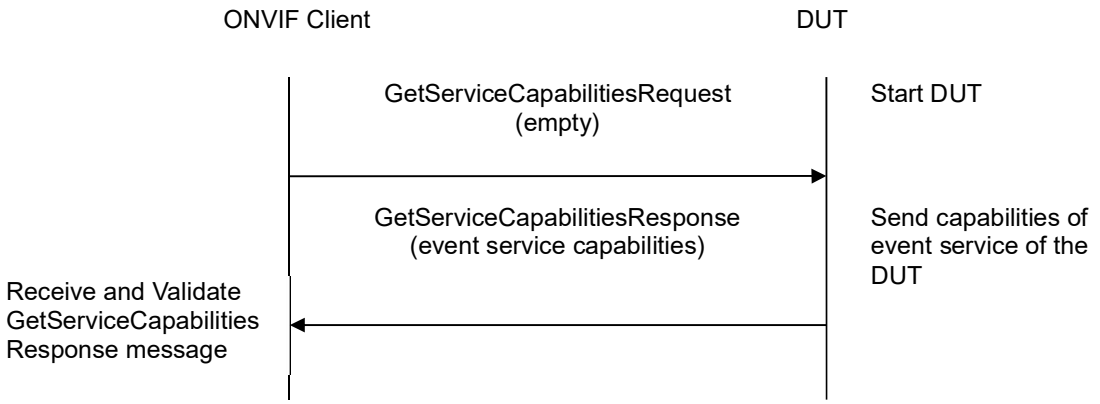
WSDL Reference: events.wsdl

Test Purpose: To verify Event Service Capabilities of the DUT.

Pre-Requisite: Event Service was received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetServiceCapabilitiesRequest message to retrieve event service capabilities of the DUT.
4. Verify the GetServiceCapabilitiesResponse from the DUT.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send valid GetServiceCapabilitiesResponse.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for GetServiceCapabilitiesResponse message (see Annex A.17).

7.5.2 GET SERVICES AND EVENT SERVICE CAPABILITIES CONSISTENCY

Test Label: Get Services and Event Service Capabilities Consistency Verification.

Test Case ID: EVENT-5-1-2

ONVIF Core Specification Coverage: Capability exchange

Command under test: GetServices, GetServiceCapabilities (for Event Service)

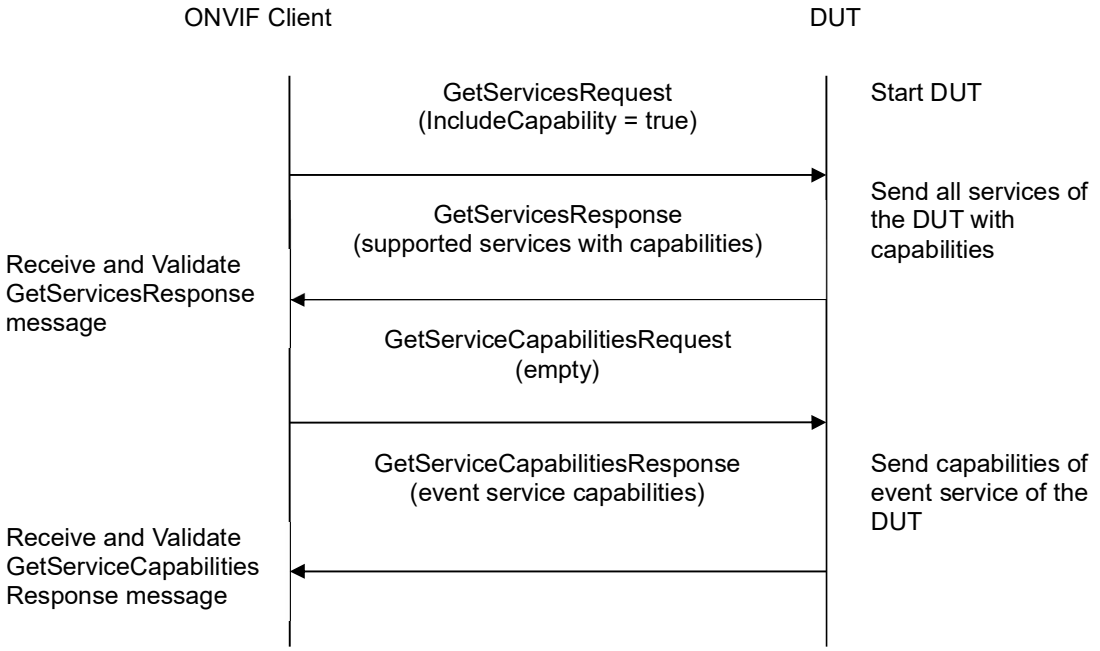
WSDL Reference: devicemgmt.wsdl, events.wsdl

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

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke GetServicesRequest message (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 GetServiceCapabilitiesRequest message to retrieve Event service capabilities of the DUT.
6. Verify the GetServiceCapabilitiesResponse message from the DUT.

Test Result:

PASS –

The DUT passed 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.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for GetServiceCapabilitiesResponse message (see Annex A.17).

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

7.6 Seek

7.6.1 SEEK EVENTS – BEGIN OF BUFFER

Test Label: Event Seek Verification.

Test Case ID: EVENT-6-1-3

ONVIF Core Specification Coverage: BeginOfBuffer (ONVIF Core Specification), Real-time Pull-Point Notification Interface (ONVIF Core Specification), Seek (ONVIF Core Specification), Persistent notification storage (ONVIF Core Specification)

Command under test: Seek

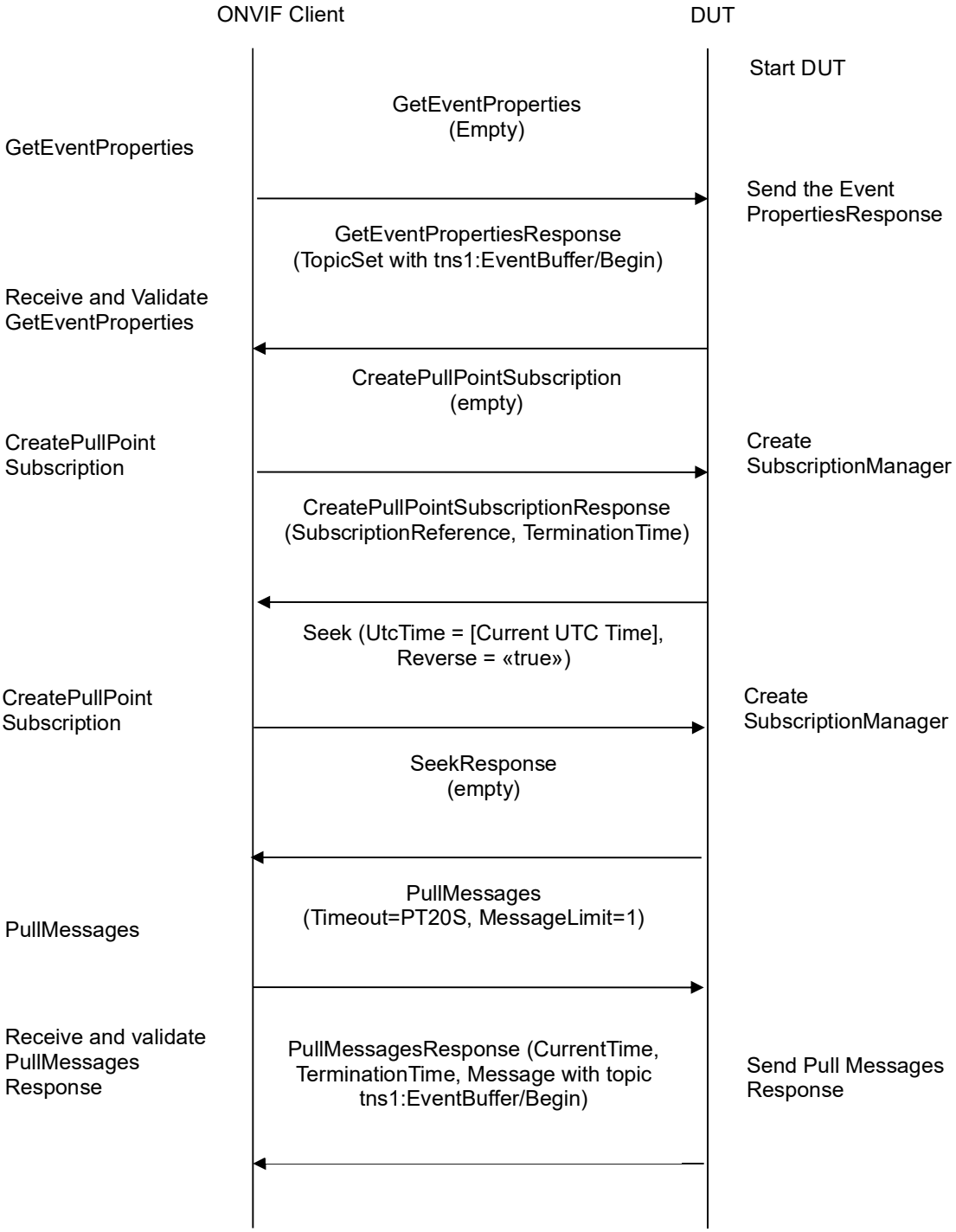
WSDL Reference: event.wsdl

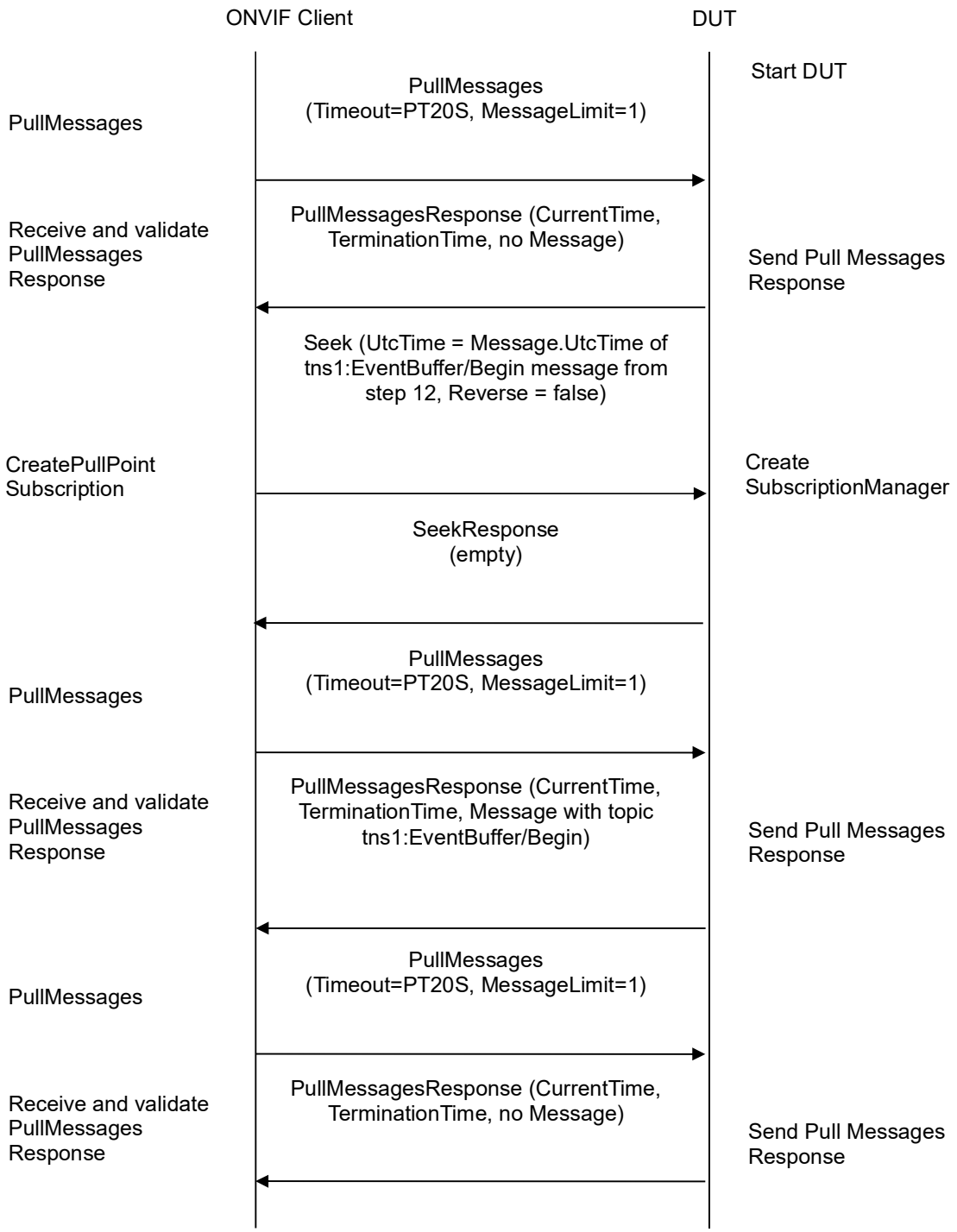
Test Purpose: To verify Seek function and that events are received from persistent notification storage when pull point was set before beginning of buffer. Verify tns1:EventBuffer/Begin event.

Pre-Requisite: Event Service was received from the DUT. Persistent notification storage is supported by the DUT. Persistent notification storage contains notifications.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
GetEventProperties
(Empty)
GetEventProperties
Send the Event PropertiesResponse
GetEventPropertiesResponse
(TopicSet with tns1:EventBuffer/Begin)
Receive and Validate GetEventProperties
CreatePullPointSubscription
(empty)
CreatePullPoint Subscription
Create SubscriptionManager
CreatePullPointSubscriptionResponse
(SubscriptionReference, TerminationTime)
Seek (UtcTime = [Current UTC Time], Reverse = «true»)
CreatePullPoint Subscription
Create SubscriptionManager
SeekResponse
(empty)
PullMessages
(Timeout=PT20S, MessageLimit=1)
PullMessages
Receive and validate PullMessages Response
PullMessagesResponse (CurrentTime, TerminationTime, Message with topic tns1:EventBuffer/Begin)
Send Pull Messages Response



Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client will invoke GetEventPropertiesRequest message 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:EventBuffer/Begin. If there is no event with such Topic, skip other steps, fail the test and go to the next test.
6. Check that this event is not a Property event (MessageDescription.IsProperty="false").
7. ONVIF Client will invoke CreatePullPointSubscription message with TopicFilter = tns1:EventBuffer/Begin.
8. Verify that the DUT sends a CreatePullPointSubscriptionResponse. Validate that correct values for CurrentTime and TerminationTime and SubscriptionReference are returned.
9. ONVIF Client will invoke Seek message (UtcTime = [Current UTC Time], Reverse = true) to start reverse seek.
10. Verify that the DUT sends a SeekResponse.
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 NotificationMessage.
13. Verify NotificationMessage (a maximum number of 1 Notification Messages is included in the PullMessages Response; well formed and valid values for CurrentTime and TerminationTime (TerminationTime>CurrentTime).
14. Verify that received event is event with Topic = tns1:EventBuffer/Begin.
15. ONVIF Client will invoke PullMessages command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
16. Verify that the DUT sends a PullMessagesResponse that contains no NotificationMessages.
17. ONVIF Client will invoke Seek message (UtcTime = Message.UtcTime of tns1:EventBuffer/Begin message from step 12, Reverse = false) to put pull point in the past.
18. Verify that the DUT sends a SeekResponse.
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 that contains one NotificationMessage.
21. Verify NotificationMessage (a maximum number of 1 Notification Messages is included in the PullMessages Response; well formed and valid values for CurrentTime and TerminationTime (TerminationTime>CurrentTime).
22. Verify that received event is event with Topic = tns1:EventBuffer/Begin.
23. ONVIF Client will invoke PullMessages command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
24. Verify that the DUT sends a PullMessagesResponse that contains no NotificationMessages.

25. Verify that there was PullMessagesResponse without messages.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).

The DUT did not send a PullMessagesResponse.

The DUT did not send a GetEventsPropertiesResponse.

The PullMessagesResponse does not contain a NotificationMessage.

The PullMessagesResponse contains more than 1 NotificationMessages.

