Driving IP-based physical security through global standardization

ONVIF™

Base Test Specification

Version 1.02.4

July, 2011

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

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 specification describes the test cases needed to verify the [ONVIF Core] 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 Core], clarifying the requirements wherever needed. And also this specification acts an input document to the development of test tool which will be used to test the ONVIF device implementation conformance towards the [ONVIF Core]. This test tool is referred as Network Video Client (NVC) hereafter.

1.1 Scope

This ONVIF Base Test Specificatoin 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 accroding to ONVIF core services which is defined in [ONVIF Core].

The principal intended purposes are:

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

This specification does not address the following.

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

The set of ONVIF Test Specification will not cover the complete set of requirements as defined in [ONVIF Core]; instead it would cover subset of it. The scope of this specification is to derive all the normative requirements of [ONVIF Core] which is related to ONVIF core services and some of optional requirements.

This ONVIF Base Test Specification covers core parts of functional blocks in [ONVIF Core]. 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 Core]. 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 Table 1 for Device Discovery Test.

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 Core]. 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)

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

- Namespace handling

1.1.4 Event Handling

Event handling covers the test cases for the verification of the Event service as mentioned in [ONVIF Core] 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.

1.1.5 Security

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

2 Terms and Definitions

2.1 Definitions

Address	An address refers to a URI
Capability	The capability commands allow an NVC to ask for the services provided by an ONVIF device.
Network	A network is an interconnected group of devices communicating using the Internet protocol.
NVC Test Tool	Network Video Client Test tool that tests the Network Video Transmitter device compliance towards the ONVIF Core Specification v1.0
Proxy Server	A server that services the requests of its clients (NVC) by forwarding requests to other servers. A Proxy provides indirect network connections to its clients (NVC).
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.

2.2 Abbreviations

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
NVT	Network Video Transmitter
NVC	Network Video Client
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 describes about the test setup and prerequisites needed, and the test policies that should be followed for test case execution.

3.1 Test Setup

3.1.1 Network Configuration for 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.

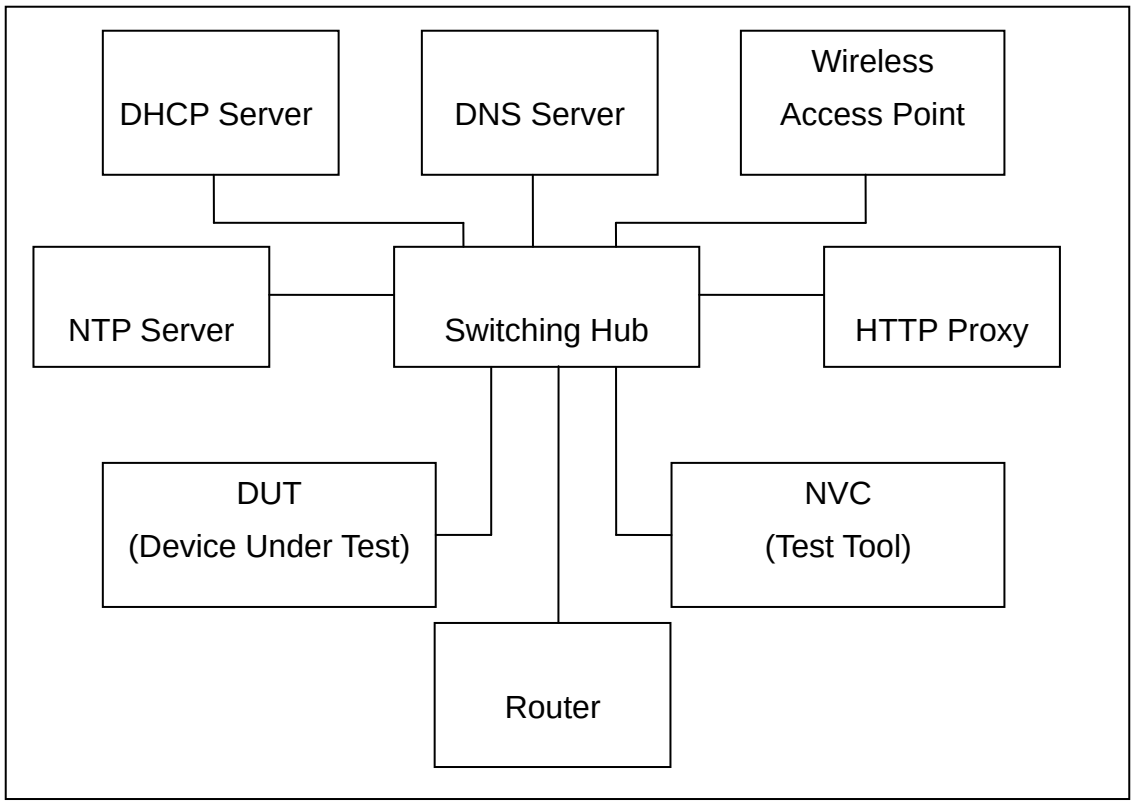


Figure 1: Test Configuration for DUT

DUT: ONVIF device to be tested. Hereafter, this is referred to as DUT (Device Under Test).

NVC 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 tunneling over HTTP.

Wireless Access Point: provides wireless connectivity to the devices that support wireless connection.

DNS Server: provides DNS related information to the connected devices.

DHCP Server: provides IPv4 Address to the connected devices.

NTP Server: provides time synchronization between NVC and DUT.

Switching Hub: provides network connectivities 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 are

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

3.3 Requirement Level

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119]. Additionally, this specification describes the key words "MUST IF SUPPORTED", "SHOULD IF SUPPORTED", "MUST IF IMPLEMENTED [A]", "MUST IF SUPPORTED [A] & IMPLEMENTED [B]", "SHOULD IF IMPLEMENTED [A]", "SHOULD IF SUPPORTED [A] & IMPLEMENTED [B]". For the details on how the requirement levels affect the test cases described in this specifciation, refer to [ONVIF Test].

3.4 Test Policy

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

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

3.4.1 IP Configuration

- The device under test must be discovered by the NVC device that exists in the testing environment.
- The device under test must support SetNetworkInterfaces method.
- The device under test that supports Link-Local address of IPv4 must support SetZeroConfiguration method.
- The following tests are performed about IPv4.
  - Static IP configuration
  - Dynamic IP configuration of Link-Local address

- Dynamic IP configuration (DHCP)

- The following tests are performed about IPv6.

- Stateless IP configuration which accepts Router Advertisement.

- Stateless IP configuration which uses Neighbour Discovery.

- Stateful IP configuration (DHCPv6)

- The device under test must have at least one network interface that gives IPv4 connectivity. And it should have at least one network interface that gives 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.

Please refer to Section 4 for IP Configuration Test Cases.

3.4.2 Device Discovery

- The device under test must be discovered by the NVC device that exists in the testing environment.

- Failure to discover the device on the network constitutes failure of the test procedure.

- Failure to locate the device services on the network constitutes failure of the test procedure.

- Failure to select the device for interaction constitutes failure of the test procedure.

- Failure to exist of the namespace of defined Core Specification constitutes failure of the test procedure.

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

Please refer to Section 5 for Device Discovery Test Cases.

3.4.3 Device Management

- The device under test must demonstrate device, media and event capability. A DUT that does not display mandatory device capability constitutes failure of test procedure.

- Some commands like CreateUsers, SetUser, etc may have restricted access. In that case NVC should execute the test cases in the administrative mode.

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

Please refer to Annex A.2 for valid host name.

Please refer to Section 6 for Device Management Test cases.

3.4.4 Event Handling

- Prior to the execution of Event handling test cases, DUT MUST be discovered by NVC and it MUST demonstrate event capabilities to NVC using the device management service.
- If the DUT supports “property events” NVC uses the SetSynchronizationPoint method from the event service to trigger events for testing Basic Notification Interface, Realtime Pullpoint Interface; NVC 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.
- NVC and DUT time SHOULD be synchronized.
- In certain test cases the Test Tool MAY create a Subscription Manager representing the subscription. In such cases the procedure will take care that all new 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.

Please refer to Section 7 for Event handling Test Cases.

3.4.5 Security

- The DUT MUST support WS-Security User token profile. Consequently the DUT MUST 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, an NVC MUST be able to access any given part of any given service supported by the DUT with a user with Administrator rights.

Please refer to Section 8 for Security Test Cases.