The NotificationMessages are not well formed.

The PullMessagesResponse contains invalid values for Current or TerminationTime.

The DUT did not send event with Topic = tns1:EventBuffer/Begin from persistent notification storage for step 12 and 20.

The DUT sent event for step 16 and 24.

The DUT did not return correct event with Topic = tns1:EventBuffer/Begin in GetEventsPropertiesResponse.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for CreatePullPointSubscriptionResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for SetSynchronizationPointResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for PullMessagesResponse message (see Annex A.17).

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

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

7.6.2 SEEK EVENTS

Test Label: Event Seek Verification.

Test Case ID: EVENT-6-1-4

ONVIF Core Specification Coverage: Real-time Pull-Point Notification Interface (ONVIF Core Specification), Seek (ONVIF Core Specification), Persistent notification storage (ONVIF Core Specification)

Command under test: Seek

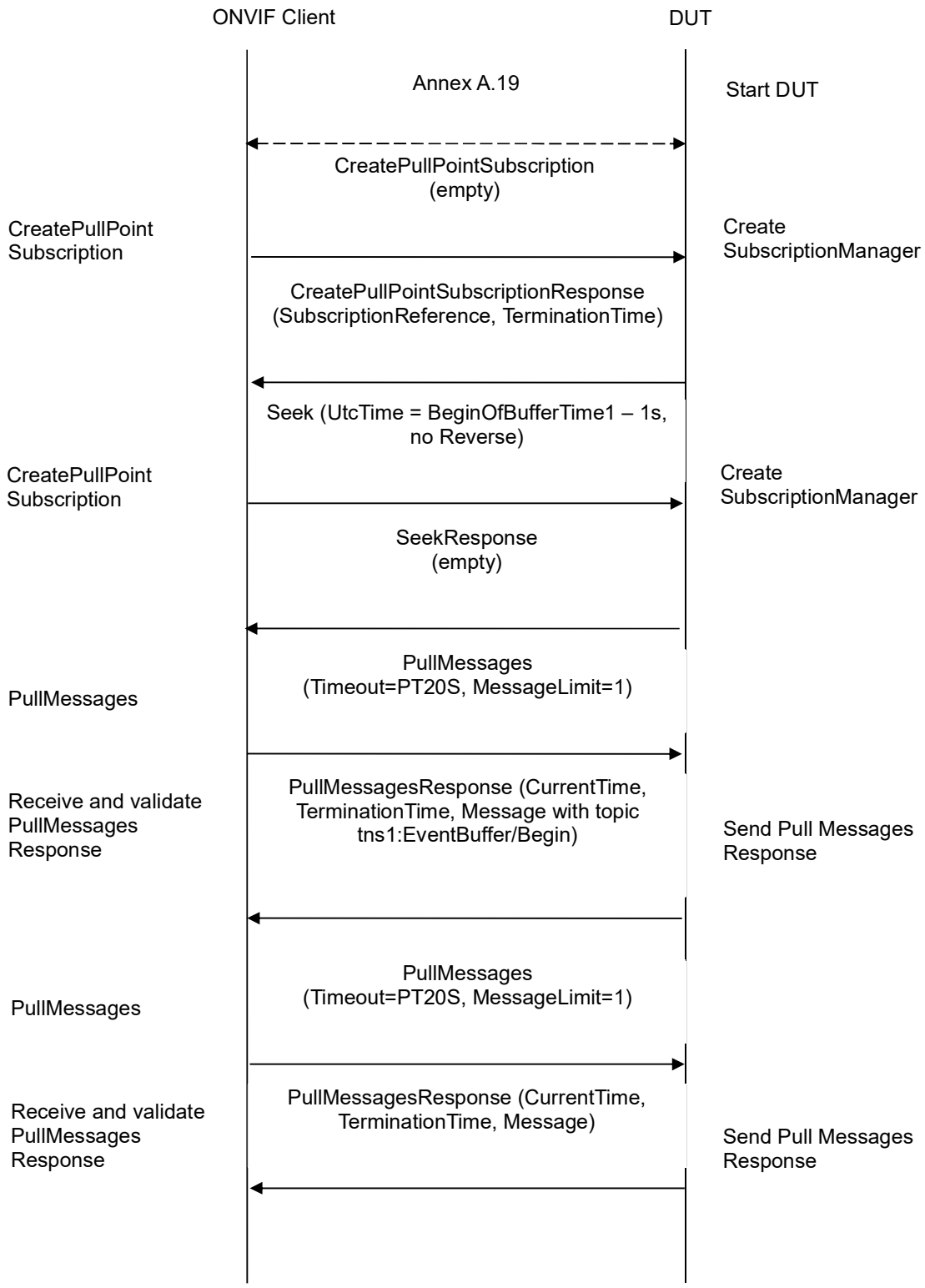
WSDL Reference: event.wsdl

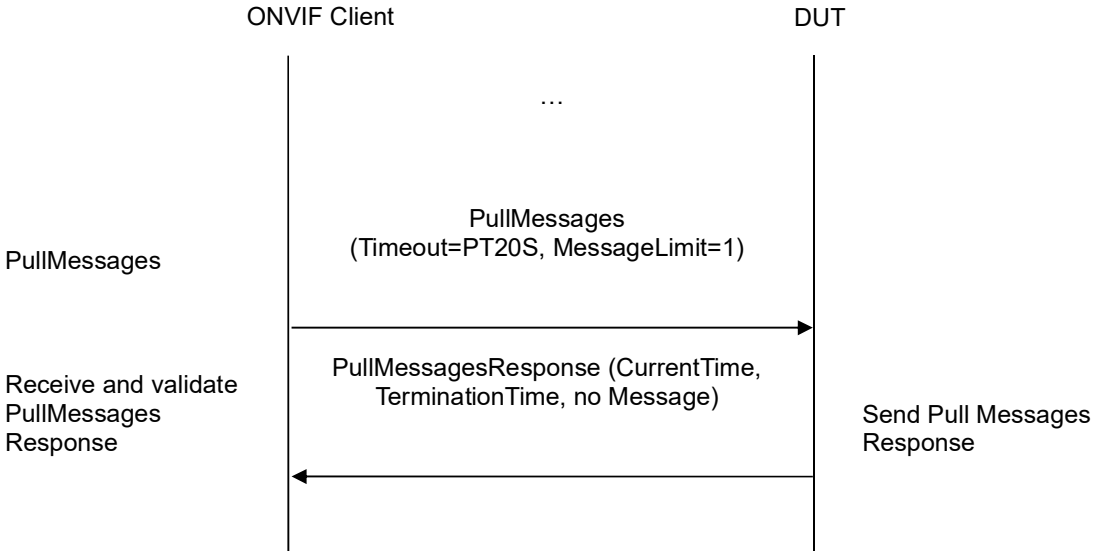
Test Purpose: To verify Seek function and those events are received from persistent notification storage.

Pre-Requisite: Event Service was received from the DUT. Persistent notification storage is supported by the DUT. Persistent notification storage contains notifications.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Annex A.19
Start DUT
CreatePullPointSubscription
(empty)
CreatePullPoint
Subscription
Create
SubscriptionManager
CreatePullPointSubscriptionResponse
(SubscriptionReference, TerminationTime)
Seek (UtcTime = BeginOfBufferTime1 – 1s,
no Reverse)
CreatePullPoint
Subscription
Create
SubscriptionManager
SeekResponse
(empty)
PullMessages
(Timeout=PT20S, MessageLimit=1)
PullMessages
PullMessagesResponse (CurrentTime,
TerminationTime, Message with topic
tns1:EventBuffer/Begin)
Receive and validate
PullMessages
Response
Send Pull Messages
Response
PullMessages
(Timeout=PT20S, MessageLimit=1)
PullMessages
PullMessagesResponse (CurrentTime,
TerminationTime, Message)
Receive and validate
PullMessages
Response
Send Pull Messages
Response



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Find the beginning of buffer time BeginOfBufferTime1 (see Annex A.19)
4. ONVIF Client will invoke CreatePullPointSubscription message.
5. Verify that the DUT sends a CreatePullPointSubscriptionResponse. Validate that correct values for CurrentTime and TerminationTime and SubscriptionReference are returned.
6. ONVIF Client will invoke Seek message (UtcTime = BeginOfBufferTime1 – 1s, no Reverse) to put pull point in the past.
7. Verify that the DUT sends a SeekResponse.
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.
10. Verify NotificationMessage (a maximum number of 1 Notification Messages is included in the PullMessages Response; well formed and valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).
11. Verify that Message.UtcTime is greater or equal to BeginOfBufferTime1.
12. Repeat steps 8-11 until Message.UtcTime is equal or greater than time of Pullpoint Subscrbtion creation (CurrentTime from step 5) or there are PullMessagesResponse without Notifications to get all messages from persistent notification storage.
13. Verify that the first notification has Topic = tns1:EventBuffer/Begin.

14. Verify that all events across PullMessagesResponses are ordered by increasing Message.UtcTime and if events occur in the incorrect order that they are only misplaced by a maximum of 5 seconds.
15. Verify that there is at least one message from persistent notification storage.
16. Verify that all Messages have Message.UtcTime that is greater or equal to BeginOfBufferTime1.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).

The DUT did not send a PullMessagesResponse.

The DUT did not send a SeekResponse.

The PullMessagesResponse does not contain a NotificationMessage.

The PullMessagesResponse contains more than 1 NotificationMessages.

The NotificationMessages are not well formed.

The PullMessagesResponse contains invalid values for Current or TerminationTime.

The DUT did not return at least one NotificationMessage from persistent notification storage.

The DUT did not return NotificationMessages from persistent notification storage ordered by increasing Message.UtcTime and events that occurred in the incorrect order were misplaced by more than 5 seconds.

The DUT returned NotificationMessages from persistent notification storage that has Message.UtcTime < BeginOfBufferTime1.

The DUT returned the first Notification with Topic which differs from tns1:EventBuffer/Begin.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for CreatePullPointSubscriptionResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for SeekResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for PullMessagesResponse message (see Annex A.17).

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

Note: If the DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit included in

PullMessagesFaultResponse.

7.6.3 SEEK EVENTS – REVERSE

Test Label: Event Seek Verification - Reverse.

Test Case ID: EVENT-6-1-5

ONVIF Core Specification Coverage: BeginOfBuffer (ONVIF Core Specification), Real-time Pull-Point Notification Interface (ONVIF Core Specification), Seek (ONVIF Core Specification), Persistent notification storage (ONVIF Core Specification)

Command under test: Seek

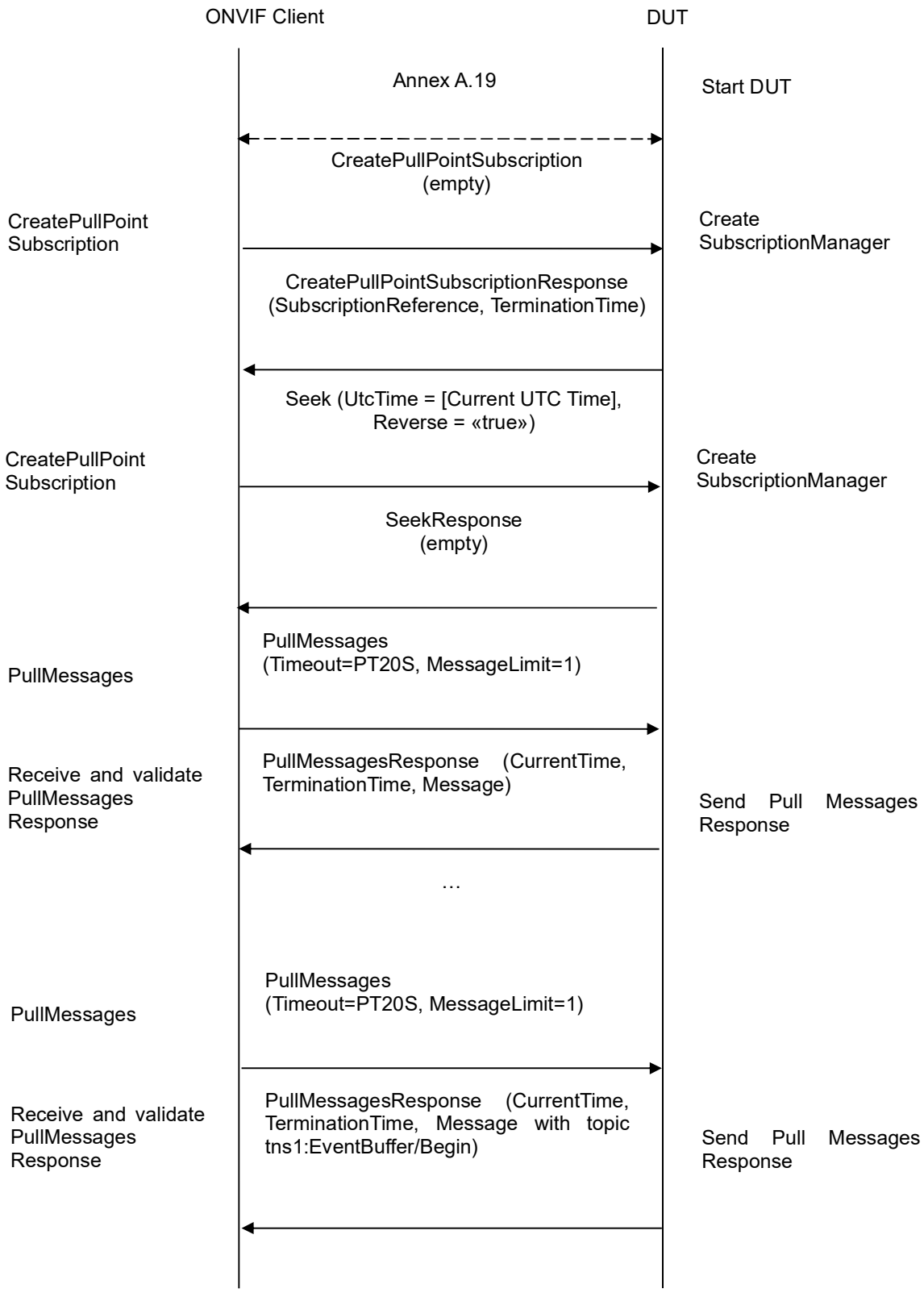
WSDL Reference: event.wsdl

Test Purpose: To verify Seek function and those events are received from persistent notification storage in reverse order. Verify tns1:EventBuffer/Begin event.

Pre-Requisite: Event Service was received from the DUT. Persistent notification storage is supported by the DUT. Persistent notification storage contains notifications.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Annex A.19
Start DUT
CreatePullPointSubscription
(empty)
CreatePullPoint Subscription
Create SubscriptionManager
CreatePullPointSubscriptionResponse
(SubscriptionReference, TerminationTime)
Seek (UtcTime = [Current UTC Time], Reverse = «true»)
CreatePullPoint Subscription
Create SubscriptionManager
SeekResponse
(empty)
PullMessages
(Timeout=PT20S, MessageLimit=1)
PullMessages
PullMessagesResponse (CurrentTime, TerminationTime, Message)
Receive and validate PullMessages Response
Send Pull Messages Response
…
PullMessages
(Timeout=PT20S, MessageLimit=1)
PullMessages
PullMessagesResponse (CurrentTime, TerminationTime, Message with topic tns1:EventBuffer/Begin)
Receive and validate PullMessages Response
Send Pull Messages Response

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Find beginning of buffer time BeginOfBufferTime1 (see Annex A.19)
4. ONVIF Client will invoke CreatePullPointSubscription message.
5. Verify that the DUT sends a CreatePullPointSubscriptionResponse. Validate that correct values for CurrentTime and TerminationTime and SubscriptionReference are returned.
6. ONVIF Client will invoke Seek message (UtcTime = [Current UTC Time], Reverse = «true») to start reverse seek.
7. Verify that the DUT sends a SeekResponse.
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.
10. Verify NotificationMessage (a maximum number of 1 Notification Messages is included in the PullMessages Response; well formed and valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).
11. Verify that Message.UtcTime is greater or equal to BeginOfBufferTime1.
12. Repeat steps 8-11 until Notification with Topic tns1:EventBuffer/Begin is received or there is PullMessagesResponse without Notifications to get all messages from persistent notification storage.
13. Verify that the last notification has Topic = tns1:EventBuffer/Begin.
14. Verify that all events across PullMessagesResponses are ordered by decreasing Message.UtcTime and if events occur in the incorrect order that they are only misplaced by a maximum of 5 seconds.
15. Verify that there is at least one message from persistent notification storage.
16. Verify that all Messages have Message.UtcTime that is greater or equal to BeginOfBufferTime1.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send CreatePullPointSubscriptionResponse message.

The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).

The DUT did not send a PullMessagesResponse.

The DUT did not send a SeekResponse.

The PullMessagesResponse does not contain a NotificationMessage.

The PullMessagesResponse contains more than 1 NotificationMessages.

The NotificationMessages are not well formed.

The PullMessagesResponse contains invalid values for Current or TerminationTime.

The DUT did not return at least one NotificationMessage from persistent notification storage.

The DUT did not return NotificationMessages from persistent notification storage ordered by decreasing Message.UtcTime and events that occurred in the incorrect order were misplaced by more than 5 seconds.

The DUT returned NotificationMessages from persistent notification storage that has Message.UtcTime < BeginOfBufferTime1.

The DUT returned the last Notification with Topic which differs from tns1:EventBuffer/Begin.

The DUT did not send valid WS-Addressing Action URI in SOAP Header for CreatePullPointSubscriptionResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for SeekResponse message (see Annex A.17).

The DUT did not send valid WS-Addressing Action URI in SOAP Header for PullMessagesResponse message (see Annex A.17).

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

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

8 Security Test Cases

8.1 USER TOKEN PROFILE

Test Label: Security – User token profile

Test Case ID: SECURITY-1-1-1

ONVIF Core Specification Coverage: Message level security

Command Under Test: None

WSDL Reference: None

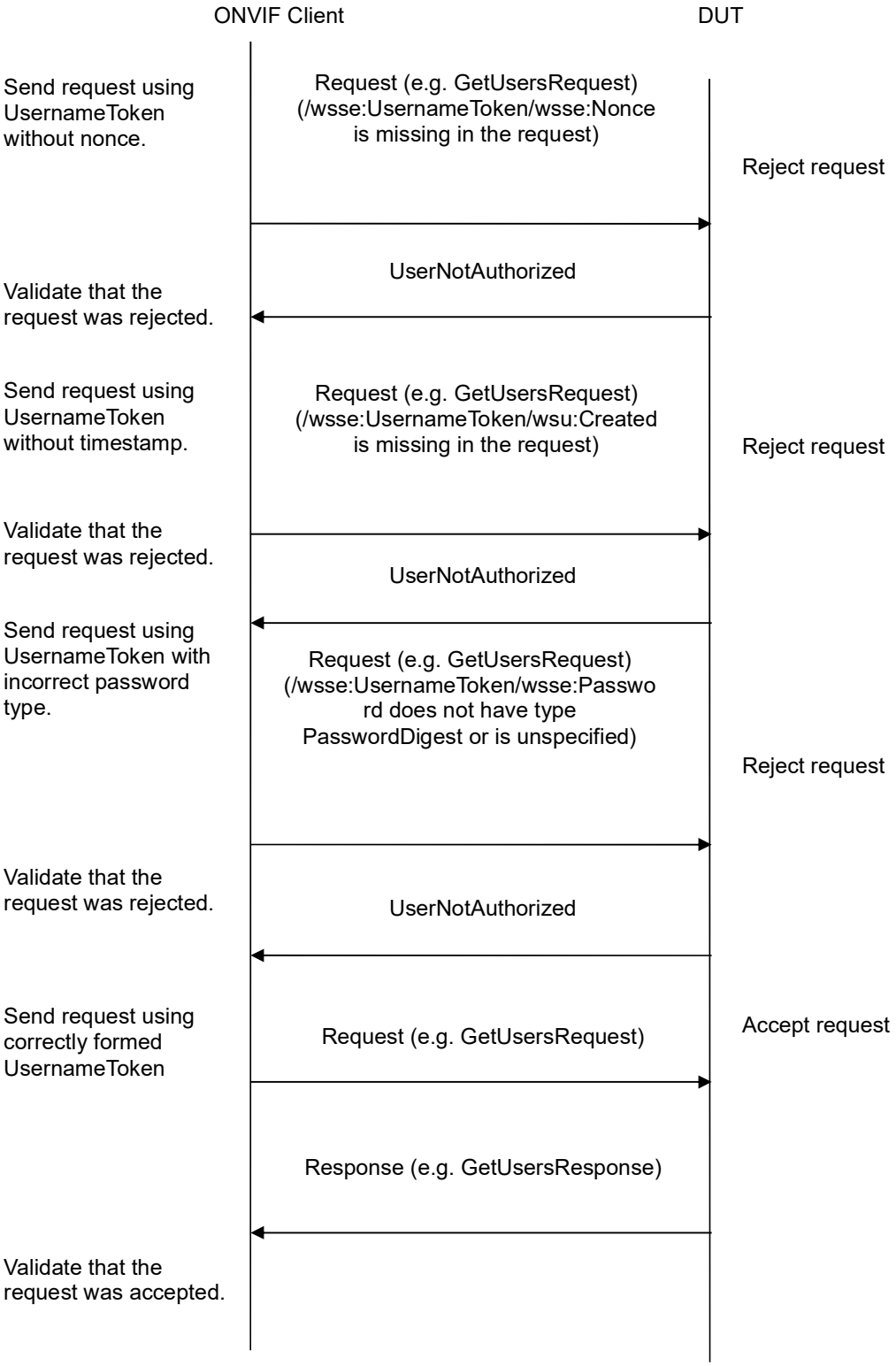
Test Purpose: To verify that the DUT supports the User Token Profile for Message level security.

Pre-Requisite:

- A user with Administrator rights (and password set) is needed in this test case. If such a user does not exist, then one shall be created before the test is executed. Similarly, if there exists such a user but the password has not been set, then the password shall be set before the test is executed. If any such changes to user settings are needed for this test case, then they should be done before starting the test sequence and the DUT should be reset to its original settings when the test sequence is finished.
- At least one operation that requires authentication is needed in this test case. If the example given in this test case (GetUsers) does not require authentication, then the test operator shall choose another operation that does require authentication.

Test Configuration: DUT and ONVIF Client

Test Sequence:


ONVIF Client
DUT
Send request using UsernameToken without nonce.
Request (e.g. GetUsersRequest) (/wsse:UsernameToken/wsse:Nonce is missing in the request)
Reject request
UserNotAuthorized
Validate that the request was rejected.
Send request using UsernameToken without timestamp.
Request (e.g. GetUsersRequest) (/wsse:UsernameToken/wsu:Created is missing in the request)
Reject request
Validate that the request was rejected.
UserNotAuthorized
Send request using UsernameToken with incorrect password type.
Request (e.g. GetUsersRequest) (/wsse:UsernameToken/wsse:Password does not have type PasswordDigest or is unspecified)
Reject request
Validate that the request was rejected.
UserNotAuthorized
Send request using correctly formed UsernameToken
Request (e.g. GetUsersRequest)
Accept request
Response (e.g. GetUsersResponse)
Validate that the request was accepted.

Test Procedure:

1. ONVIF Client sends a request that requires authentication (e.g. GetUsers) to the DUT with incorrectly implemented UsernameToken. Each of the following shall be tested:
   - Missing nonce.
   - Missing timestamp.
   - Incorrect password type.
2. Verify that the DUT rejects all incorrect requests.
3. ONVIF Client sends a request (e.g. GetUsers) to the DUT with correctly formatted UsernameToken.
4. Verify that the DUT accepts the correct request.

Test Result

PASS –

The DUT passed all assertions.

FAIL –

DUT does not support Username Token profile.

DUT accepts Username Token without nonce.

DUT accepts Username Token without timestamp.

DUT accepts Username Token without password type “PasswordDigest”.

DUT rejects Username Token with nonce and timestamp.

Note: Below is an example of a correctly formed UsernameToken for message level security, with nonce, timestamp and correct password type. For further details, refer to [ONVIF Network Interface Specs] and [WS-Security].

```
<SOAP:Envelope xmlns:SOAP="..." xmlns:wsse="..." xmlns:wsu= "...">
  <SOAP:Header>
    ...
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>...</wsse:Username>
        <wsse:Password Type="...#PasswordDigest">…</wsse:Password>
        <wsse:Nonce>…</wsse:Nonce>
```

```
        <wsu:Created>…</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
    ...
  </SOAP:Header>
  ...
</SOAP:Envelope>
```

Note: Other types of authentication shall not be used during this test.

8.2 DIGEST AUTHENTICATION

Test Label: Security – HTTP Digest Authentication.

Test Case ID: SECURITY-1-1-2

ONVIF Core Specification Coverage: Security

Command under test: None

WSDL Reference: None

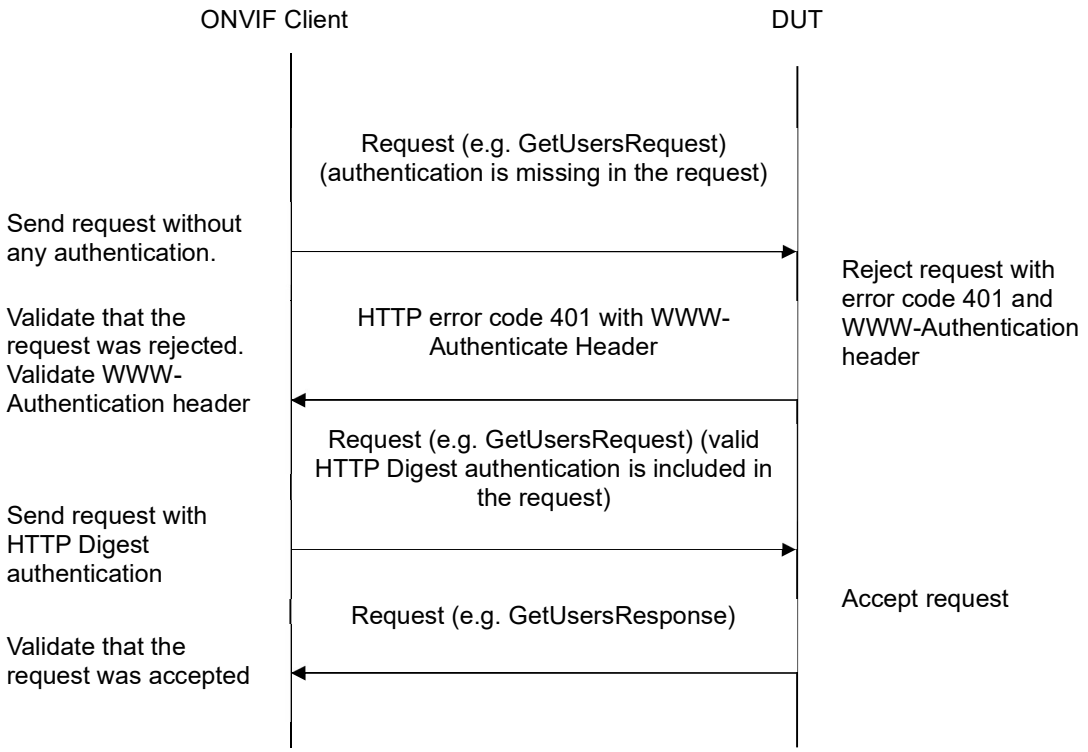
Test Purpose: To verify that the DUT supports the HTTP Digest Authentication for HTTP level security.

Pre-Requisite:

- A user with Administrator rights (and password set) is needed in this test case. If such a user does not exist, then one shall be created before the test is executed. Similarly, if there exists such a user but the password has not been set, then the password shall be set before the test is executed. If any such changes to user settings are needed for this test case, then they should be done before starting the test sequence and the DUT should be reset to its original settings when the test sequence is finished.
- At least one operation that requires authentication is needed in this test case. If the example given in this test case (GetUsers) does not require authentication, then the test operator shall choose another operation that does require authentication.

Test Configuration: ONVIF Client and DUT

Test Sequence:



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client sends a request that requires authentication (e.g. GetUsers) to the DUT without any authentication.
4. Verify that the DUT rejects the request with HTTP error code 401.
5. Verify that DUT returns the required information in HTTP Response Header.
6. Send a valid request with HTTP Digest Authentication.
7. Verify that the DUT accepts the correct request.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send HTTP error code 401 at step 3.

The DUT did not send valid WWW-Authenticate Header at step 3 (there are no Digest indications, no realm or no nonce).

DUT rejects request with a valid HTTP digest authentication.

Note: No WS-UsernameToken authentication will be used in requests for this test.

Annex A

This section describes the meaning of the following definitions. These definitions are used in the test case description.

A.1 Invalid Device Type and Scope Type

Device Type in the <d:Types> declaration:

- <d:Types> list shall include “dn:NetworkVideoTransmitter” or “tds:Device” depending on supported types, otherwise it is considered as invalid <d:Types> value.

Device Type namespaces shall also be declared in a message:

- http://www.onvif.org/ver10/network/wsdl for “dn:NetworkVideoTransmitter”
- http://www.onvif.org/ver10/device/wsdl for “tds:Device”

Invalid Scope Type:

- Scope URI is not formed according to the rules of RFC 3986.

A.2 Invalid Hostname, DNSname

A string which is not formed according to the rules of RFC 952 and RFC 1123 is considered as an invalid string.

A.3 Invalid TimeZone

The Time Zone format is specified by POSIX, refer to POSIX 1003.1 section 8.3.

Example: Europe, Paris TZ=CET-1CEST,M3.5.0/2,M10.5.0/3

CET = designation for standard time when daylight saving is not in force.

-1 = offset in hours = negative so 1 hour east of Greenwich meridian.

CEST = designation when daylight saving is in force ("Central European Summer Time")

, = no offset number between code and comma, so default to one hour ahead for daylight saving

M3.5.0 = when daylight saving starts = the last Sunday in March (the "5th" week means the last in the month)

/2, = the local time when the switch occurs = 2 a.m. in this case

M10.5.0 = when daylight saving ends = the last Sunday in October.

/3, = the local time when the switch occurs = 3 a.m. in this case

A TimeZone token which is not formed according to the rules of POSIX 1003.1 section 8.3 is considered as an invalid time zone.

A.4 Invalid SOAP 1.2 Fault Message

A SOAP 1.2 fault message which is not formed according to the rules defined in SOAP 1.2, Part 1 Section 5.4 is considered as invalid.

A.5 Invalid WSDL URL

An URL which is not formed according to the rules of RFC 3986 is considered as an invalid WSDL URL.

A.6 Valid/Invalid IPv4 Address

IPv4 Address token is represented in dotted decimal notation (32 bit internet address is divided into four 8-bit fields and each field is represented in decimal number separated by a dot).

Valid IPv4 addresses are in the range 0.0.0.0 to 255.255.255.255 excluding 0/8, 255/8, and 127/8, as defined in RFC 758, and 169.254/16 as defined in RFC 3927.

Valid IPv4 addresses for a device shall be valid according to the defined network mask and gateway (the gateway shall be reachable and shall not be identical to the assigned IPv4 address).

Reserved addresses such as 240.0.0.0 through 255.255.255.254, as defined in RFC 2780 are prohibited for IPv4 devices.