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

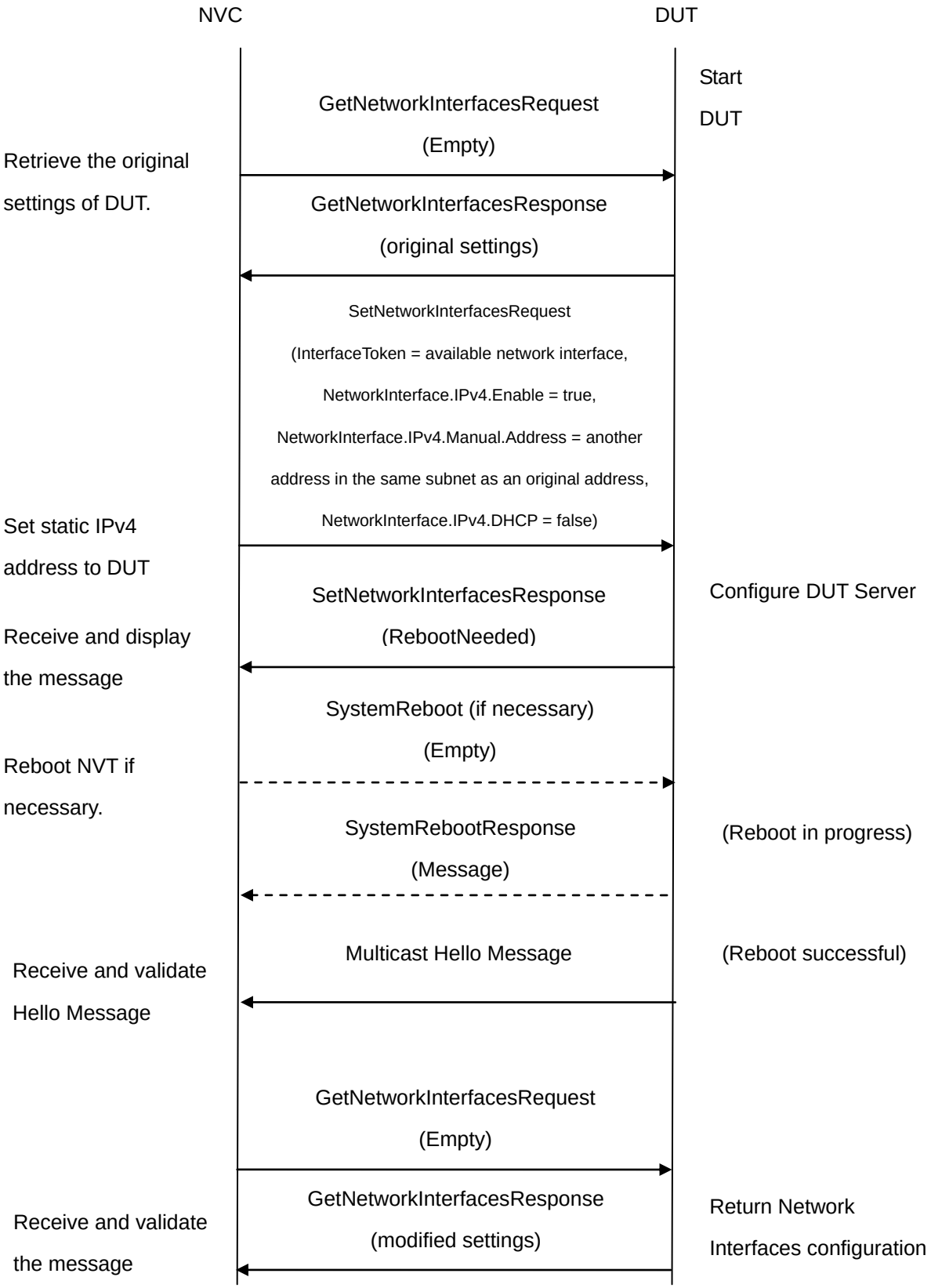
Requirement Level: MUST

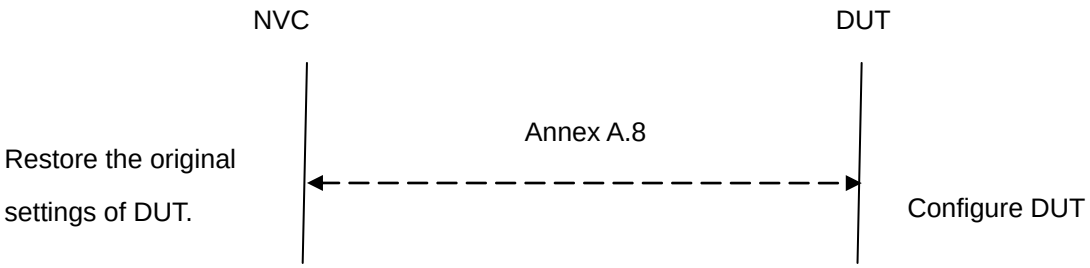
Test Purpose: To test IPv4 Static IP Configuration.

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:


NVC
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 NVT 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 NVC.
2. Start the DUT.
3. NVC will invoke GetNetworkInterfacesRequest message to retrieve the original settings of DUT.
4. NVC 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, NVC will invoke SystemReboot message to restart DUT. Otherwise, continue to step-8.
7. DUT will return SystemRebootResponse message.
8. DUT will send Multicast Hello message from newly configured address.
9. NVC will receive and validate Hello message sent from newly configured address by DUT.
10. NVC will invoke GetNetworkInterfacesRequest message to newly configured address to retrieve the modified settings of DUT.
11. NVC will receive and validate GetNetworkInterfacesResponse message sent from newly configured address by DUT.
12. NVC will restore the original settings by following the procedure mentioned in Annex A.8.

Test Result:

PASS –

DUT passes 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 LINK LOCAL ADDRESS

Test Label: IP Configuration IPv4 Link-Local Address Configuration

Test Case ID: IPCONFIG-1-1-2

ONVIF Core Specification Coverage: IP Configuration, Set zero configuration

Command Under Test: SetZeroConfiguration, GetZeroConfiguration

WSDL Reference: devicemgmt.wsdl

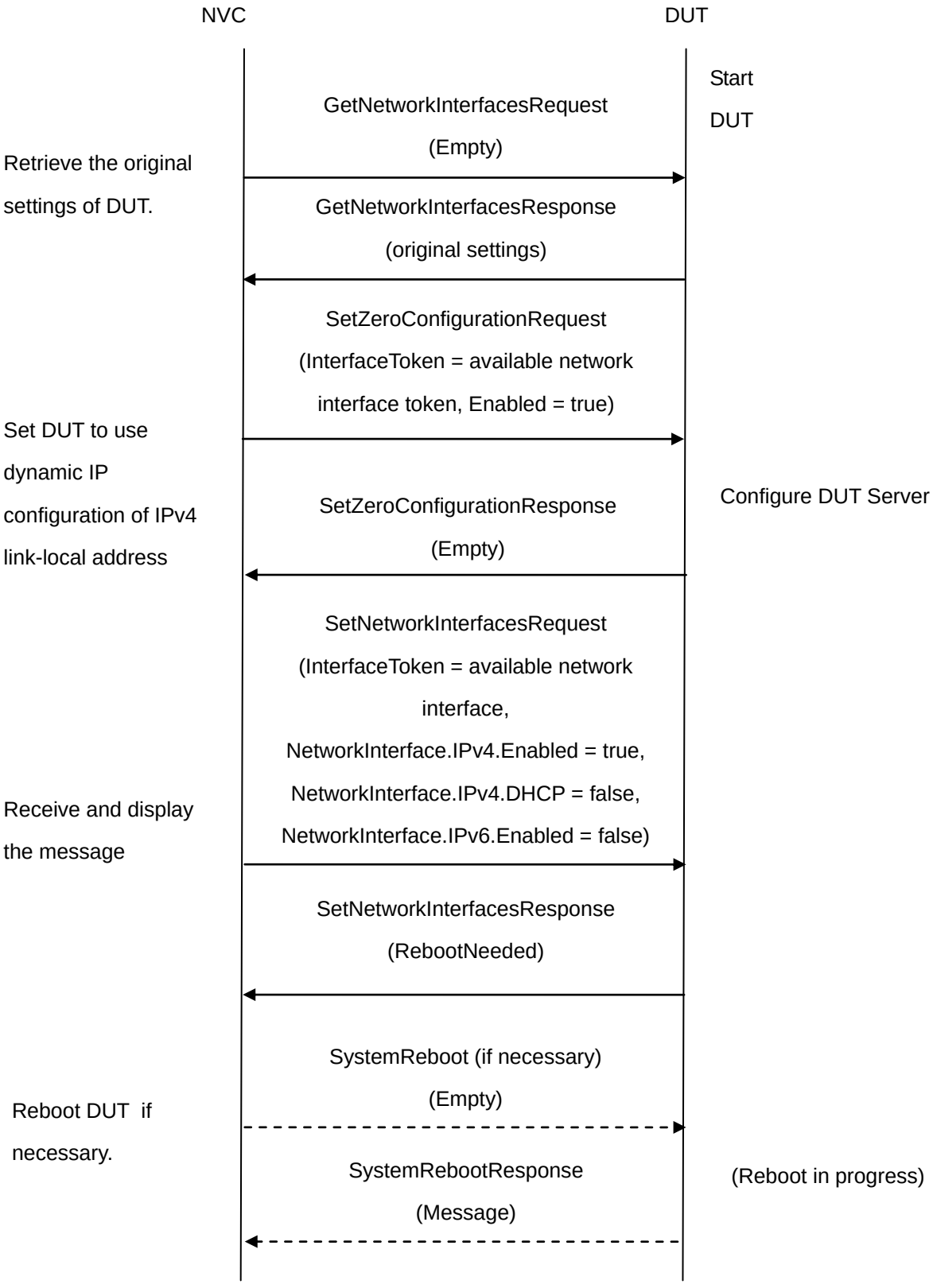
Requirement Level: SHOULD

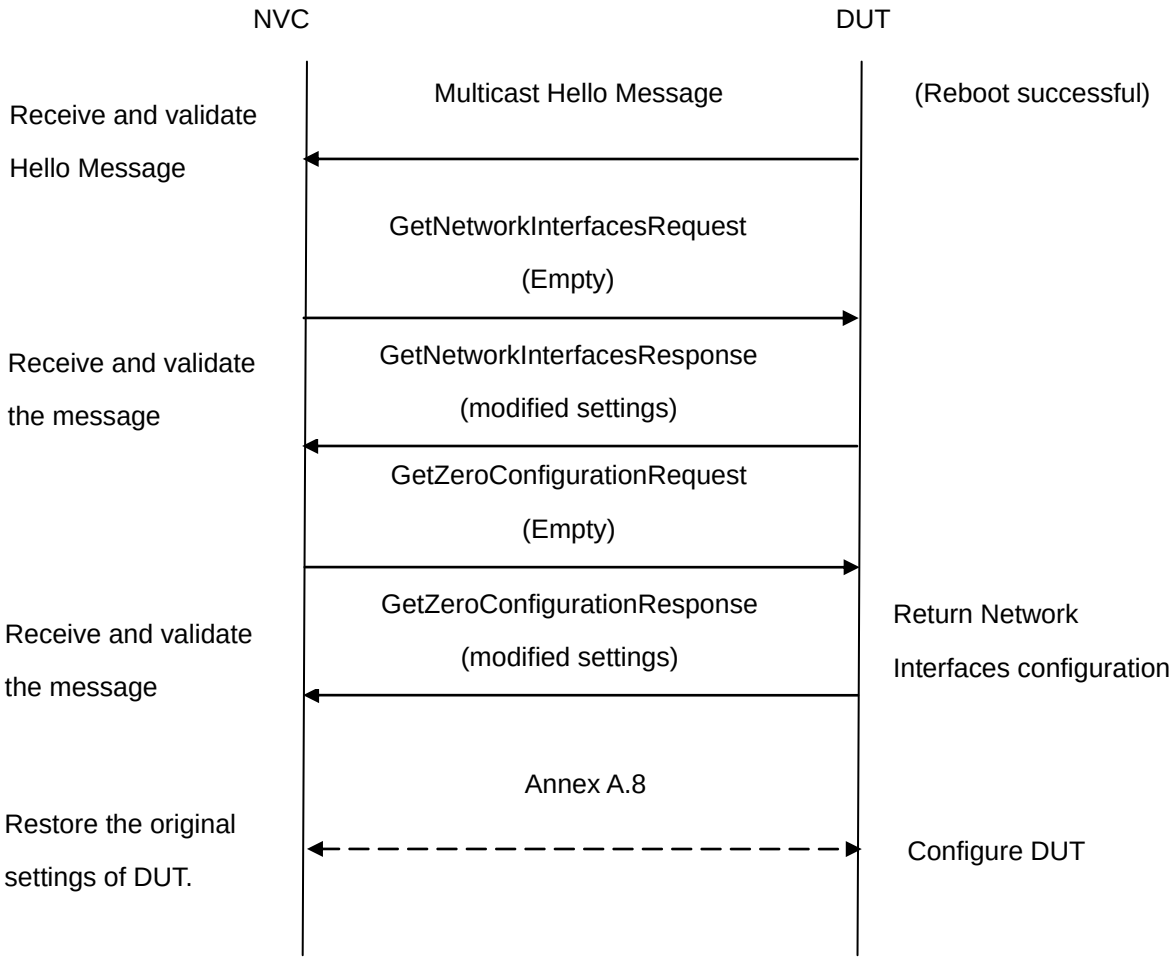
Test Purpose: To test IPv4 Link-Local Address Configuration.