A.7 WS-Discovery timeout value

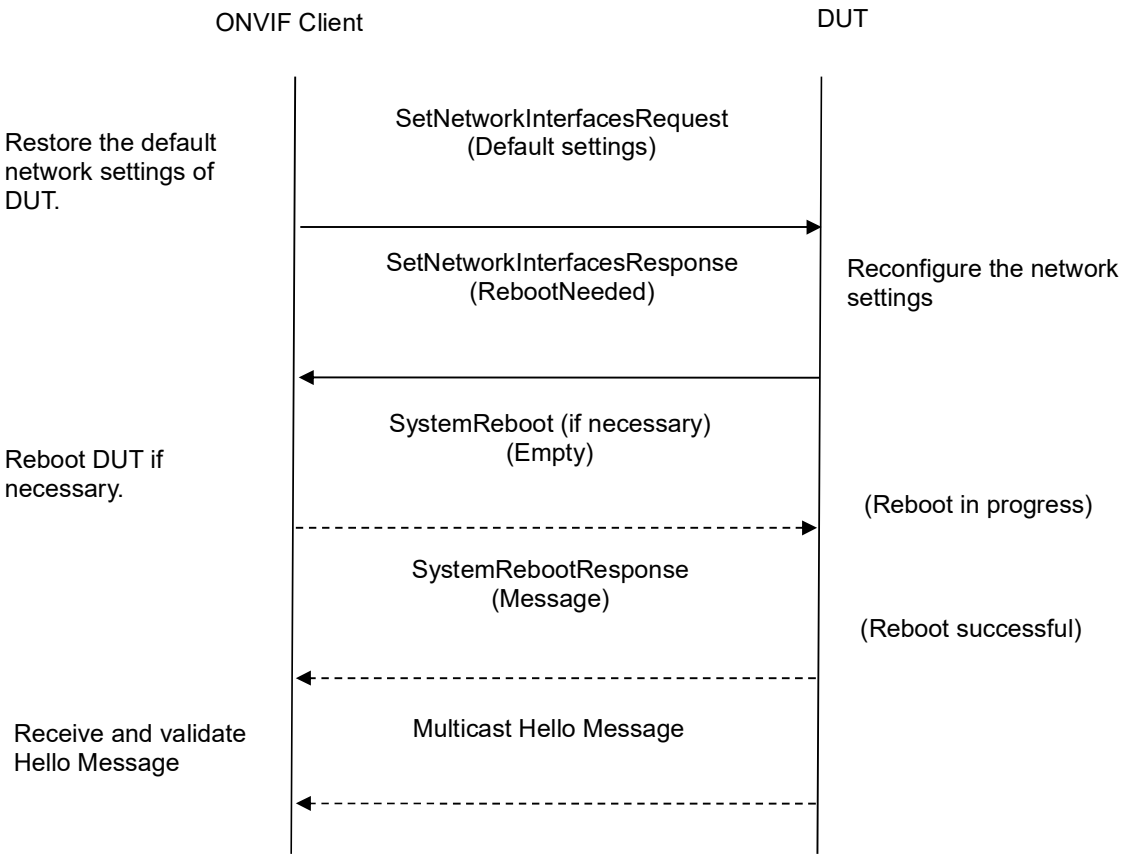
The ONVIF Client will use a hard coded timeout value (DISCOVERY_TIMEOUT) when waiting for discovery responses. The value for this timeout in the respective Test result steps is:

DISCOVERY_TIMEOUT = 5 sec.

A.8 Restore Network Settings

When the default network settings of the DUT are changed during the execution of Network configuration related test cases, ONVIF Client follows the following procedure to restore the original default settings at the end of actual test sequence.

1. Restore the default network settings by invoking SetNetworkInterfaces (**Default settings**) command.
2. If Reboot is needed by the DUT, invoke SystemReboot command.
3. If SystemReboot is invoked, wait for HELLO message from the default network interface.
4. Move to the next test case execution.



A.9 Subscribe and CreatePullPointSubscription for receiving all Events

When subscribing for events an ONVIF Client might be interested in receiving all or some of the Events produced by the DUT.

If an ONVIF Client is interested in receiving some events it includes a filter tag in the CreatePullPointSubscription or Subscribe requests describing the events which the ONVIF Client is interested in (see examples in the Core Spec).

If an ONVIF Client is interested in receiving all events from a device it does not include the Filter sub tag in the Subscribe or CreatePullPointSubscription request.

Example:

The following Subscribe and CreatePullPointSubscription requests can be used if an ONVIF Client is interested in receiving all events.

1) Subscribe Request:

```
<m:Subscribe xmlns:m="http://docs.oasis-open.org/wsn/b-2"
xmlns:m0="http://www.w3.org/2005/08/addressing">
```

```
  <m:ConsumerReference>
    <m0:Address>
      http://192.168.0.1/events
    </m0:Address>
    </m:ConsumerReference>
</m:Subscribe>
```

2) CreatePullPointSubscriptionRequest

```
<m:CreatePullPointSubscription xmlns:m="http://www.onvif.org/ver10/events/wsdl"/>
```

A.10 Valid expression indicating empty IP Address

If a certain IP Address is not set (e.g. an NTP or a DNS Address) the device can use one of the following three possibilities:

1) Use 0.0.0.0 as empty IP Address

```
<IPAddress>0.0.0.0</IPAddress>
```

2) Use an empty IP Address tag

```
<IPAddress/>
```

3) Omit the IP Address tag (if it is an optional element)

Example:

1) Use 0.0.0.0

```
<m:GetDNSResponse xmlns:m="http://www.onvif.org/ver10/device/wsdl"
xmlns:m0="http://www.onvif.org/ver10/schema">
  <m:DNSInformation>
    <m0:FromDHCP>false</m0:FromDHCP>
    <m0:DNSManual>
    <m0:Type>IPv4</m0:Type>
      <m0:IPv4Address>0.0.0.0</m0:IPv4Address>
    </m0:DNSManual>
  </m:DNSInformation>
</m:GetDNSResponse>
```

2) Use an empty tag

```
<m:GetDNSResponse xmlns:m="http://www.onvif.org/ver10/device/wsdl"
xmlns:m0="http://www.onvif.org/ver10/schema">

  <m:DNSInformation>

    <m0:FromDHCP>false</m0:FromDHCP>

    <m0:DNSManual>

      <m0:Type>IPv4</m0:Type>

      <m0:IPv4Address/>

    </m0:DNSManual>

  </m:DNSInformation>

</m:GetDNSResponse>
```

3) Omit tag

```
<m:GetDNSResponse xmlns:m="http://www.onvif.org/ver10/device/wsdl"
xmlns:m0="http://www.onvif.org/ver10/schema">

  <m:DNSInformation>

    <m0:FromDHCP>false</m0:FromDHCP>

      <m0:DNSManual>

      <m0:Type>IPv4</m0:Type>

    </m0:DNSManual>

  </m:DNSInformation>

</m:GetDNSResponse>
```

A.11 Example of Requests for Namespaces Test Cases

For the execution of namespaces test cases, ONVIF Client shall send a request with specific namespaces definition. Examples of how this request shall look like are the following.

1) Defaults Namespaces Definition in Each Tag Examples

GetDNSRequest message example:

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">

  <Header xmlns="http://www.w3.org/2003/05/soap-envelope">

    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">

      <UsernameToken xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
```

```
      <Username xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">user</Username>

      <Password xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" Type="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-
1.0#PasswordDigest">5zjIbmvWxVevGlpqg6Qnt9h8Fmo=</Password>

      <Nonce xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">ikcoiK+AmJvA5UpfxTzG8Q==</Nonce>

      <Created xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">2011-04-25T09:27:48Z</Created>

     </UsernameToken>

   </Security>

  </Header>

  <Body xmlns="http://www.w3.org/2003/05/soap-envelope">

    <GetDNS xmlns="http://www.onvif.org/ver10/device/wsdl" />

  </Body>

</Envelope>
```

SetDNSRequest message example:

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">

  <Header xmlns="http://www.w3.org/2003/05/soap-envelope">

    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">

      <UsernameToken xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">

        <Username xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">service</Username>

        <Password xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" Type="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-
1.0#PasswordDigest">znYLkZuoGqC5RrFD4KDs529JvHI=</Password>

        <Nonce xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">7L0dq2/ZF3zWYEpQpFhcHA==</Nonce>

        <Created xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">2011-04-25T09:27:49Z</Created>

      </UsernameToken>

    </Security>

  </Header>

  <Body xmlns="http://www.w3.org/2003/05/soap-envelope">

    <SetDNS xmlns="http://www.onvif.org/ver10/device/wsdl">
```

```
      <FromDHCP xmlns="http://www.onvif.org/ver10/device/wsdl">false</FromDHCP>

      <DNSManual xmlns="http://www.onvif.org/ver10/device/wsdl">

        <Type xmlns="http://www.onvif.org/ver10/schema">IPv4</Type>

        <IPv4Address
xmlns="http://www.onvif.org/ver10/schema">10.1.1.1</IPv4Address>

      </DNSManual>

    </SetDNS>

  </Body>

</Envelope>
```

GetCapabilitiesRequest message example:

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">

  <Header xmlns="http://www.w3.org/2003/05/soap-envelope">

    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">

      <UsernameToken xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">

        <Username xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">service</Username>

        <Password xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" Type="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-
1.0#PasswordDigest">tg6EnHtyMWW8eUfntHO6XpPjOsg=</Password>

        <Nonce xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">iBIGpSuHtNPbdSWGzG48ng==</Nonce>

        <Created xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">2011-04-25T10:54:34Z</Created>

      </UsernameToken>

    </Security>

  </Header>

  <Body xmlns="http://www.w3.org/2003/05/soap-envelope">

    <GetCapabilities xmlns="http://www.onvif.org/ver10/device/wsdl">

      <Category xmlns="http://www.onvif.org/ver10/device/wsdl">Events</Category>

    </GetCapabilities>

  </Body>

</Envelope>
```

SubscribeRequest message example:

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">

  <Header xmlns="http://www.w3.org/2003/05/soap-envelope">

    <Action u:shallUnderstand="1" xmlns:u="http://www.w3.org/2003/05/soap-envelope"
xmlns="http://www.w3.org/2005/08/addressing">http://docs.oasis-open.org/wsn/bw-
2/NotificationProducer/SubscribeRequest</Action>

    <MessageID xmlns="http://www.w3.org/2005/08/addressing">urn:uuid:9f2a12de-3a76-
461b-a421-e472517bcc7e</MessageID>

    <ReplyTo xmlns="http://www.w3.org/2005/08/addressing">

      <Address
xmlns="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing/a
nonymous</Address>

    </ReplyTo>

    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">

      <UsernameToken xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">

        <Username xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">user</Username>

        <Password xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" Type="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-
1.0#PasswordDigest">5zjIbmvWxVevGlpqg6Qnt9h8Fmo=</Password>

        <Nonce xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">ikcoiK+AmJvA5UpfxTzG8Q==</Nonce>

        <Created xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">2011-04-25T09:27:48Z</Created>

      </UsernameToken>

    </Security>

    <To t:shallUnderstand="1" xmlns:t="http://www.w3.org/2003/05/soap-envelope"
xmlns="http://www.w3.org/2005/08/addressing">http://169.254.141.200/onvif/services<
/To>

  </Header>

  <Body xmlns="http://www.w3.org/2003/05/soap-envelope">

    <Subscribe xmlns="http://docs.oasis-open.org/wsn/b-2">

      <ConsumerReference xmlns="http://docs.oasis-open.org/wsn/b-2">

        <Address
xmlns="http://www.w3.org/2005/08/addressing">http://192.168.10.66/onvif_notify_serv
er</Address>

      </ConsumerReference>

      <InitialTerminationTime>PT10S</InitialTerminationTime>
```

```
    </Subscribe>

  </Body>

</Envelope>
```

PROBE message example:

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">

  <Header xmlns="http://www.w3.org/2003/05/soap-envelope">

    <MessageID
xmlns="http://schemas.xmlsoap.org/ws/2004/08/addressing">uuid:9c35aca0-d1cc-41bf-
a932-2dd0fe45cc87</MessageID>

    <To xmlns="http://schemas.xmlsoap.org/ws/2004/08/addressing">urn:schemas-
xmlsoap-org:ws:2005:04:discovery</To>

    <Action
xmlns="http://schemas.xmlsoap.org/ws/2004/08/addressing">http://schemas.xmlsoap.org
/ws/2005/04/discovery/Probe</Action>

  </Header>

  <Body xmlns="http://www.w3.org/2003/05/soap-envelope">

    <Probe xmlns="http://schemas.xmlsoap.org/ws/2005/04/discovery">

      <Types xmlns="http://schemas.xmlsoap.org/ws/2005/04/discovery"
xmlns:dn="http://www.onvif.org/ver10/network/wsdl">dn:NetworkVideoTransmitter</Type
s>

      <Scopes
xmlns="http://schemas.xmlsoap.org/ws/2005/04/discovery">onvif://www.onvif.org/type
onvif://www.onvif.org/location onvif://www.onvif.org/hardware
onvif://www.onvif.org/name</Scopes>

    </Probe>

  </Body>

</Envelope>
```

2) Defaults Namespaces Definition in Parent Tag Examples

GetDNSRequest message example:

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">

  <Header>

    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">

      <UsernameToken>

        <Username>user</Username>
```

```
        <Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordDigest">5zjIbmvWxVevGlpqg6Qnt9h8Fmo=</Password>

        <Nonce>ikcoiK+AmJvA5UpfxTzG8Q==</Nonce>

        <Created xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">2011-04-25T09:27:48Z</Created>

      </UsernameToken>

    </Security>

  </Header>

  <Body>

    <GetDNS xmlns="http://www.onvif.org/ver10/device/wsdl" />

  </Body>

</Envelope>
```

SetDNSRequest message example:

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">

  <Header>

    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">

      <UsernameToken>

        <Username>service</Username>

        <Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordDigest">znYLkZuoGqC5RrFD4KDs529JvHI=</Password>

        <Nonce>7L0dq2/ZF3zWYEpQpFhcHA==</Nonce>

        <Created xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">2011-04-25T09:27:49Z</Created>

      </UsernameToken>

    </Security>

  </Header>

  <Body>

    <SetDNS xmlns="http://www.onvif.org/ver10/device/wsdl">

      <FromDHCP>false</FromDHCP>

      <DNSManual>

        <Type xmlns="http://www.onvif.org/ver10/schema">IPv4</Type>

        <IPv4Address
xmlns="http://www.onvif.org/ver10/schema">10.1.1.1</IPv4Address>

      </DNSManual>
```

```
    </SetDNS>

  </Body>

</Envelope>
```

GetCapabilitiesRequest message example:

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">

  <Header>

    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">

      <UsernameToken>

        <Username>service</Username>

        <Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordDigest">tg6EnHtyMWW8eUfntHO6XpPjOsg=</Password>

        <Nonce>iBIGpSuHtNPbdSWGzG48ng==</Nonce>

        <Created xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">2011-04-25T10:54:34Z</Created>

      </UsernameToken>

    </Security>

  </Header>

  <Body>

    <GetCapabilities xmlns="http://www.onvif.org/ver10/device/wsdl">

      <Category>Events</Category>

    </GetCapabilities>

  </Body>

</Envelope>
```

SubscribeRequest message example:

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">

  <Header>

    <Action u:shallUnderstand="1" xmlns:u="http://www.w3.org/2003/05/soap-envelope"
xmlns="http://www.w3.org/2005/08/addressing">http://docs.oasis-open.org/wsn/bw-
2/NotificationProducer/SubscribeRequest</Action>

    <MessageID xmlns="http://www.w3.org/2005/08/addressing">urn:uuid:9f2a12de-3a76-
461b-a421-e472517bcc7e</MessageID>

    <ReplyTo xmlns="http://www.w3.org/2005/08/addressing">
```

```
      <Address>http://www.w3.org/2005/08/addressing/anonymous</Address>

    </ReplyTo>

    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">

      <UsernameToken>

        <Username>user</Username>

        <Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordDigest">5zjIbmvWxVevGlpqg6Qnt9h8Fmo=</Password>

        <Nonce>ikcoiK+AmJvA5UpfxTzG8Q==</Nonce>

        <Created xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">2011-04-25T09:27:48Z</Created>

      </UsernameToken>

    </Security>

    <To t:shallUnderstand="1" xmlns:t="http://www.w3.org/2003/05/soap-envelope"
xmlns="http://www.w3.org/2005/08/addressing">http://169.254.141.200/onvif/services<
/To>

  </Header>

  <Body>

    <Subscribe xmlns="http://docs.oasis-open.org/wsn/b-2">

      <ConsumerReference xmlns="http://docs.oasis-open.org/wsn/b-2">

        <Address
xmlns="http://www.w3.org/2005/08/addressing">http://192.168.10.66/onvif_notify_serv
er</Address>

      </ConsumerReference>

      <InitialTerminationTime xmlns="http://docs.oasis-open.org/wsn/b-
2">PT10S</InitialTerminationTime>

    </Subscribe>

  </Body>

</Envelope>
```

PROBE message example:

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">

  <Header>

    <MessageID
xmlns="http://schemas.xmlsoap.org/ws/2004/08/addressing">uuid:9c35aca0-d1cc-41bf-
a932-2dd0fe45cc87</MessageID>

    <To xmlns="http://schemas.xmlsoap.org/ws/2004/08/addressing">urn:schemas-
xmlsoap-org:ws:2005:04:discovery</To>
```

```
    <Action
xmlns="http://schemas.xmlsoap.org/ws/2004/08/addressing">http://schemas.xmlsoap.org
/ws/2005/04/discovery/Probe</Action>

  </Header>

  <Body>

    <Probe xmlns="http://schemas.xmlsoap.org/ws/2005/04/discovery">

      <Types
xmlns:dn="http://www.onvif.org/ver10/network/wsdl">dn:NetworkVideoTransmitter</Type
s>

      <Scopes>onvif://www.onvif.org/type onvif://www.onvif.org/location
onvif://www.onvif.org/hardware onvif://www.onvif.org/name</Scopes>

    </Probe>

  </Body>

</Envelope>
```

3) Namespaces Definition with non-Standard Prefixes Examples

GetDNSRequest message example:

```
<prefix1:Envelope

  xmlns:prefix1="http://www.w3.org/2003/05/soap-envelope"

  xmlns:prefix2="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"

  xmlns:prefix3="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"

  xmlns:prefix4="http://www.onvif.org/ver10/device/wsdl">

  <prefix1:Header>

    <prefix2:Security>

      <prefix2:UsernameToken>

        <prefix2:Username>user</prefix2:Username>

        <prefix2:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-
1.0#PasswordDigest">5zjIbmvWxVevGlpqg6Qnt9h8Fmo=</prefix2:Password>

        <prefix2:Nonce>ikcoiK+AmJvA5UpfxTzG8Q==</prefix2:Nonce>

        <prefix3:Created>2011-04-25T09:27:48Z</prefix3:Created>

      </prefix2:UsernameToken>

    </prefix2:Security>

  </prefix1:Header>

  <prefix1:Body>
```

```
    <prefix4:GetDNS/>
  </prefix1:Body>
</prefix1:Envelope>
```

SetDNSRequest message example:

```
<prefix2:Envelope
  xmlns:prefix2="http://www.w3.org/2003/05/soap-envelope"
  xmlns:prefix1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"
  xmlns:prefix4="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"
  xmlns:prefix3="http://www.onvif.org/ver10/device/wsdl"
  xmlns:wsu="http://www.onvif.org/ver10/schema">
  <prefix2:Header>
    <prefix1:Security>
      <prefix1:UsernameToken>
        <prefix1:Username>service</prefix1:Username>
        <prefix1:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-
1.0#PasswordDigest">znYLkZuoGqC5RrFD4KDs529JvHI=</prefix1:Password>
        <prefix1:Nonce>7L0dq2/ZF3zWYEpQpFhcHA==</prefix1:Nonce>
        <prefix4:Created>2011-04-25T09:27:49Z</prefix4:Created>
      </prefix1:UsernameToken>
    </prefix1:Security>
  </prefix2:Header>
  <prefix2:Body>
    <prefix3:SetDNS>
      <prefix3:FromDHCP>false</prefix3:FromDHCP>
      <prefix3:DNSManual>
        <wsu:Type>IPv4</wsu:Type>
        <wsu:IPv4Address>10.1.1.1</wsu:IPv4Address>
      </prefix3:DNSManual>
    </prefix3:SetDNS>
  </prefix2:Body>
</prefix2:Envelope>
```

GetCapabilitiesRequest message example:

```
<prefix1:Envelope

  xmlns:prefix4="http://www.onvif.org/ver10/device/wsdl"

  xmlns:prefix3="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"

  xmlns:prefix1="http://www.w3.org/2003/05/soap-envelope"

  xmlns:prefix2="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">

  <prefix1:Header>

    <prefix2:Security>

      <prefix2:UsernameToken>

        <prefix2:Username>service</prefix2:Username>

        <prefix2:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-
1.0#PasswordDigest">tg6EnHtyMWW8eUfntHO6XpPjOsg=</prefix2:Password>

        <prefix2:Nonce>iBIGpSuHtNPbdSWGzG48ng==</prefix2:Nonce>

        <prefix3:Created>2011-04-25T10:54:34Z</prefix3:Created>

      </prefix2:UsernameToken>

    </prefix2:Security>

  </prefix1:Header>

  <prefix1:Body>

    <prefix4:GetCapabilities>

      <prefix4:Category>Events</prefix4:Category>

    </prefix4:GetCapabilities>

  </prefix1:Body>

</prefix1:Envelope>
```

SubscribeRequest message example:

```
<prefix0:Envelope

  xmlns:prefix0="http://www.w3.org/2003/05/soap-envelope"

  xmlns:prefix1="http://www.w3.org/2003/05/soap-envelope"

  xmlns:prefix2="http://www.w3.org/2005/08/addressing"

  xmlns:prefix3="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"
```

```
  xmlns:prefix4="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"

  xmlns:prefix5="http://docs.oasis-open.org/wsn/b-2"

  >

  <prefix0:Header>

    <prefix2:Action prefix1:shallUnderstand="1">http://docs.oasis-open.org/wsn/bw-
2/NotificationProducer/SubscribeRequest</prefix2:Action>

    <prefix2:MessageID>urn:uuid:9f2a12de-3a76-461b-a421-
e472517bcc7e</prefix2:MessageID>

    <prefix2:ReplyTo>


<prefix2:Address>http://www.w3.org/2005/08/addressing/anonymous</prefix2:Address>

    </prefix2:ReplyTo>

    <prefix4:Security>

      <prefix4:UsernameToken>

        <prefix4:Username>service</prefix4:Username>

        <prefix4:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-
1.0#PasswordDigest">tg6EnHtyMWW8eUfntHO6XpPjOsg=</prefix4:Password>

        <prefix4:Nonce>iBIGpSuHtNPbdSWGzG48ng==</prefix4:Nonce>

        <prefix3:Created>2011-04-25T10:54:34Z</prefix3:Created>

      </prefix4:UsernameToken>

    </prefix4:Security>

    <prefix2:To
prefix1:shallUnderstand="1">http://169.254.141.200/onvif/services</prefix2:To>

  </prefix0:Header>

  <prefix0:Body>

    <prefix5:Subscribe>

      <prefix5:ConsumerReference>

        <prefix2:Address
xmlns="http://www.w3.org/2005/08/addressing">http://192.168.10.66/onvif_notify_serv
er</prefix2:Address>

      </prefix5:ConsumerReference>

      <prefix5:InitialTerminationTime>PT10S</prefix5:InitialTerminationTime>

    </prefix5:Subscribe>

  </prefix0:Body>

</prefix0:Envelope>
```

PROBE message example:

```
<prefix2:Envelope

  xmlns:prefix1="http://www.onvif.org/ver10/network/wsdl"

  xmlns:prefix2="http://www.w3.org/2003/05/soap-envelope"

  xmlns:prefix3="http://schemas.xmlsoap.org/ws/2004/08/addressing"

  xmlns:prefix4="http://schemas.xmlsoap.org/ws/2005/04/discovery">

  <prefix2:Header>

    <prefix3:MessageID>uuid:9c35aca0-d1cc-41bf-a932-
2dd0fe45cc87</prefix3:MessageID>

    <prefix3:To>urn:schemas-xmlsoap-org:ws:2005:04:discovery</prefix3:To>


<prefix3:Action>http://schemas.xmlsoap.org/ws/2005/04/discovery/Probe</prefix3:Acti
on>

  </prefix2:Header>

  <prefix2:Body>

    <prefix4:Probe>

      <prefix4:Types>prefix1:NetworkVideoTransmitter</prefix4:Types>

      <prefix4:Scopes>onvif://www.onvif.org/type onvif://www.onvif.org/location
onvif://www.onvif.org/hardware onvif://www.onvif.org/name</prefix4:Scopes>

    </prefix4:Probe>

  </prefix2:Body>

</prefix2:Envelope>
```

4) Namespaces Definition with Different Prefixes for the Same Namespace Examples

GetDNSRequest message example:

```
<p1:Envelope xmlns:p1="http://www.w3.org/2003/05/soap-envelope">

  <p2:Header xmlns:p2="http://www.w3.org/2003/05/soap-envelope">

    <p3:Security xmlns:p3="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">

      <p4:UsernameToken xmlns:p4="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd">

        <p5:Username xmlns:p5="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">user</p5:Username>

        <p6:Password xmlns:p6="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" Type="http://docs.oasis-open.org/wss/2004/01/oasis-
```

```
200401-wss-username-token-profile-
1.0#PasswordDigest">5zjIbmvWxVevGlpqg6Qnt9h8Fmo=</p6:Password>

        <p7:Nonce xmlns:p7="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">ikcoiK+AmJvA5UpfxTzG8Q==</p7:Nonce>

        <p8:Created xmlns:p8="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">2011-04-25T09:27:48Z</p8:Created>

      </p4:UsernameToken>

    </p3:Security>

  </p2:Header>

  <p9:Body xmlns:p9="http://www.w3.org/2003/05/soap-envelope">

    <p10:GetDNS xmlns:p10="http://www.onvif.org/ver10/device/wsdl" />

  </p9:Body>

</p1:Envelope>
```

SetDNSRequest message example:

```
<q1:Envelope xmlns:q1="http://www.w3.org/2003/05/soap-envelope">

  <q2:Header xmlns:q2="http://www.w3.org/2003/05/soap-envelope">

    <q3:Security xmlns:q3="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">

      <q4:UsernameToken xmlns:q4="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd">

        <q3:Username>service</q3:Username>

        <q4:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-
1.0#PasswordDigest">znYLkZuoGqC5RrFD4KDs529JvHI=</q4:Password>

        <q3:Nonce>7L0dq2/ZF3zWYEpQpFhcHA==</q3:Nonce>

        <q5:Created xmlns:q5="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">2011-04-25T09:27:49Z</q5:Created>

      </q4:UsernameToken>

    </q3:Security>

  </q2:Header>

  <q1:Body>

    <q6:SetDNS xmlns:q6="http://www.onvif.org/ver10/device/wsdl">

      <q7:FromDHCP
xmlns:q7="http://www.onvif.org/ver10/device/wsdl">false</q7:FromDHCP>

      <q6:DNSManual>

        <q8:Type xmlns:q8="http://www.onvif.org/ver10/schema">IPv4</q8:Type>
```

```
        <q9:IPv4Address
xmlns:q9="http://www.onvif.org/ver10/schema">10.1.1.1</q9:IPv4Address>

      </q6:DNSManual>

    </q6:SetDNS>

  </q1:Body>

</q1:Envelope>
```

GetCapabilitiesRequest message example:

```
<p1:Envelope xmlns:p1="http://www.w3.org/2003/05/soap-envelope">

  <p2:Header xmlns:p2="http://www.w3.org/2003/05/soap-envelope">

    <p4:Security xmlns:p4="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">

      <p5:UsernameToken xmlns:p5="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd">

        <p6:Username xmlns:p6="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">service</p6:Username>

        <p7:Password xmlns:p7="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" Type="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-
1.0#PasswordDigest">tg6EnHtyMWW8eUfntHO6XpPjOsg=</p7:Password>

        <p8:Nonce xmlns:p8="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">iBIGpSuHtNPbdSWGzG48ng==</p8:Nonce>

        <p9:Created xmlns:p9="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">2011-04-25T10:54:34Z</p9:Created>

      </p5:UsernameToken>

    </p4:Security>

  </p2:Header>

  <p3:Body xmlns:p3="http://www.w3.org/2003/05/soap-envelope">

    <p10:GetCapabilities xmlns:p10="http://www.onvif.org/ver10/device/wsdl">

      <p11:Category
xmlns:p11="http://www.onvif.org/ver10/device/wsdl">Events</p11:Category>

    </p10:GetCapabilities>

  </p3:Body>

</p1:Envelope>
```

SubscribeRequest message example:

```
<e1:Envelope xmlns:e1="http://www.w3.org/2003/05/soap-envelope">
```

```
  <e2:Header xmlns:e2="http://www.w3.org/2003/05/soap-envelope">

    <e3:Action u:shallUnderstand="1" xmlns:u="http://www.w3.org/2003/05/soap-
envelope" xmlns:e3="http://www.w3.org/2005/08/addressing">http://docs.oasis-
open.org/wsn/bw-2/NotificationProducer/SubscribeRequest</e3:Action>

    <e4:MessageID
xmlns:e4="http://www.w3.org/2005/08/addressing">urn:uuid:9f2a12de-3a76-461b-a421-
e472517bcc7e</e4:MessageID>

    <e5:ReplyTo xmlns:e5="http://www.w3.org/2005/08/addressing">

      <e6:Address
xmlns:e6="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressin
g/anonymous</e6:Address>

    </e5:ReplyTo>

    <e7:Security xmlns:e7="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">

      <e8:UsernameToken xmlns:e8="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd">

        <e9:Username xmlns:e9="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">user</e9:Username>

        <e10:Password xmlns:e10="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" Type="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-username-token-profile-
1.0#PasswordDigest">5zjIbmvWxVevGlpqg6Qnt9h8Fmo=</e10:Password>

        <e11:Nonce xmlns:e11="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">ikcoiK+AmJvA5UpfxTzG8Q==</e11:Nonce>

        <e12:Created xmlns:e12="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd">2011-04-25T09:27:48Z</e12:Created>

      </e8:UsernameToken>

    </e7:Security>

    <e13:To t:shallUnderstand="1" xmlns:t="http://www.w3.org/2003/05/soap-envelope"
xmlns:e13="http://www.w3.org/2005/08/addressing">http://169.254.141.200/onvif/servi
ces</e13:To>

  </e2:Header>

  <e1:Body>

    <e14:Subscribe xmlns:e14="http://docs.oasis-open.org/wsn/b-2">

      <e15:ConsumerReference xmlns:e15="http://docs.oasis-open.org/wsn/b-2">

        <e16:Address
xmlns:e16="http://www.w3.org/2005/08/addressing">http://192.168.10.66/onvif_notify_
server</e16:Address>

      </e15:ConsumerReference>

      <e14:InitialTerminationTime>PT10S</e14:InitialTerminationTime>

    </e14:Subscribe>

  </e1:Body>
```

```
</e1:Envelope>
```

PROBE message example:

```
<p1:Envelope xmlns:p1="http://www.w3.org/2003/05/soap-envelope">

  <p2:Header xmlns:p2="http://www.w3.org/2003/05/soap-envelope">

    <p3:MessageID
xmlns:p3="http://schemas.xmlsoap.org/ws/2004/08/addressing">uuid:9c35aca0-d1cc-
41bf-a932-2dd0fe45cc87</p3:MessageID>

    <p4:To xmlns:p4="http://schemas.xmlsoap.org/ws/2004/08/addressing">urn:schemas-
xmlsoap-org:ws:2005:04:discovery</p4:To>

    <p5:Action
xmlns:p5="http://schemas.xmlsoap.org/ws/2004/08/addressing">http://schemas.xmlsoap.
org/ws/2005/04/discovery/Probe</p5:Action>

  </p2:Header>

  <p6:Body xmlns:p6="http://www.w3.org/2003/05/soap-envelope">

    <p7:Probe xmlns:p7="http://schemas.xmlsoap.org/ws/2005/04/discovery">

      <p8:Types xmlns:p8="http://schemas.xmlsoap.org/ws/2005/04/discovery"
xmlns:dn="http://www.onvif.org/ver10/network/wsdl">dn:NetworkVideoTransmitter</p8:T
ypes>

      <p9:Scopes
xmlns:p9="http://schemas.xmlsoap.org/ws/2005/04/discovery">onvif://www.onvif.org/ty
pe onvif://www.onvif.org/location onvif://www.onvif.org/hardware
onvif://www.onvif.org/name</p9:Scopes>

    </p7:Probe>

  </p6:Body>

</p1:Envelope>
```

5) Namespaces Definition with the Same Prefixes for Different Namespaces Examples