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

Test Configuration: NVC and DUT

Test Sequence:


NVC
DUT
Start DUT
GetNetworkInterfacesRequest
(Empty)
Retrieve the original settings of DUT.
GetNetworkInterfacesResponse
(original settings)
SetZeroConfigurationRequest
(InterfaceToken = available network interface token, Enabled = true)
Set DUT to use dynamic IP configuration of IPv4 link-local address
SetZeroConfigurationResponse
(Empty)
Configure DUT Server
SetNetworkInterfacesRequest
(InterfaceToken = available network interface,
NetworkInterface.IPv4.Enabled = true,
NetworkInterface.IPv4.DHCP = false,
NetworkInterface.IPv6.Enabled = false)
Receive and display the message
SetNetworkInterfacesResponse
(RebootNeeded)
SystemReboot (if necessary)
(Empty)
Reboot DUT if necessary.
SystemRebootResponse
(Message)
(Reboot in progress)



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke GetNetworkInterfacesRequest message to retrieve the original settings of DUT.
4. NVC will invoke SetZeroConfigurationRequest message to set DUT to use dynamic IP configuration of IPv4 link-local address (InterfaceToken = available network interface token, Enabled = true).
5. DUT will return SetZeroConfigurationResponse message.
6. NVC will invoke SetNetworkInterfacesRequest message to set static IPv4 address to DUT (InterfaceToken = available network interface, NetworkInterface.IPv4.Enabled = true, NetworkInterface.IPv4.DHCP = false, NetworkInterface.IPv6.Enabled = false).
7. DUT will return SetNetworkInterfacesResponse message.

8. If necessary, NVC will invoke SystemReboot message to restart DUT. Otherwise, continue to step-10.
9. DUT will return SystemRebootResponse message.
10. DUT will send Multicast Hello message from newly configured address.
11. NVC will receive and validate Hello message sent from newly configured address by DUT.
12. NVC will invoke GetNetworkInterfacesRequest message to newly configured address to retrieve the modified settings of DUT.
13. NVC will receive and validate GetNetworkInterfacesResponse message sent from newly configured address by DUT.
14. NVC will invoke GetZeroConfigurationRequest message to newly configured address to retrieve the modified settings of DUT.
15. NVC will receive and validate GetZeroConfigurationResponse message sent from newly configured address by DUT.
16. NVC will restore the original settings by following the procedure mentioned in Annex A.8.

Test Result:

PASS –

DUT passes 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 SetZeroConfigurationResponse message

The DUT did not send GetNetworkInterfacesResponse message.

The DUT did not send GetZeroConfigurationResponse message.

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

The DUT did not send correct zero configuration information (i.e. InterfaceToken = available network interface token, Enabled = true, Address = new IP address).in GetZeroConfigurationResponse message.