GetDNSRequest message example:

```
<p1:Envelope xmlns:p1="http://www.w3.org/2003/05/soap-envelope">

  <p1:Header>

    <p1:Security xmlns:p1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">

      <p1:UsernameToken>

        <p1:Username>user</p1:Username>

        <p1:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-
1.0#PasswordDigest">5zjIbmvWxVevGlpqg6Qnt9h8Fmo=</p1:Password>

        <p1:Nonce>ikcoiK+AmJvA5UpfxTzG8Q==</p1:Nonce>
```

```
        <p1:Created xmlns:p1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">2011-04-25T09:27:48Z</p1:Created>

      </p1:UsernameToken>

    </p1:Security>

  </p1:Header>

  <p1:Body>

    <p1:GetDNS xmlns:p1="http://www.onvif.org/ver10/device/wsdl" />

  </p1:Body>

</p1:Envelope>
```

SetDNSRequest message example:

```
<q1:Envelope xmlns:q1="http://www.w3.org/2003/05/soap-envelope">

  <q1:Header>

    <q1:Security xmlns:q1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">

      <q1:UsernameToken>

        <q1:Username>service</q1:Username>

        <q1:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-
1.0#PasswordDigest">znYLkZuoGqC5RrFD4KDs529JvHI=</q1:Password>

        <q1:Nonce>7L0dq2/ZF3zWYEpQpFhcHA==</q1:Nonce>

        <q1:Created xmlns:q1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">2011-04-25T09:27:49Z</q1:Created>

      </q1:UsernameToken>

    </q1:Security>

  </q1:Header>

  <q1:Body>

    <q1:SetDNS xmlns:q1="http://www.onvif.org/ver10/device/wsdl">

      <q1:FromDHCP>false</q1:FromDHCP>

      <q1:DNSManual>

        <q1:Type xmlns:q1="http://www.onvif.org/ver10/schema">IPv4</q1:Type>

        <q1:IPv4Address
xmlns:q1="http://www.onvif.org/ver10/schema">10.1.1.1</q1:IPv4Address>

      </q1:DNSManual>

    </q1:SetDNS>

  </q1:Body>
```

```
</q1:Envelope>
```

GetCapabilitiesRequest message example:

```
<p1:Envelope xmlns:p1="http://www.w3.org/2003/05/soap-envelope">

  <p1:Header>

    <p1:Security xmlns:p1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">

      <p1:UsernameToken>

        <p1:Username>service</p1:Username>

        <p1:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-
1.0#PasswordDigest">tg6EnHtyMWW8eUfntHO6XpPjOsg=</p1:Password>

        <p1:Nonce>iBIGpSuHtNPbdSWGzG48ng==</p1:Nonce>

        <p1:Created xmlns:p1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">2011-04-25T10:54:34Z</p1:Created>

      </p1:UsernameToken>

    </p1:Security>

  </p1:Header>

  <p1:Body>

    <p1:GetCapabilities xmlns:p1="http://www.onvif.org/ver10/device/wsdl">

      <p1:Category>Events</p1:Category>

    </p1:GetCapabilities>

  </p1:Body>

</p1:Envelope>
```

SubscribeRequest message example:

```
<p1:Envelope xmlns:p1="http://www.w3.org/2003/05/soap-envelope">

  <p1:Header>

    <p1:Action u:shallUnderstand="1" xmlns:u="http://www.w3.org/2003/05/soap-
envelope" xmlns:p1="http://www.w3.org/2005/08/addressing">http://docs.oasis-
open.org/wsn/bw-2/NotificationProducer/SubscribeRequest</p1:Action>

    <p1:MessageID
xmlns:p1="http://www.w3.org/2005/08/addressing">urn:uuid:9f2a12de-3a76-461b-a421-
e472517bcc7e</p1:MessageID>

    <p1:ReplyTo xmlns:p1="http://www.w3.org/2005/08/addressing">

      <p1:Address>http://www.w3.org/2005/08/addressing/anonymous</p1:Address>
```

```
    </p1:ReplyTo>

    <p1:Security xmlns:p1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">

      <p1:UsernameToken>

        <p1:Username>user</p1:Username>

        <p1:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-
1.0#PasswordDigest">5zjIbmvWxVevGlpqg6Qnt9h8Fmo=</p1:Password>

        <p1:Nonce>ikcoiK+AmJvA5UpfxTzG8Q==</p1:Nonce>

        <p1:Created xmlns:p1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">2011-04-25T09:27:48Z</p1:Created>

      </p1:UsernameToken>

    </p1:Security>

    <p1:To t:shallUnderstand="1" xmlns:t="http://www.w3.org/2003/05/soap-envelope"
xmlns:p1="http://www.w3.org/2005/08/addressing">http://169.254.141.200/onvif/servic
es</p1:To>

  </p1:Header>

  <p1:Body>

    <p1:Subscribe xmlns:p1="http://docs.oasis-open.org/wsn/b-2">

      <p1:ConsumerReference xmlns:p1="http://docs.oasis-open.org/wsn/b-2">

        <p1:Address
xmlns:p1="http://www.w3.org/2005/08/addressing">http://192.168.10.66/onvif_notify_s
erver</p1:Address>

      </p1:ConsumerReference>

      <p1:InitialTerminationTime xmlns:p1="http://docs.oasis-open.org/wsn/b-
2">PT10S</p1:InitialTerminationTime>

    </p1:Subscribe>

  </p1:Body>

</p1:Envelope>
```

PROBE message example:

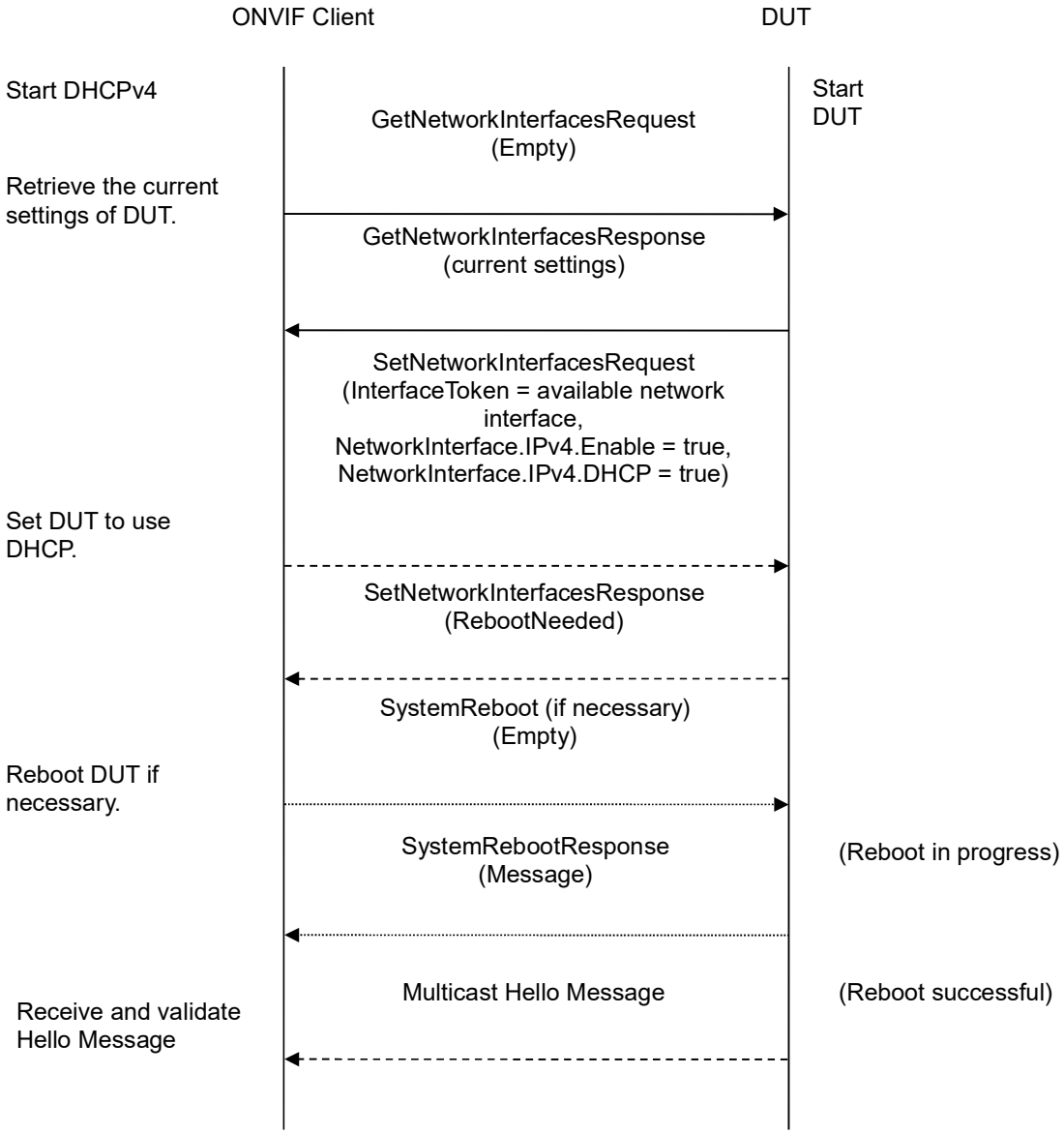
```
<p1:Envelope xmlns:p1="http://www.w3.org/2003/05/soap-envelope">

  <p1:Header xmlns:p1="http://www.w3.org/2003/05/soap-envelope">

    <p1:MessageID
xmlns:p1="http://schemas.xmlsoap.org/ws/2004/08/addressing">uuid:9c35aca0-d1cc-
41bf-a932-2dd0fe45cc87</p1:MessageID>

    <p1:To xmlns:p1="http://schemas.xmlsoap.org/ws/2004/08/addressing">urn:schemas-
xmlsoap-org:ws:2005:04:discovery</p1:To>
```

```
    <p1:Action
xmlns:p1="http://schemas.xmlsoap.org/ws/2004/08/addressing">http://schemas.xmlsoap.
org/ws/2005/04/discovery/Probe</p1:Action>

  </p1:Header>

  <p1:Body xmlns:p1="http://www.w3.org/2003/05/soap-envelope">

    <p1:Probe xmlns:p1="http://schemas.xmlsoap.org/ws/2005/04/discovery">

      <p1:Types xmlns:p1="http://schemas.xmlsoap.org/ws/2005/04/discovery"
xmlns:dn="http://www.onvif.org/ver10/network/wsdl">dn:NetworkVideoTransmitter</p1:T
ypes>

      <p1:Scopes
xmlns:p1="http://schemas.xmlsoap.org/ws/2005/04/discovery">onvif://www.onvif.org/ty
pe onvif://www.onvif.org/location onvif://www.onvif.org/hardware
onvif://www.onvif.org/name</p1:Scopes>

    </p1:Probe>

  </p1:Body>

</p1:Envelope>
```

A.12 Procedure to turn on IPv4 DHCP

Certain test cases require turning on IPv4 DHCP configuration if DUT is configured as manual IPv4 configuration. The following is the procedure to turn on IPv4 DHCP.

1. ONVIF Client will invoke GetNetworkInterfacesRequest message to retrieve the current settings of the DUT.
2. If NetworkInterface.IPv4.DHCP == false in the current settings, ONVIF Client will invoke SetNetworkInterfacesRequest message to set DUT to use DHCP (InterfaceToken = available network interface, NetworkInterface.IPv4.Enabled = true, NetworkInterface.IPv4.DHCP = true). Otherwise, skip other steps of Annex and continue test case execution.
3. If necessary, ONVIF Client will invoke SystemReboot message to restart DUT.
4. DUT will send Multicast Hello message from newly configured address.
5. ONVIF Client will receive and validate Hello message sent from newly configured address by DUT.

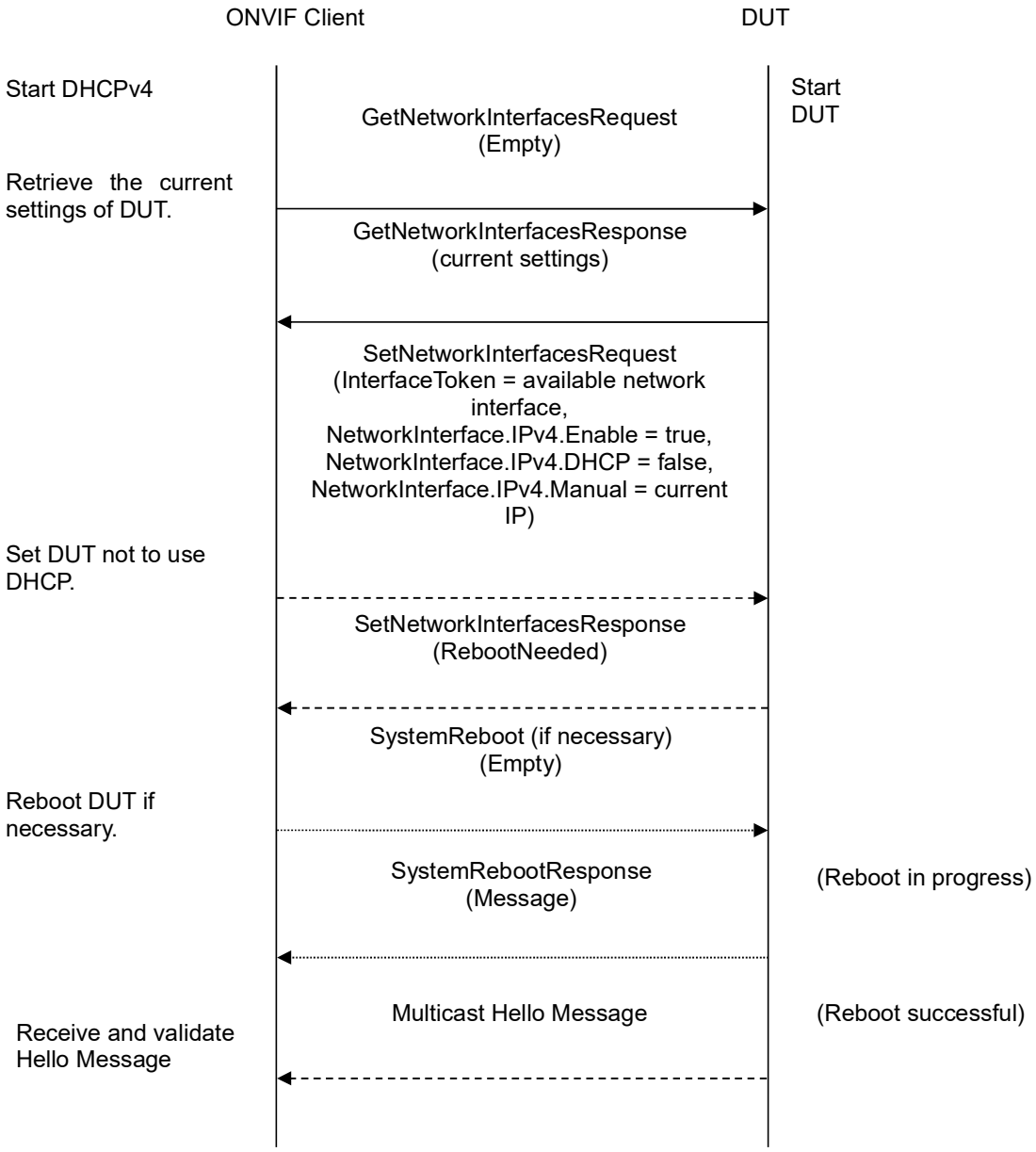


A.13 Procedure to turn off IPv4 DHCP

Certain test cases require turning off IPv4 DHCP configuration if DUT is configured as DHCP IPv4 configuration. The following is the procedure to turn off IPv4 DHCP.

1. ONVIF Client will invoke GetNetworkInterfacesRequest message to retrieve the current settings of the DUT.
2. If NetworkInterface.IPv4.DHCP == true in the current settings, ONVIF Client will invoke SetNetworkInterfacesRequest message to set the DUT not to use DHCP (InterfaceToken = available network interface, NetworkInterface.IPv4.Enabled = true, NetworkInterface.IPv4.DHCP = false, NetworkInterface.IPv4.Manual = current IP). Otherwise, skip other steps of Annex and continue test case execution.