4.1.3 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

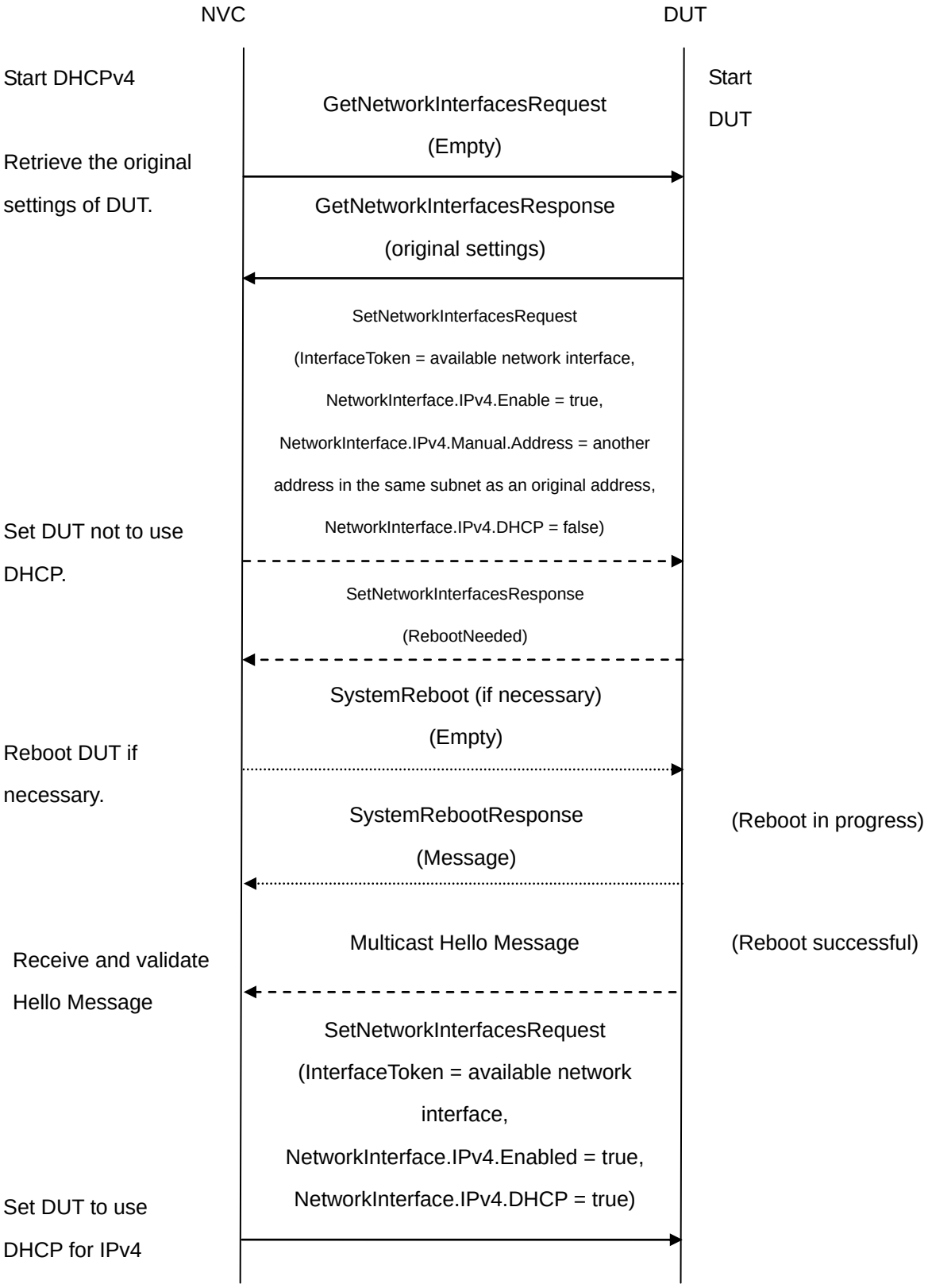
Requirement Level: MUST

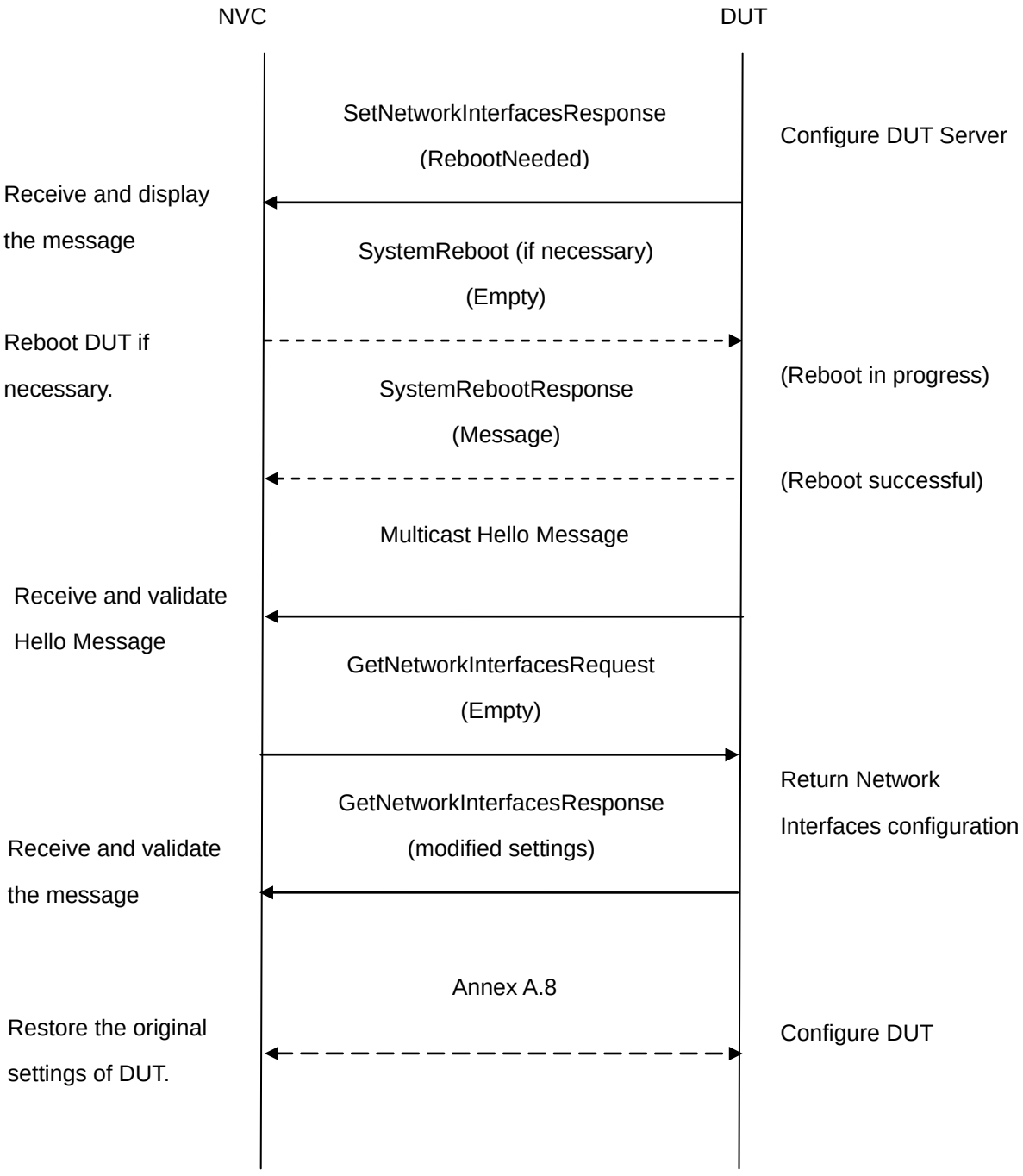
Test Purpose: To test IPv4 DHCP Configuration.

Pre-Requisite: None.

Test Configuration: NVC and DUT

Test Sequence:


NVC
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)
Set DUT to use DHCP for IPv4



Test Procedure:

1. Start DHCPv4 server
2. Start an NVC.
3. Start the DUT.

4. NVC will invoke GetNetworkInterfacesRequest message to retrieve the original settings of DUT.
5. If NetworkInterface.IPv4.DHCP == true in the original settings, NVC 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, NVC will invoke SystemReboot message to restart DUT.
7. DUT will send Multicast Hello message from newly configured address.
8. NVC 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, NVC will invoke SystemReboot message to restart DUT. Otherwise, continue to step-12.
11. DUT will return SystemRebootResponse message.
12. DUT will send Multicast Hello message from newly configured address.
13. NVC will receive and validate Hello message sent from newly configured address by DUT.
14. NVC will invoke GetNetworkInterfacesRequest message to newly configured address to retrieve the modified settings of DUT.
15. NVC will receive and validate GetNetworkInterfacesResponse message sent from newly configured address by DUT.
16. If the modified settings are different from original settings then NVC will restore the original settings by following the procedure mentioned in Annex A.8.

Test Result:

PASS –

DUT passes 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.

4.2 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

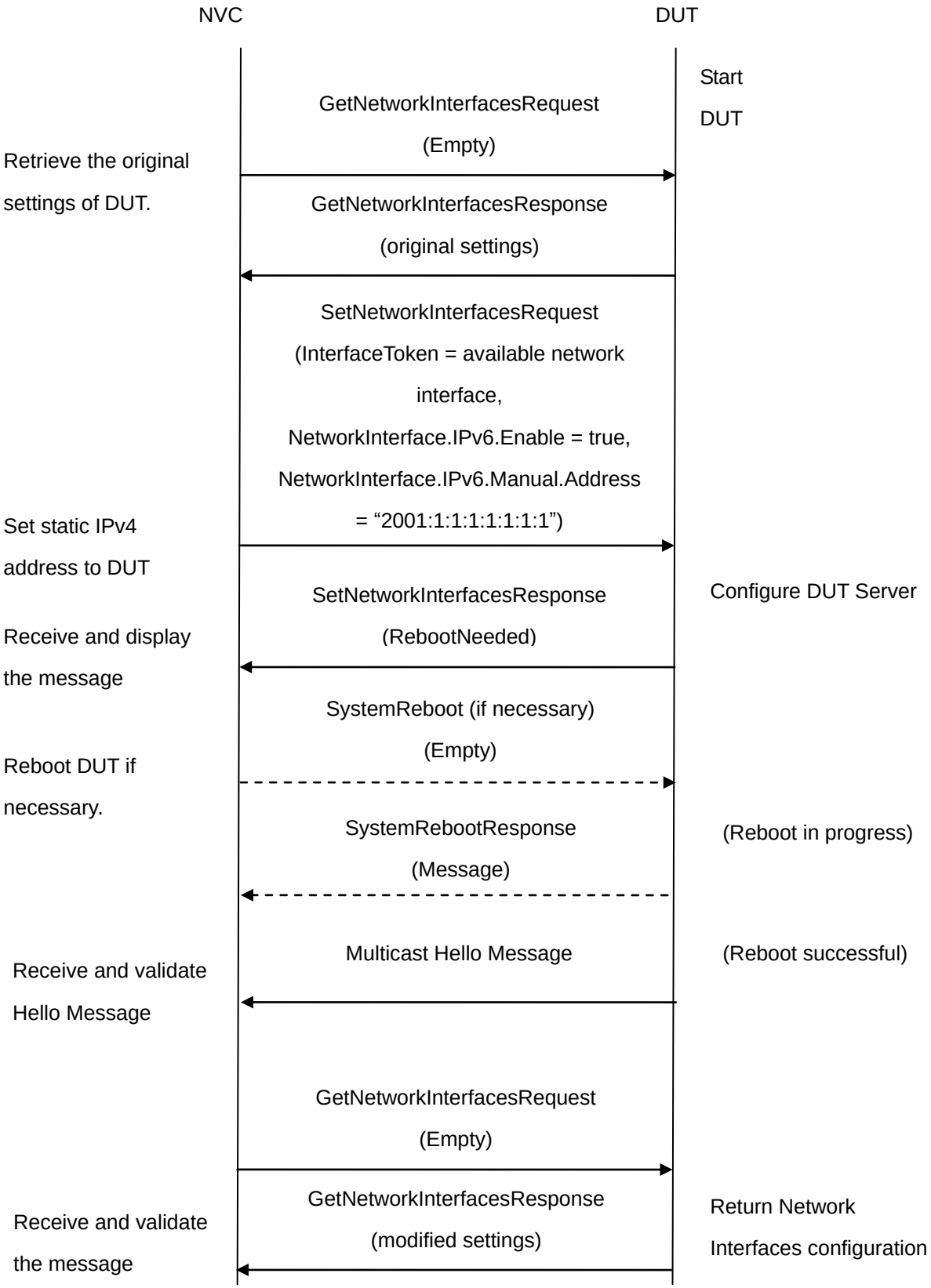
Requirement Level: MUST IF IMPLEMENTED (IPv6)

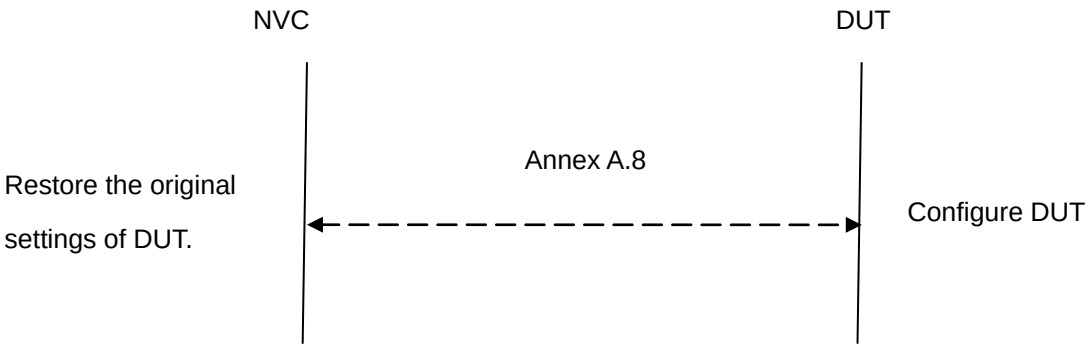
Test Purpose: To test IPv6 Static IP Configuration.

Pre-Requisite: IPv6 is implemented by DUT.

Test Configuration: NVC and DUT

Test Sequence:


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

Test Result:

PASS –

DUT passes 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

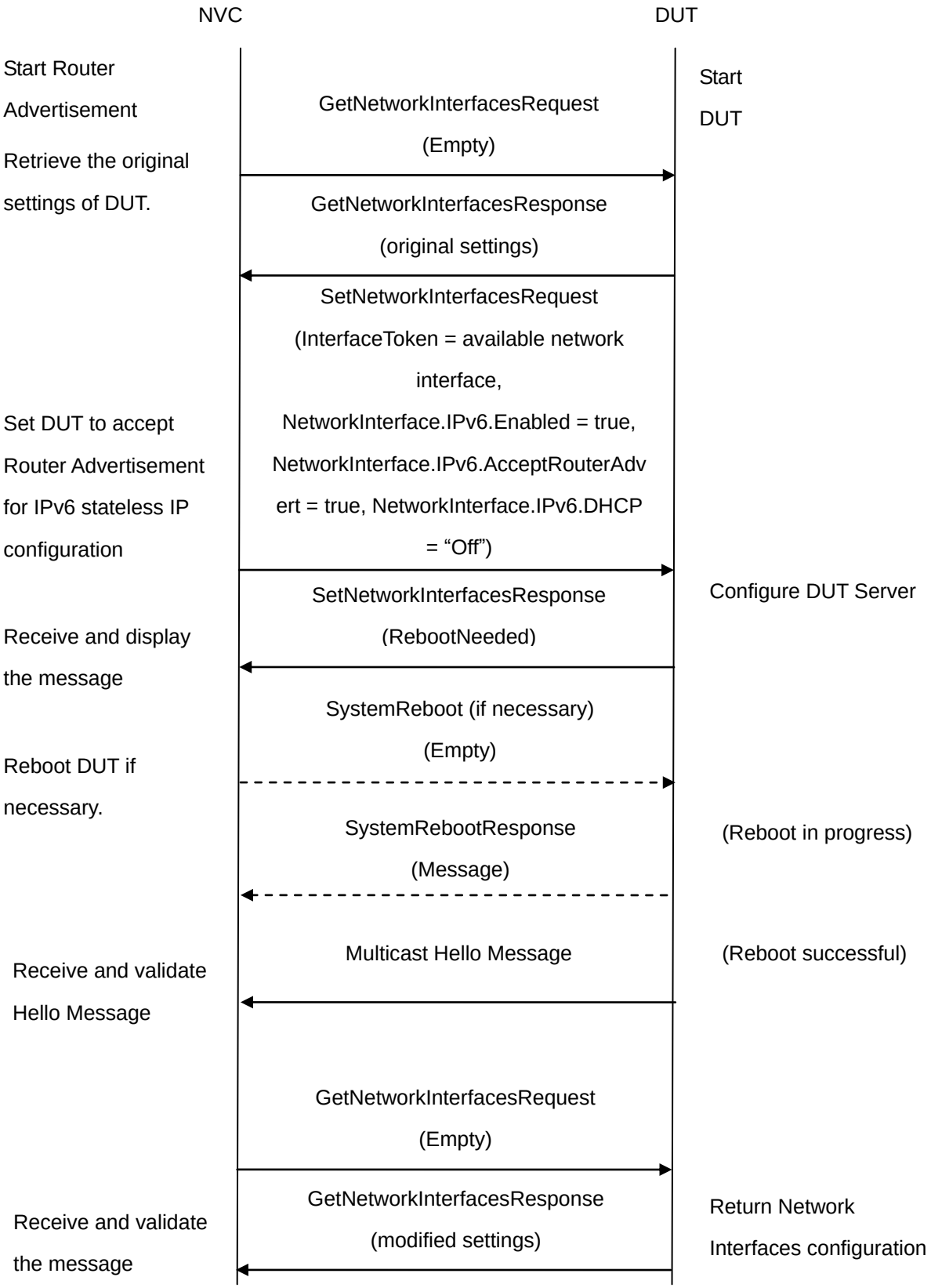
Requirement Level: MUST IF IMPLEMENTED (IPv6)

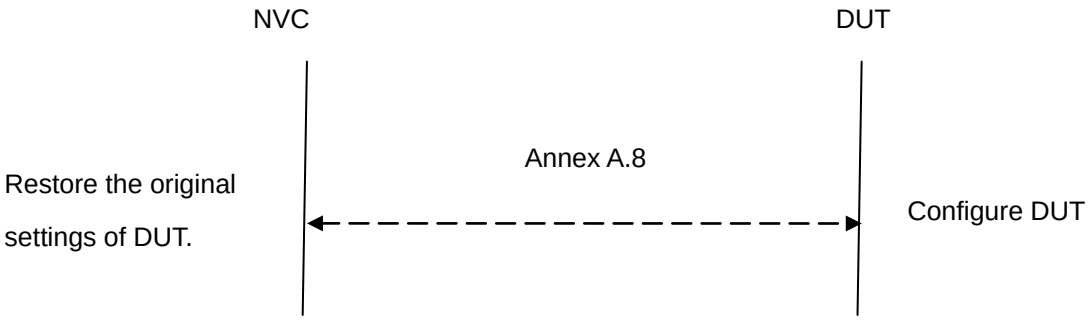
Test Purpose: To test IPv6 Stateless IP Configuration which Accepts Router Advertisement.

Pre-Requisite: IPv6 is implemented by DUT.

Test Configuration: NVC and DUT

Test Sequence:


NVC
DUT
Start Router Advertisement
Retrieve the original settings of DUT.
GetNetworkInterfacesRequest
(Empty)
GetNetworkInterfacesResponse
(original settings)
Start DUT
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 NVC.
3. Start the DUT
4. NVC will invoke GetNetworkInterfacesRequest message to retrieve the original settings of DUT
5. NVC 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. DUT will return SetNetworkInterfacesResponse message.
7. If necessary, NVC will invoke SystemReboot message to restart DUT. Otherwise, continue to step-9.
8. DUT will return SystemRebootResponse message.
9. DUT will send Multicast Hello message from newly configured address.
10. NVC will receive and validate Hello message sent from newly configured address by DUT.
11. NVC will invoke GetNetworkInterfacesRequest message to newly configured address to retrieve the modified settings of DUT.
12. NVC will receive and validate GetNetworkInterfacesResponse message sent from newly configured address by DUT.
13. NVC will restore the original settings by following the procedure mentioned in Annex A.8.

Test Result:

PASS –

DUT passes 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 Neighbour Discovery.

Test Case ID: IPCONFIG-2-1-3

ONVIF Core Specification Coverage: IP Configuration

Command Under Test: None

WSDL Reference: devicemgmt.wsdl

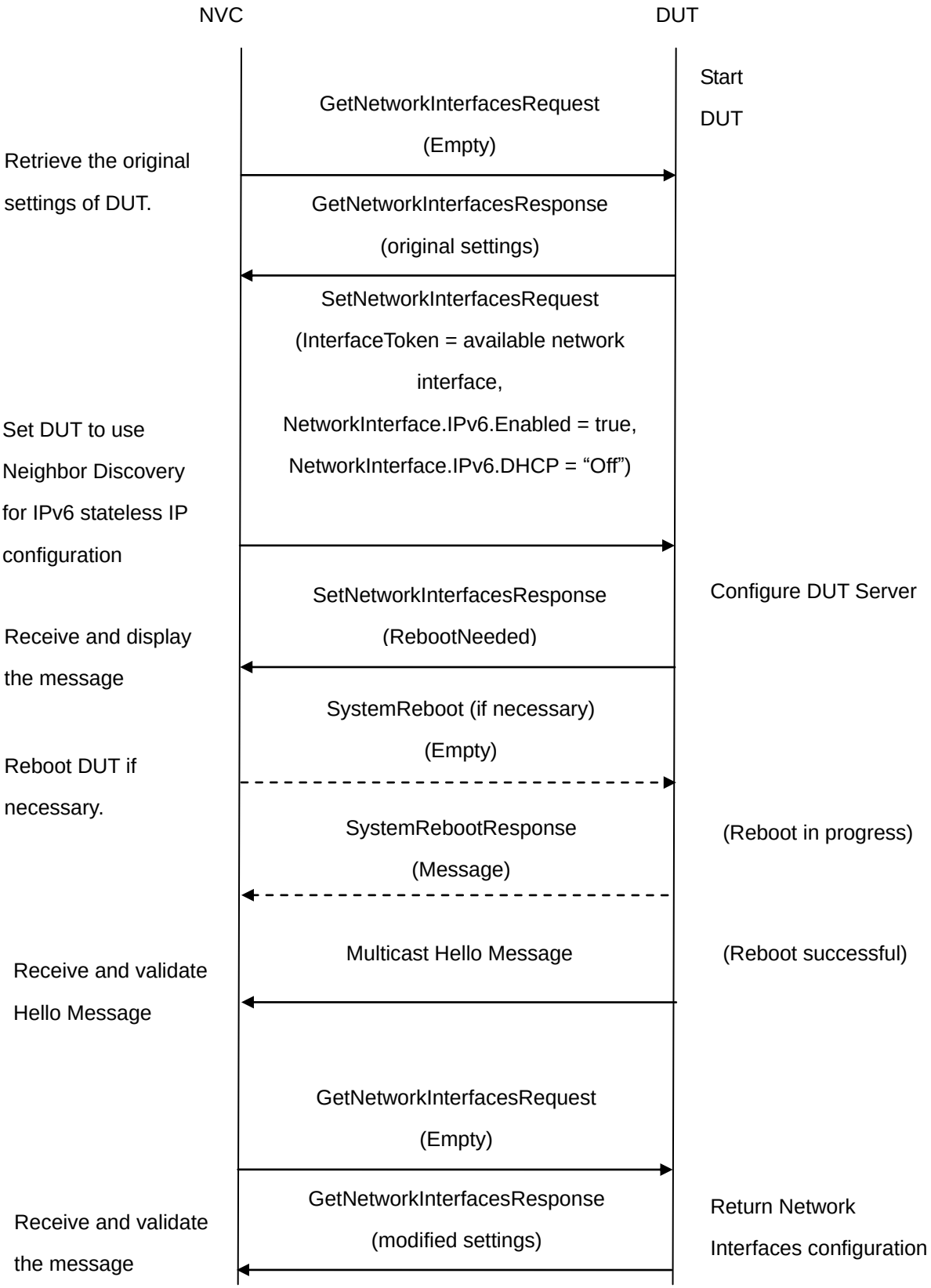
Requirement Level: MUST IF IMPLEMENTED (IPv6)

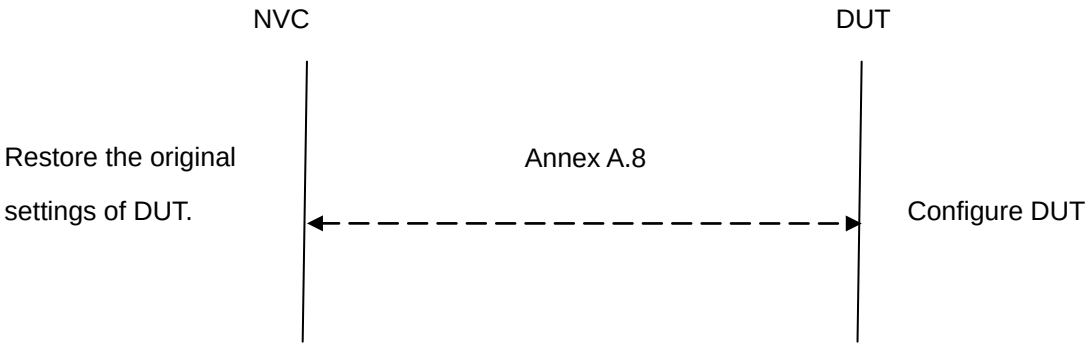
Test Purpose: To test IPv6 Stateless IP Configuration which use Neighbour Discovery

Pre-Requisite: IPv6 is implemented by DUT.