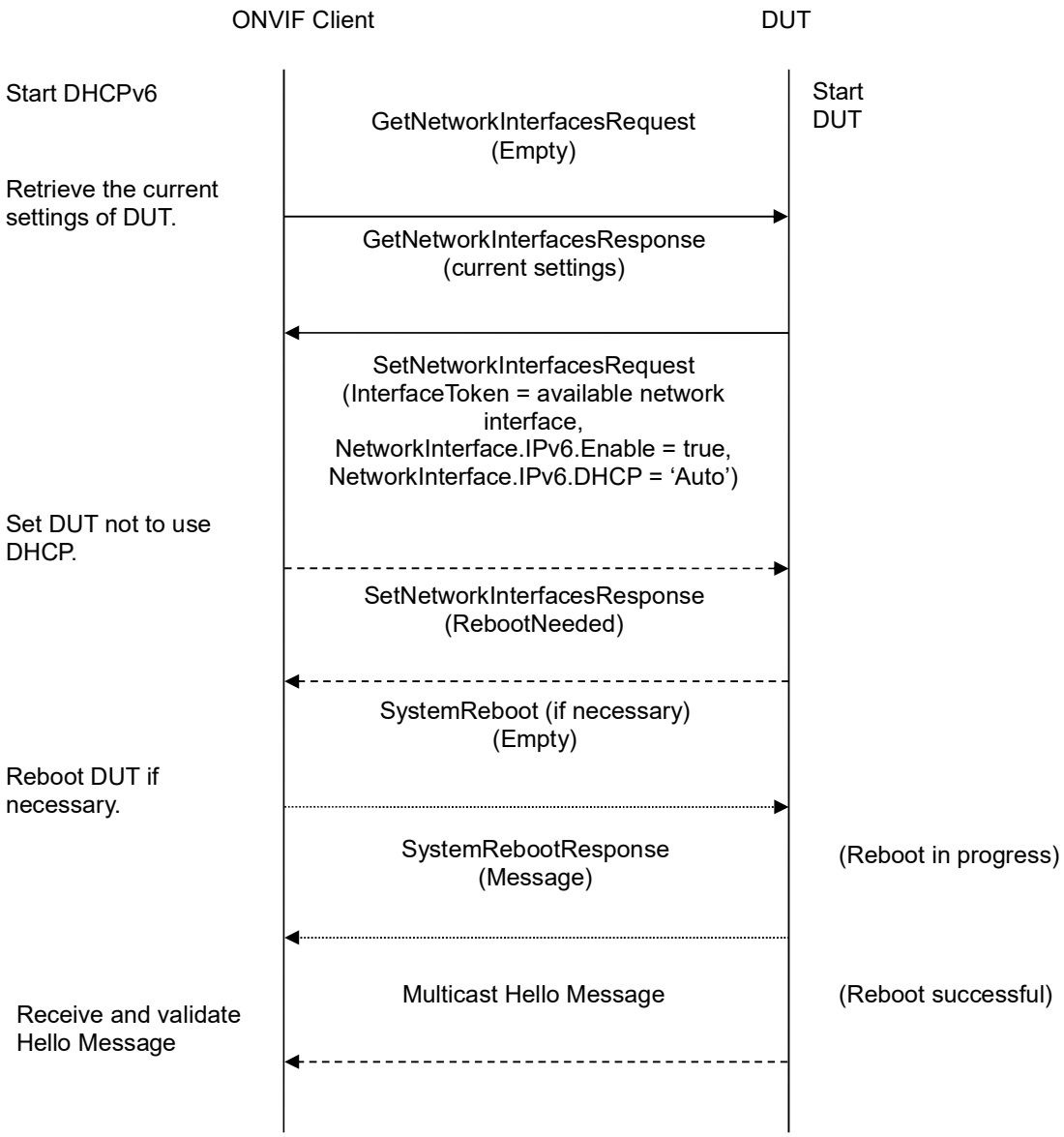
3. If necessary, ONVIF Client will invoke SystemReboot message to restart DUT.
4. DUT will send Multicast Hello message from a newly configured address.
5. ONVIF Client will receive and validate Hello message sent from a newly configured address by the DUT.



A.14 Procedure to turn on IPv6 DHCP

Certain test cases require turning on IPv6 DHCP configuration if DUT IPv6 network is configured differently. The following is the procedure to turn on IPv6 DHCP.

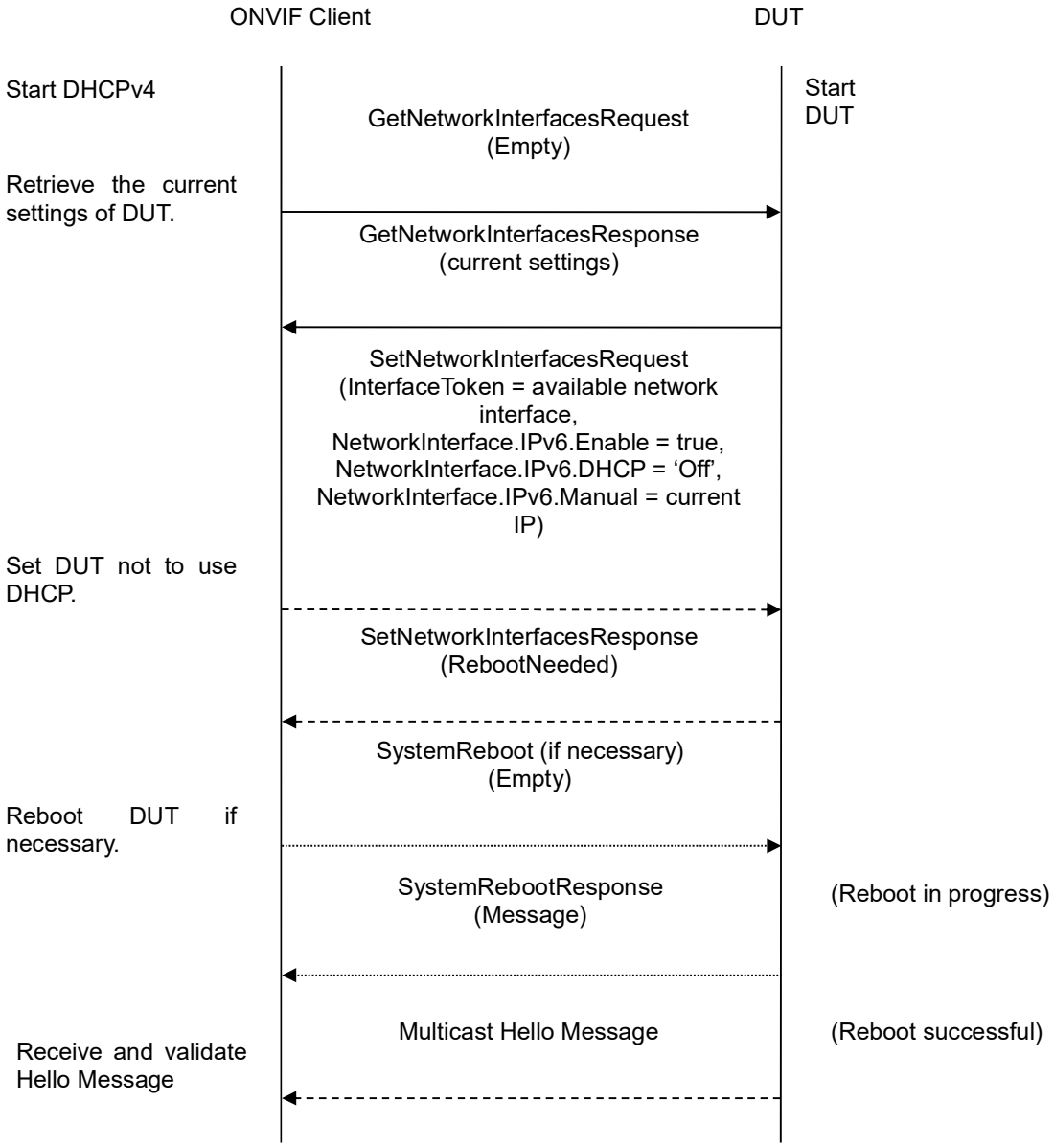
1. ONVIF Client will invoke GetNetworkInterfacesRequest message to retrieve the current settings of the DUT.
2. If NetworkInterface.IPv6.DHCP == 'Off' in the current settings, ONVIF Client will invoke SetNetworkInterfacesRequest message to set DUT to use DHCP (InterfaceToken = available network interface, NetworkInterface.IPv6.Enabled = true, NetworkInterface.IPv6.DHCP = 'Auto'). Otherwise, skip other steps of Annex and continue test case execution.
3. If necessary, ONVIF Client will invoke SystemReboot message to restart the DUT.
4. DUT will send Multicast Hello message from a newly configured address.
5. ONVIF Client will receive and validate Hello message sent from a newly configured address by DUT.



A.15 Procedure to turn off IPv6 DHCP

Certain test cases require turning off IPv6 DHCP configuration if DUT is configured as IPv6 DHCP. The following is the procedure to turn off IPv6 DHCP.

1. ONVIF Client will invoke GetNetworkInterfacesRequest message to retrieve the current settings of the DUT.
2. If NetworkInterface.IPv6.DHCP != 'Off' and IPv6.FromDHCP.Address is set in the current settings, ONVIF Client will invoke SetNetworkInterfacesRequest message to set DUT not to use DHCP (InterfaceToken = available network interface, NetworkInterface.IPv6.Enabled = true, NetworkInterface.IPv6.DHCP = 'Off', NetworkInterface.IPv6.Manual = current IP). Else if NetworkInterface.IPv6.DHCP != 'Off' and IPv6.FromDHCP.Address is not set in the current settings, ONVIF Client will invoke SetNetworkInterfacesRequest message to set DUT not to use DHCP (InterfaceToken = available network interface, NetworkInterface.IPv6.Enabled = true, NetworkInterface.IPv6.DHCP = 'Off'). Otherwise, skip other steps and continue test execution.
3. If necessary, ONVIF Client will invoke SystemReboot message to restart the DUT.
4. DUT will send Multicast Hello message from a newly configured address.
5. ONVIF Client will receive and validate Hello message sent from a newly configured address by the DUT.



A.16 Details on consistency checking for GetCapabilitiesResponse and GetServiceCapabilitiesResponse in Device Management Service

The following table shows some details on how the consistency checking will be performed in capabilities and device service capabilities consistency test case.

Table 1 Details on consistency checking for GetCapabilitiesResponse and GetServiceCapabilitiesResponse in Device Management Service

ID	GetCapabilitiesResponse.Capabilities. Device	GetServiceCapabilitiesResponse. Capabilities
1	Network	
1.1	(IPFilter == false) or (IPFilter is not included in response)	(IPFilter == false) or (IPFilter is not included in response)
1.2	(IPFilter == true)	(IPFilter == true)
1.3	(ZeroConfiguration == false) or (ZeroConfiguration is not included in response)	(ZeroConfiguration == false) or (ZeroConfiguration is not included in response)
1.4	(ZeroConfiguration == true)	(ZeroConfiguration == true)
1.5	(IPVersion6== false) or (IPVersion6 is not included in response)	(IPVersion6 == false) or (IPVersion6 is not included in response)
1.6	(IPVersion6 == true)	(IPVersion6 == true)
1.7	(DynDNS == false) or (DynDNS is not included in response)	(DynDNS == false) or (DynDNS is not in response)
1.8	(DynDNS == true)	(DynDNS == true)
1.9	(Extention.Dot11Configuration == false) or (Extention.Dot11Configuration is not included in response)	(Dot11Configuration == false) or (Dot11Configuration is not included in response)
1.10	(Extention.Dot11Configuration == true)	(Dot11Configuration == true)
2	System	
2.1	(DiscoveryResolve == false) or (DiscoveryResolve is not included in response)	(DiscoveryResolve == false) or (DiscoveryResolve is not included in response)
2.2	(Capabilities.Device.System.DiscoveryResolve == true)	(DiscoveryResolve == true)
2.3	(DiscoveryBye == false) or (DiscoveryBye is not included in response)	(DiscoveryBye == false) or (DiscoveryBye is not included in response)
2.4	(DiscoveryBye == true)	(DiscoveryBye == true)
2.5	(RemoteDiscovery == false) or	(RemoteDiscovery == false) or

	(RemoteDiscovery is not included in response)	(RemoteDiscovery is not included in response)
2.6	(RemoteDiscovery == true)	(RemoteDiscovery == true)
2.7	(SystemBackup == false) or (SystemBackup is not included in response)	(SystemBackup == false) or (SystemBackup is not included in response)
2.8	(SystemBackup == true)	(SystemBackup == true)
2.9	(SystemLogging == false) or (SystemLogging is not included in response)	(SystemLogging == false) or (SystemLogging is not included in response)
2.10	(SystemLogging == true)	(SystemLogging == true)
2.11	(FirmwareUpgrade == false) or (FirmwareUpgrade is not included in response)	(FirmwareUpgrade == false) or (FirmwareUpgrade is not included in response)
2.12	(FirmwareUpgrade == true)	(FirmwareUpgrade == true)
2.13	(Extension.HttpSystemBackup == false) or (Extension.HttpSystemBackup is not included in response)	(HttpSystemBackup == false) or (HttpSystemBackup is not included in response)
2.14	(Capabilities.Device.System.Extension.HttpSystemBackup == true)	(Capabilities.System.HttpSystemBackup == true)
2.15	(Extension.HttpSystemLogging == false) or (Extension.HttpSystemLogging is not included in response)	(HttpSystemLogging == false) or (HttpSystemLogging is not included in response)
2.16	(Extension.HttpSystemLogging == true)	(HttpSystemLogging == true)
2.17	(Extension.HttpFirmwareUpgrade == false) or (Extension.HttpFirmwareUpgrade is not included in response)	(HttpFirmwareUpgrade == false) or (HttpFirmwareUpgrade is not included in response)
2.18	(Extension.HttpFirmwareUpgrade == true)	(HttpFirmwareUpgrade == true)
3	Security	
3.1	(TLS1.1 == false) or (TLS1.1 is not included in response)	(TLS1.1 == false) or (TLS1.1 is not included in response)
3.2	(TLS1.1 == true)	(TLS1.2 == true)
3.3	(TLS1.2 == false) or	(TLS1.2 == false) or

	(TLS1.2 is not included in response)	(TLS1.2 is not included in response)
3.4	(TLS1.2 == true)	(TLS1.2 == true)
3.5	(OnBoardKeyGeneration == false) or (OnBoardKeyGeneration is not included in response)	(OnBoardKeyGeneration == false) or (OnBoardKeyGeneration is not included in response)
3.6	(OnBoardKeyGeneration == true)	(OnBoardKeyGeneration == true)
3.7	(AccessPolicyconfig == false) or (AccessPolicyconfig is not included in response)	(AccessPolicyconfig == false) or (AccessPolicyconfig is not included in response)
3.8	(AccessPolicyconfig == true)	(AccessPolicyconfig == true)
3.9	(X.509Token == false) or (X.509Token is not included in response)	(X.509Token == false) or (X.509Token is not included in response)
3.10	(X.509Token == true)	(X.509Token == true)
3.11	(SAMLToken == false) or (SAMLToken is not included in response)	(SAMLToken == false) or (SAMLToken is not included in response)
3.12	(SAMLToken == true)	(SAMLToken == true)
3.13	(KerberosToken == false) or (KerberosToken is not included in response)	(KerberosToken == false) or (KerberosToken is not included in response)
3.14	(KerberosToken == true)	(KerberosToken == true)
3.15	(RELToken == false) or (RELToken is not included in response)	(RELToken == false) or (RELToken is not included in response)
3.16	(RELToken == true)	(RELToken == true)
3.17	(Extention.TLS1.0 == false) or (Extention.TLS1.0 is not included in response)	(TLS1.0== false) or (TLS1.0 is not included in response)
3.18	(Extention.TLS1.0 == true)	(TLS1.0== true)
3.19	(Extention.Extention.Dot1X == false) or (Extention.Extention.Dot1X is not included in response)	(Dot1X == false) or (Dot1X is not included in response)
3.20	(Extention.Extention.Dot1X == true)	(Dot1X == true)
3.21	(Extention.Extention.SupportedEAPMethod == false) or (Extention.Extention.SupportedEAPMethod is	(SupportedEAPMethod == false) or (SupportedEAPMethod is not included in

	not included in response)	response)
3.22	(Extention.Extention.SupportedEAPMethod == true)	(SupportedEAPMethod == true)
3.23	(Extention.Extention.RemoteUserHandling == false) or (Extention.Extention.RemoteUserHandling is not included in response)	(RemoteUserHandling == false) or (RemoteUserHandling is not included in response)
3.24	(Extention.Extention.RemoteUserHandling == true)	(RemoteUserHandling == true)