Test Configuration: NVC and DUT

Test Sequence:


NVC
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
Configure DUT Server
SetNetworkInterfacesResponse
(RebootNeeded)
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 NVC.
2. Start the DUT.
3. NVC will invoke GetNetworkInterfacesRequest message to retrieve the original settings of DUT.
4. NVC 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, NVC will invoke SystemReboot message to restart DUT. Otherwise, continue to step-8.
7. DUT will return SystemRebootResponse message.
8. DUT will send Multicast Hello message from newly configured address.
9. NVC will receive and validate Hello message sent from newly configured address by DUT.
10. NVC will invoke GetNetworkInterfacesRequest message to newly configured address to retrieve the modified settings of DUT.
11. NVC will receive and validate GetNetworkInterfacesResponse message sent from newly configured address by DUT.
12. NVC will restore the original settings by following the procedure mentioned in Annex A.8.

Test Result:

PASS –

DUT passes 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

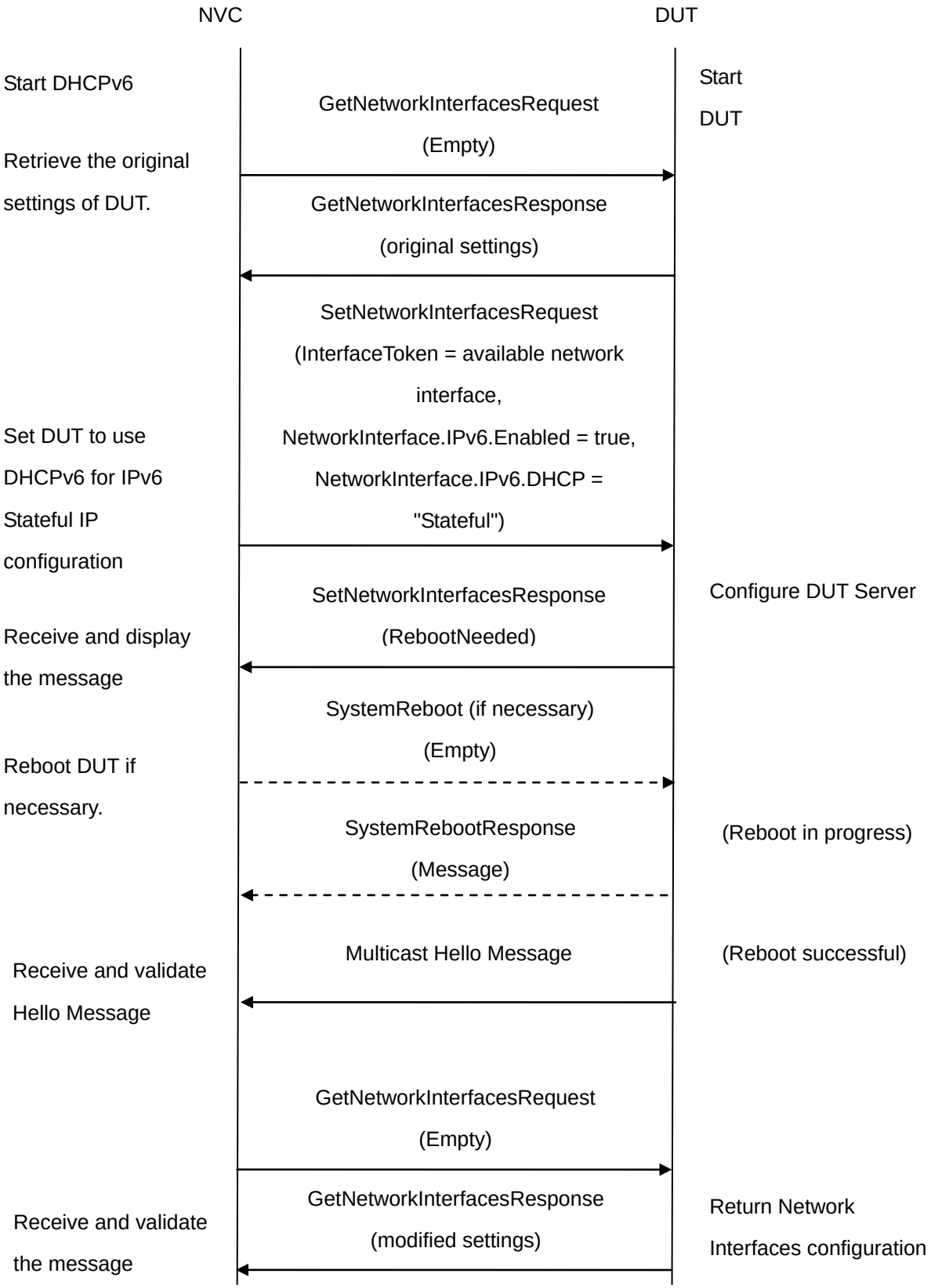
Requirement Level: MUST IF IMPLEMENTED (IPv6)

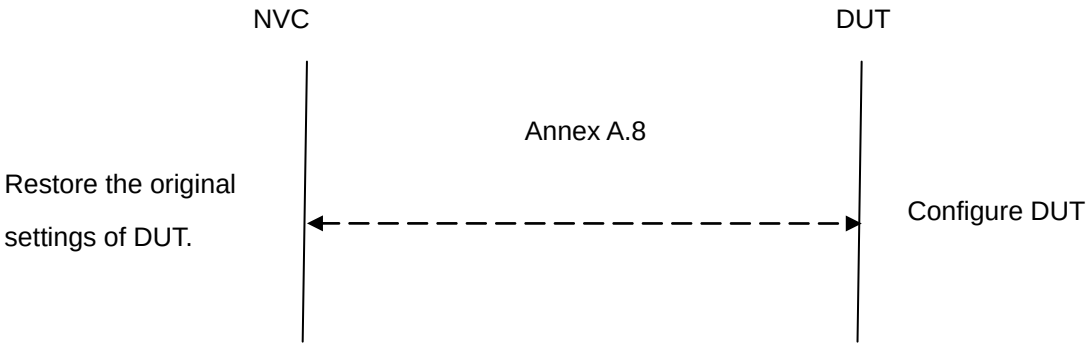
Test Purpose: To test IPv6 Stateful IP Configuration (DHCPv6)

Pre-Requisite: IPv6 is implemented by DUT.

Test Configuration: NVC and DUT

Test Sequence:


NVC
DUT
Start DHCPv6
Retrieve the original settings of DUT.
GetNetworkInterfacesRequest
(Empty)
GetNetworkInterfacesResponse
(original settings)
Start DUT
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 NVC.
3. Start the DUT.
4. NVC will invoke GetNetworkInterfacesRequest message to retrieve the original settings of DUT.
5. NVC 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. DUT will return SetNetworkInterfacesResponse message.
7. If necessary, NVC will invoke SystemReboot message to restart DUT. Otherwise, continue to step-9.
8. DUT will return SystemRebootResponse message.
9. DUT will send Multicast Hello message from newly configured address.
10. NVC will receive and validate Hello message sent from newly configured address by DUT.
11. NVC will invoke GetNetworkInterfacesRequest message to newly configured address to retrieve the modified settings of DUT.
12. NVC will receive and validate GetNetworkInterfacesResponse message sent from newly configured address by DUT.
13. NVC will restore the original settings by following the procedure mentioned in Annex A.8.

Test Result:

PASS –

DUT passes 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

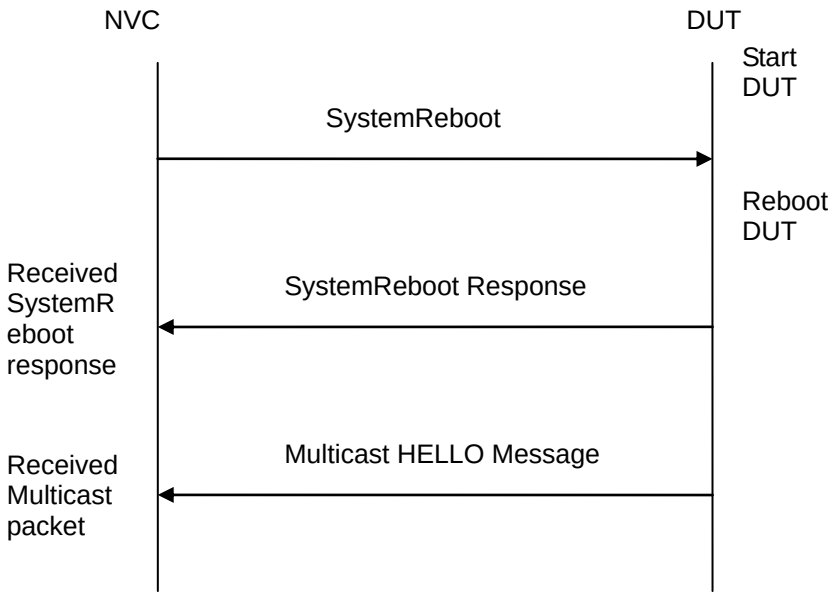
Requirement Level: MUST

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: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC
2. Start the DUT
3. NVC invokes SystemReboot message to reboot the DUT.
4. DUT sends SystemRebootResponse message.