A.17 Action URI's for Event Service Messages

The following Action URI's shall be used for Event Service:

Table 2 Action URI's for Event Service Messages

Message	Action URI of WS-Addressing
Notify	http://docs.oasis-open.org/wsn/bw-2/NotificationConsumer/Notify
SubscribeRequest	http://docs.oasis-open.org/wsn/bw-2/NotificationProducer/SubscribeRequest
SubscribeResponse	http://docs.oasis-open.org/wsn/bw-2/NotificationProducer/SubscribeResponse
RenewRequest	http://docs.oasis-open.org/wsn/bw-2/SubscriptionManager/RenewRequest
RenewResponse	http://docs.oasis-open.org/wsn/bw-2/SubscriptionManager/RenewResponse
UnsubscribeRequest	http://docs.oasis-open.org/wsn/bw-2/SubscriptionManager/UnsubscribeRequest
UnsubscribeResponse	http://docs.oasis-open.org/wsn/bw-2/SubscriptionManager/UnsubscribeResponse
GetEventPropertiesRequest	http://www.onvif.org/ver10/events/wsdl/EventPortType/GetEventPropertiesRequest
GetEventPropertiesResponse	http://www.onvif.org/ver10/events/wsdl/EventPortType/GetEventPropertiesResponse
CreatePullPointSubscriptionRequest	http://www.onvif.org/ver10/events/wsdl/EventPortType/CreatePullPointSubscriptionRequest
CreatePullPointSubscriptionResponse	http://www.onvif.org/ver10/events/wsdl/EventPortType/CreatePullPointSubscriptionResponse
PullMessagesRequest	http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/PullMessagesRequest

PullMessagesResponse	http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/PullMessagesResponse
SetSynchronizationPointRequest	http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/SetSynchronizationPointRequest
SetSynchronizationPointResponse	http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/SetSynchronizationPointResponse
GetServiceCapabilitiesResponse	http://www.onvif.org/ver10/events/wsdl/EventPortType/GetServiceCapabilities
GetServiceCapabilitiesRequest	http://www.onvif.org/ver10/events/wsdl/EventPortType/GetServiceCapabilitiesRequest
SeekRequest	http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/SeekRequest
SeekResponse	http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/SeekResponse
All faults	http://www.w3.org/2005/08/addressing/soap/fault

A.18 Name and Token Parameters Maximum Length

There are the following limitations on maximum length of Name and Token parameters that shall be used during tests by ONVIF Device Test Tool to prevent faults from the DUT:

- Name shall be less than or equal to 64 characters (only readable characters are accepted).
- Token shall be less than or equal to 64 characters (only readable characters are accepted).

UTF-8 character set shall be used for Name and Token.

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

A.19 Find begin of buffer time

The following algorithm will be used to get a current begin of buffer time:

1. ONVIF Client will invoke CreatePullPointSubscription message with TopicFilter = tns1:EventBuffer/Begin.
2. Verify that the DUT sends a CreatePullPointSubscriptionResponse. Validate that correct values for CurrentTime and TerminationTime and SubscriptionReference are returned.
3. ONVIF Client will invoke Seek message (UtcTime = [Current UTC Time], Reverse = true) to start a reverse seek.
4. Verify that the DUT sends a SeekResponse.
5. ONVIF Client will invoke PullMessages command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
6. Verify that the DUT sends a PullMessagesResponse that contains one NotificationMessage.

7. Verify that the received event is event with Topic = tns1:EventBuffer/Begin.

Message.UtcTime from Notification message with Topic = tns1:EventBuffer/Begin will be used as BeginOfBufferTime1.

Note: if PullMessagesResponse will be empty or there will be no Notification message with Topic = tns1:EventBuffer/Begin, test cases shall be failed.

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

A.20 TooManyUsers fault check

The following algorithm will be used to check if the DUT correctly returns TooManyUsers fault to CreateUsers request:

1. If DUT does not support GetServices, then ONVIF Client assumes test as PASSED and skips other steps
2. If the DUT supports GetServices, ONVIF Client invokes GetServiceCapabilitiesRequest to retrieve MaxUsers capability from the DUT.
3. Verify GetServiceCapabilitiesResponse from the DUT. Check MaxUsers capability.
4. If the DUT does not return MaxUsers capability, then ONVIF Client assumes test as PASSED and skips other steps.
5. If number of existing users is equal or more than MaxUsers capabilities, then ONVIF Client assumes test as PASSED. Otherwice ONVIF Client assumes test as FAILED.

A.21 Get service capabilities

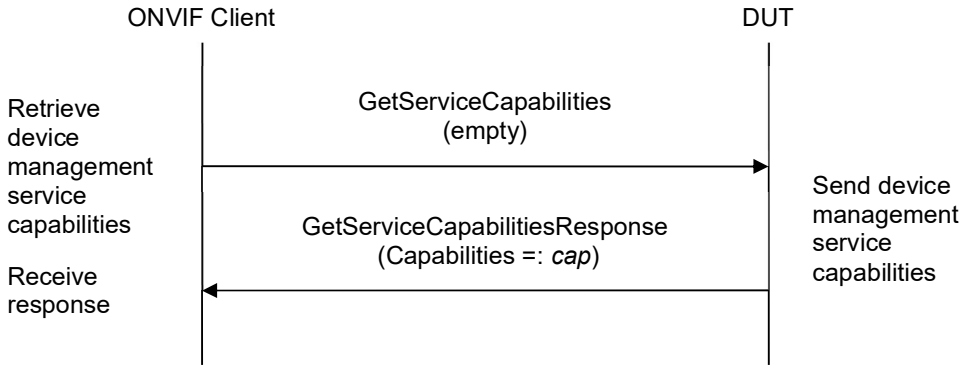
Name: HelperGetServiceCapabilities

Procedure Purpose: Helper procedure to get service capabilities.

Pre-requisite: GetServices command is supported by the DUT.

Input: None

Returns: The service capabilities (*cap*).



Procedure:

1. ONVIF Client invokes **GetServiceCapabilities**.
2. The DUT responds with a **GetServiceCapabilitiesResponse** message with parameters
   - Capabilities =: *cap*

Procedure Result:

PASS –

The DUT passed all assertions.

FAIL –

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

A.22 Get Device Management capabilities

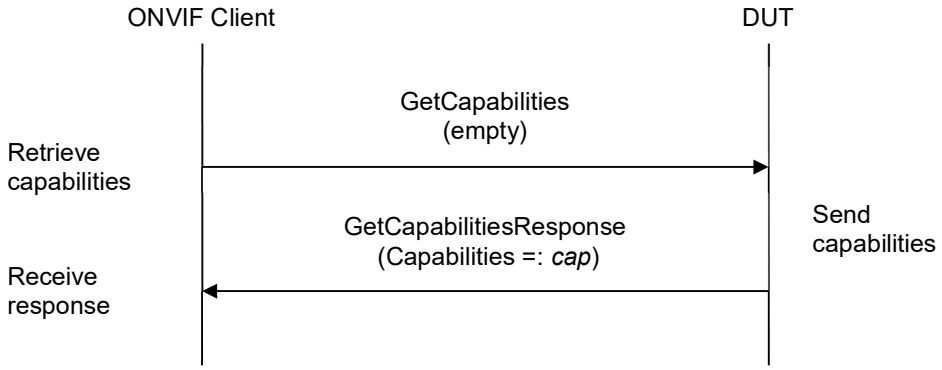
Name: HelperGetDeviceCapabilities

Procedure Purpose: Helper procedure to get device managemet capabilities.

Pre-requisite: GetCapabilities command is supported by the DUT.

Input: None

Returns: The service capabilities (*deviceManagementCap*).



Procedure:

1. ONVIF Client invokes **GetCapabilities**.
2. The DUT responds with a **GetCapabilitiesResponse** message with parameters
   - Capabilities =: *cap*
3. Set the following:
   - *deviceManagementCap* =: *cap*.Device

Procedure Result:

PASS –

The DUT passed all assertions.

FAIL –

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

A.23 Restoring System Date and Time

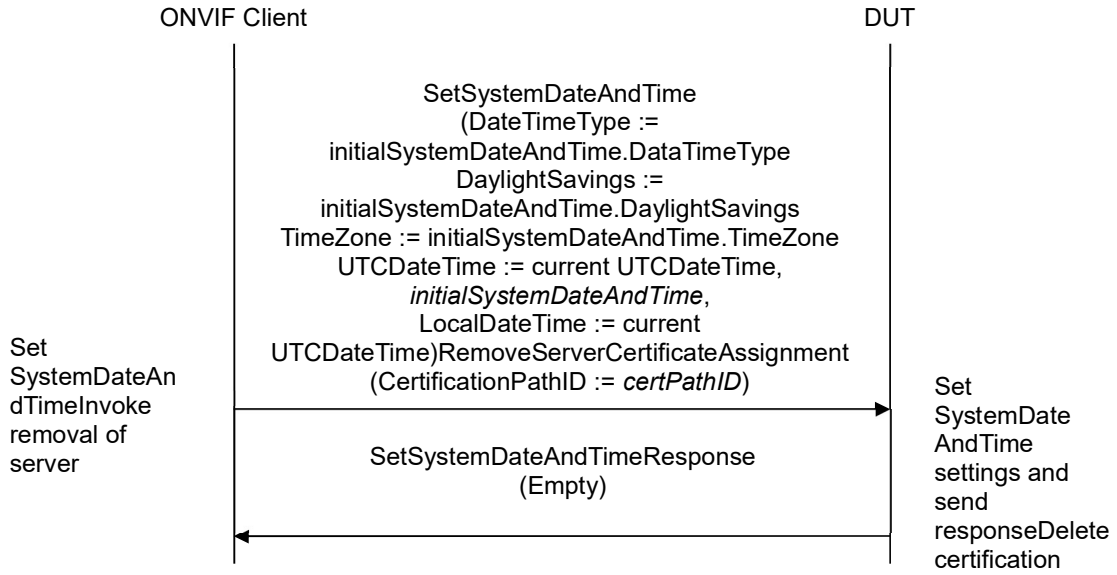
Name: HelperRestoreSystemDateAndTime

Procedure Purpose: Helper procedure to restoring System Date and Time.

Pre-requisite: None.

Input: The SystemDateAndTime (*initialSystemDateAndTime*) to restore.

Returns: None.



Procedure:

1. ONVIF Client invokes **SetSystemDateAndTime** with parameters
   - DateTimeType := *initialSystemDateAndTime*.DataTimeType
   - DaylightSavings := *initialSystemDateAndTime*.DaylightSavings
   - TimeZone := *initialSystemDateAndTime*.TimeZone
   - If *initialSystemDateAndTime*.UTCDateTime is not null UTCDateTime := current UTCDateTime
   - If *initialSystemDateAndTime*.LocalDateTime is not null LocalDateTime := current UTCDateTime
2. The DUT responds with a **SetSystemDateAndTimeResponse** message.

Procedure Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send **SetSystemDateAndTimeResponse** message.

A.24 Set NTP settings

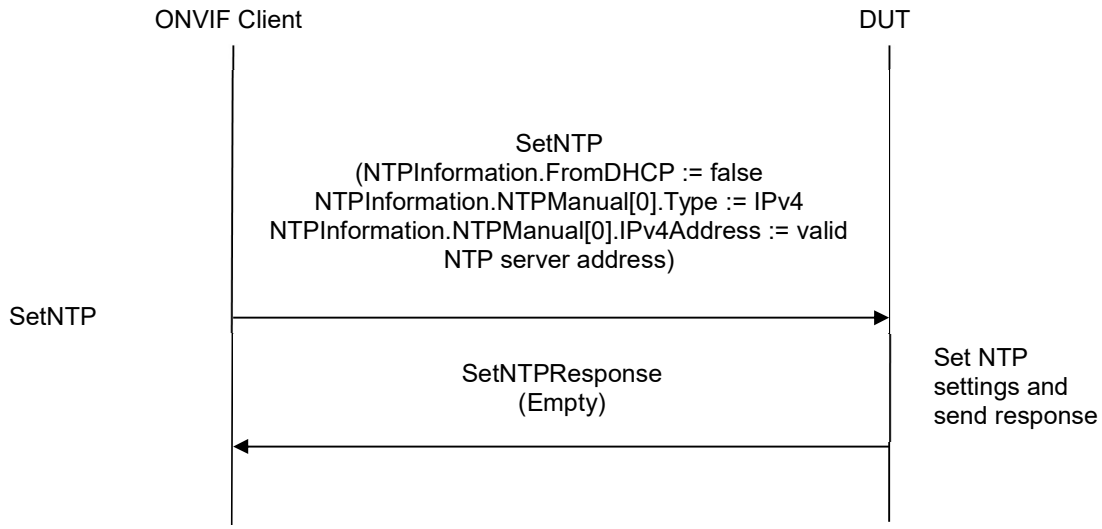
Name: HelperSetNTP

Procedure Purpose: Helper procedure to configure DUT with proper NTP server.

Pre-requisite: NTP is supported by the DUT as indicated by the Network.NTP capability. A valid NTP server address should be configured in the DUT.

Input: None.

Returns: None.



Procedure:

1. ONVIF Client invokes **SetNTP** with parameters
   - NTPInformation.FromDHCP := false
   - NTPInformation.NTPManual[0].Type := IPv4
   - NTPInformation.NTPManual[0].IPv4Address := valid NTP server address
2. The DUT responds with a **SetNTPResponse** message.

Procedure Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send **SetNTPResponse** message.

A.25 Restoring NTP settings

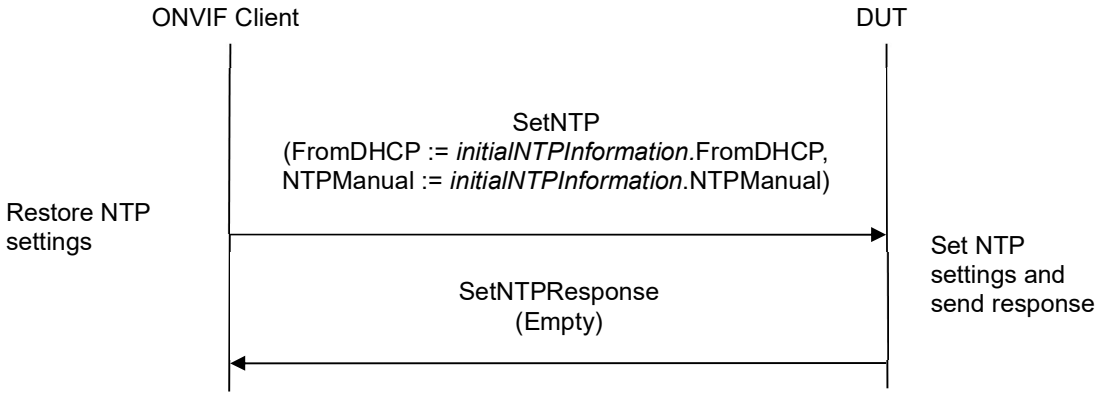
Name: HelperRestoreNTP

Procedure Purpose: Helper procedure to restoring NTP settings.

Pre-requisite: NTP is supported by the DUT as indicated by the Network.NTP capability.

Input: The initial NTP settings (*initialNTPInformation*) to restore.

Returns: None.



Procedure:

3. ONVIF Client invokes **SetNTP** with parameters
   - FromDHCP := *initialNTPInformation*.FromDHCP
   - NTPManual := *initialNTPInformation*.NTPManual
4. The DUT responds with a **SetNTPResponse** message.

Procedure Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send **SetNTPResponse** message.