5. NVC 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 –

DUT passes 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

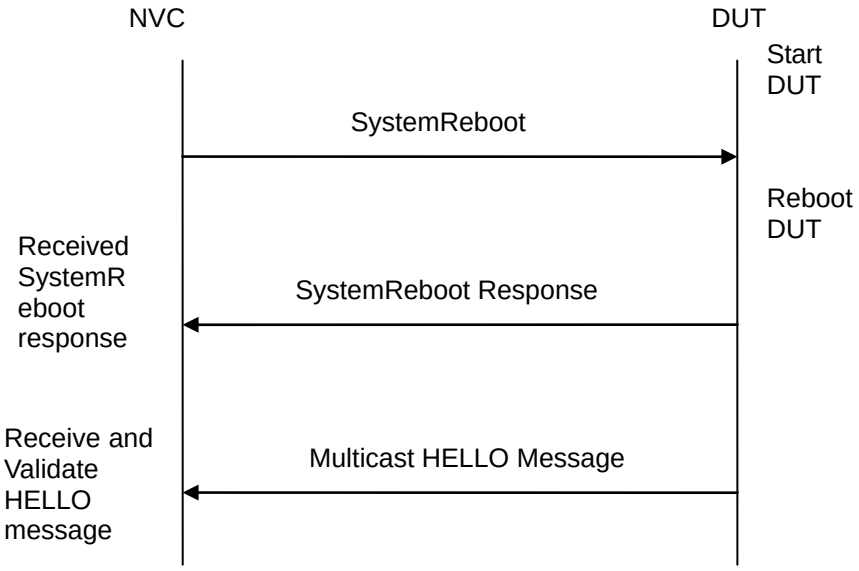
Requirement Level: MUST

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: NVC and DUT

Test Sequence:



Test Procedure:

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

Test Result:

PASS –

DUT passes 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 ("http://www.onvif.org/ver10/network/wsdl").

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

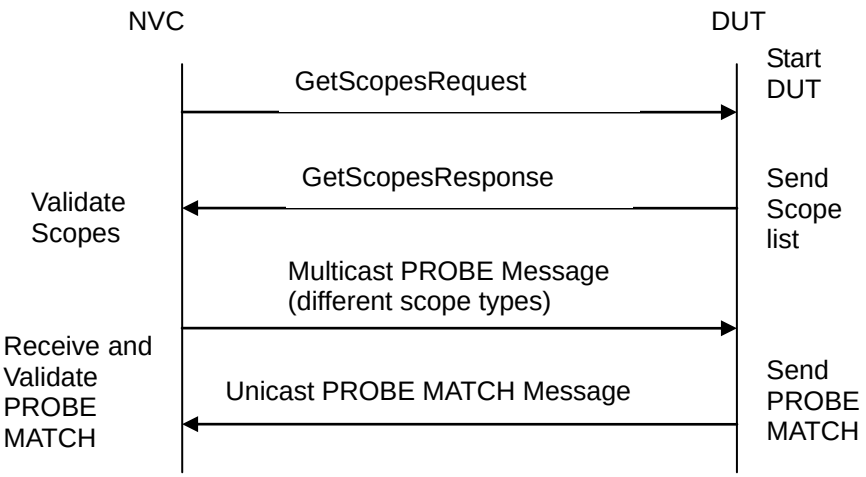
Requirement Level: MUST

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

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

Test Result:

PASS –

DUT passes 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, scope type, 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 APP_MAX_DELAY

The DUT did not send PROBE MATCH message with a namespace of the Types value ("http://www.onvif.org/ver10/network/wsdl").

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 APP_MAX_DELAY.

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

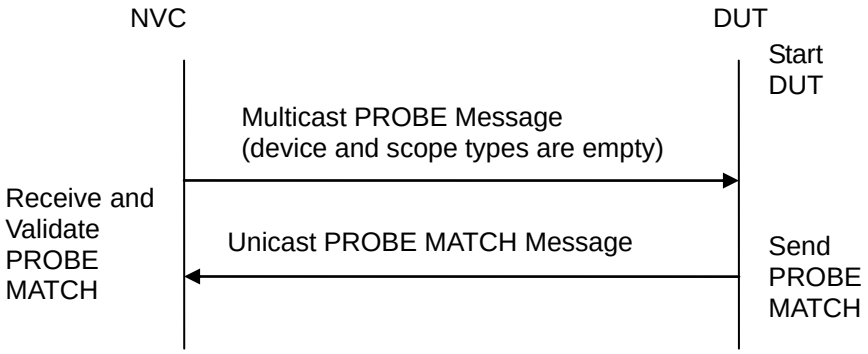
Requirement Level: MUST

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

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

Test Result:

PASS –

DUT passes all assertions.

FAIL –

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

The DUT did not send mandatory XML elements (device, scope type, 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 ("http://www.onvif.org/ver10/network/wsdl").

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.7 for the value of APP_MAX_DELAY.

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

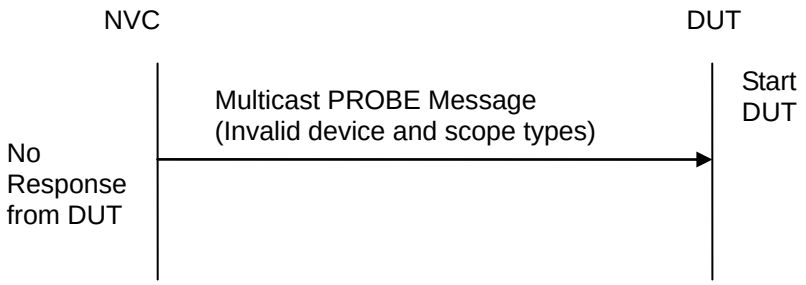
Requirement Level: MUST

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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 –

DUT passes 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

Requirement Level: OPTIONAL

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

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

5.7 DEVICE SCOPES CONFIGURATION

Test Label: Device Discovery Device Scope configurations.

Test Case ID: DISCOVERY-1-1-7

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

Command Under Test: AddScopes, SetScopes, RemoveScopes

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

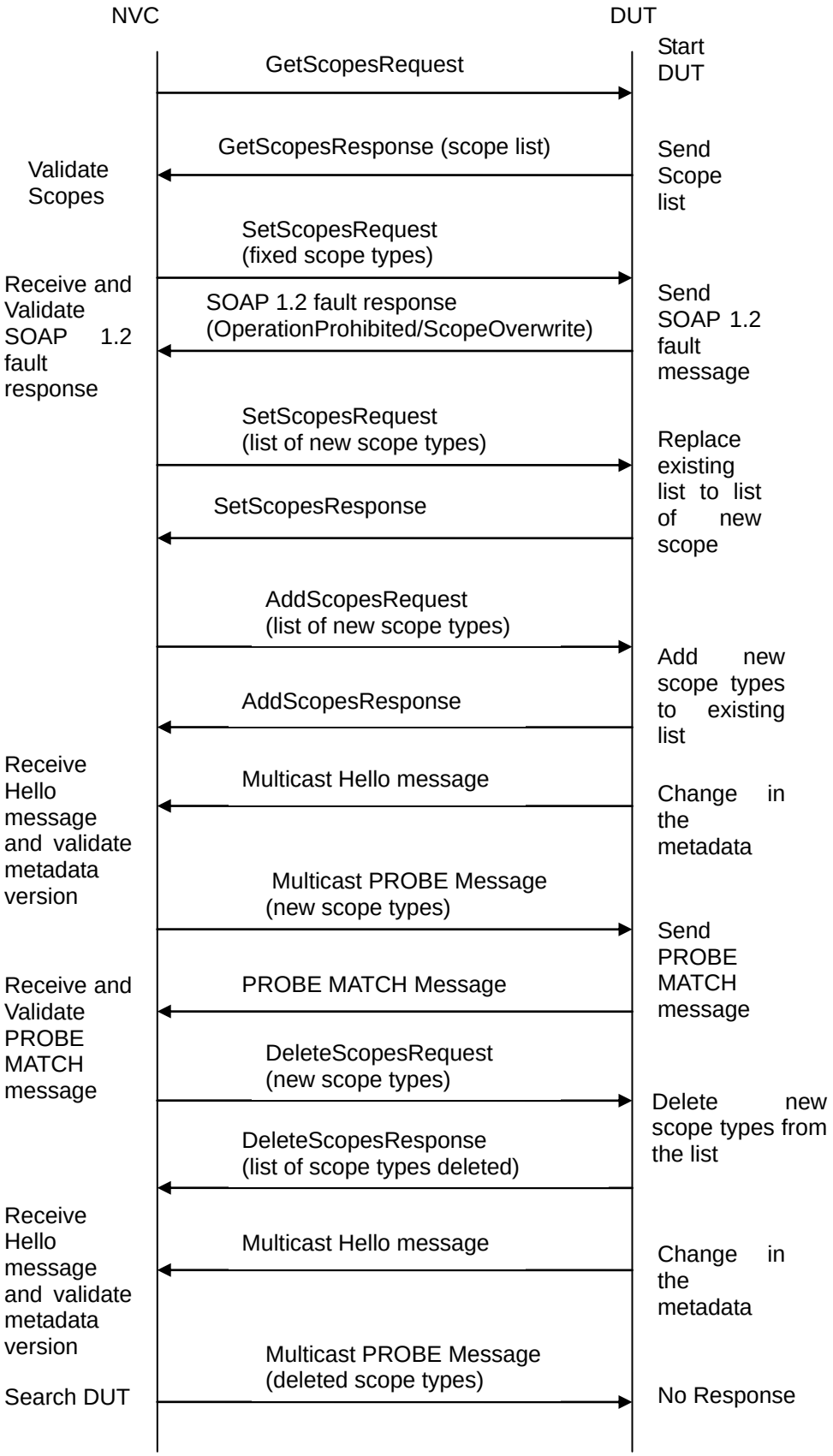
Requirement Level: MUST

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:


NVC
DUT
GetScopesRequest
Start DUT
GetScopesResponse (scope list)
Send Scope list
Validate Scopes
SetScopesRequest (fixed scope types)
Receive and Validate SOAP 1.2 fault response
SOAP 1.2 fault response (OperationProhibited/ScopeOverwrite)
Send SOAP 1.2 fault message
SetScopesRequest (list of new scope types)
Replace existing list to list of new scope
SetScopesResponse
AddScopesRequest (list of new scope types)
Add new scope types to existing list
AddScopesResponse
Receive Hello message and validate metadata version
Multicast Hello message
Change in the metadata
Multicast PROBE Message (new scope types)
Send PROBE MATCH message
Receive and Validate PROBE MATCH message
PROBE MATCH Message
DeleteScopesRequest (new scope types)
Delete new scope types from the list
DeleteScopesResponse (list of scope types deleted)
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 NVC.
2. Start the DUT.
3. NVC will invoke GetScopesRequest message to retrieve existing scope types.
4. DUT replies with the list of scopes types in the GetScopesResponse message.
5. NVC will invoke SetScopesRequest message to overwrite existing scope types with new scope types.
6. DUT generates SOAP 1.2 fault message as fixed scope types cannot be overwritten.
7. NVC will invoke SetScopesRequest message to replace existing list to list of new scope.
8. DUT replies with SetScopesResponse message indicating success.
9. NVC will invoke AddScopesRequest message to add new scope types to the existing scope list.
10. DUT replies with AddScopesResponse message indicating success.
11. DUT sends Multicast Hello message to indicate the change in the metadata (i.e. addition of new scope types to the existing list).
12. NVC will invoke Multicast PROBE message to search DUT with newly added scope types.
13. Verify that DUT issued a PROBE MATCH message.
14. NVC will invoke DeleteScopesRequest message to delete the newly configured scope types.
15. DUT replies with DeleteScopesResponse message indicating success.
16. DUT sends Multicast Hello message to indicate the change in the metadata (i.e. deletion of scope types from the existing list).
17. NVC will invoke Multicast PROBE message to search DUT with deleted scope types.
18. Verify that the DUT did not send PROBE MATCH message.

Test Result:

PASS –

DUT passes 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 SOAP 1.2 fault message (OperationProhibited/ScopeOverwrite) after executing Test Procedure 5.

The DUT did not send SetScopesResponse message after executing Test Procedure 7.

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).

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 DeleteScopesResponse message.

The DUT did not send PROBE MATCH message within the time out period of APP_MAX_DELAY after executing Test Procedure steps 12 and 17.

Note:

It may be possible that DUT may return SOAP Fault 1.2 “TooManyScopes” for the SetScopes (step 7) or AddScopes (step 9) 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 APP_MAX_DELAY.

5.8 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

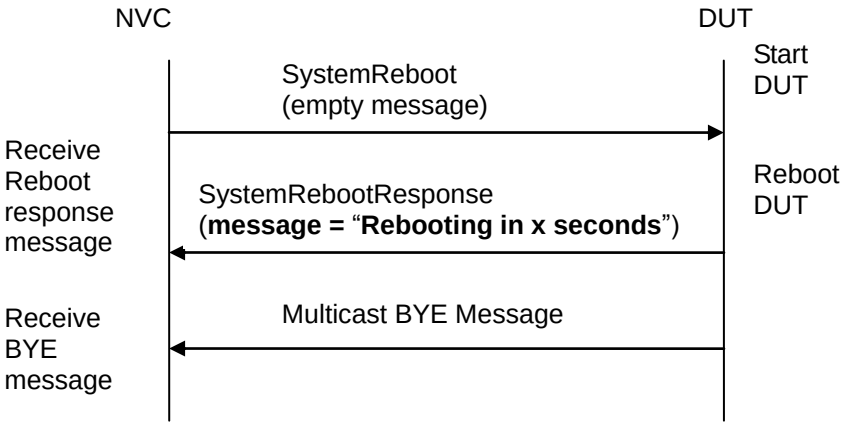
Requirement Level: SHOULD

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

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

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send SystemRebootResponse message.

The DUT did not send BYE message.

5.9 DISCOVERY MODE CONFIGURATION

Test Label: Device Discovery 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

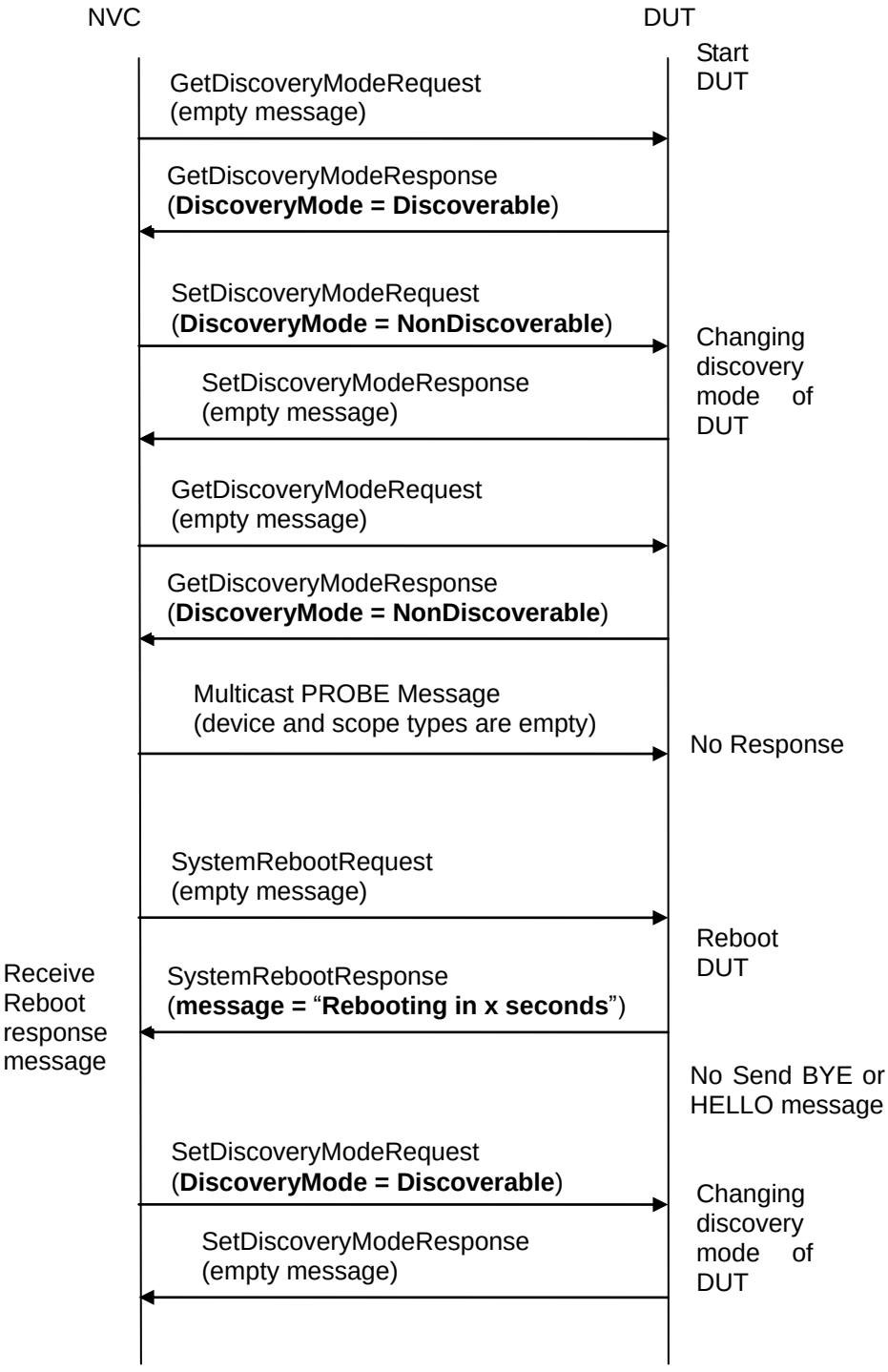
Requirement Level: MUST

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke GetDiscoveryMode message to verify discovery mode of the DUT.
4. Verify that DUT sends GetDiscoveryModeResponse message (DiscoveryMode DiscoveryMode = Discoverable).
5. NVC will invoke SetDiscoveryMode message to set discovery mode of the DUT to Non-Discoverable.
6. Verify that DUT sends SetDiscoveryModeResponse message.
7. NVC will invoke GetDiscoveryMode message to verify discovery mode of the DUT.
8. Verify that DUT sends GetDiscoveryModeResponse message (DiscoveryMode DiscoveryMode = NonDiscoverable).
9. NVC 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. NVC will invoke SystemReboot message to reboot the DUT.
12. Verify that DUT sends SystemRebootResponse message (example message string = “Rebooting in x seconds”).
13. Verify that the DUT did not send BYE or HELLO message.
14. NVC waits for the user defined boot time before proceeding to execute next step.
15. NVC will invoke SetDiscoveryMode message to set discovery mode of the DUT to Discoverable.
16. Verify that DUT sends SetDiscoveryModeResponse message.

Test Result:

PASS –

DUT passes 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 APP_MAX_DELAY

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 APP_MAX_DELAY.

5.10 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

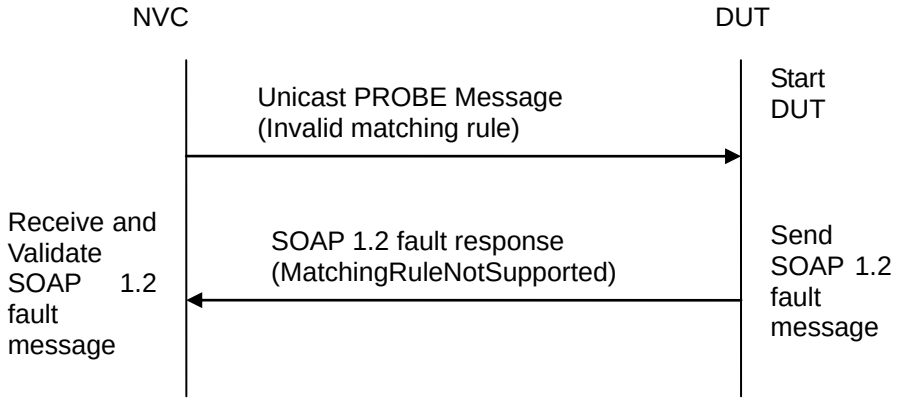
Requirement Level: OPTIONAL

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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 –

DUT passes 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.

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

Requirement Level: MUST

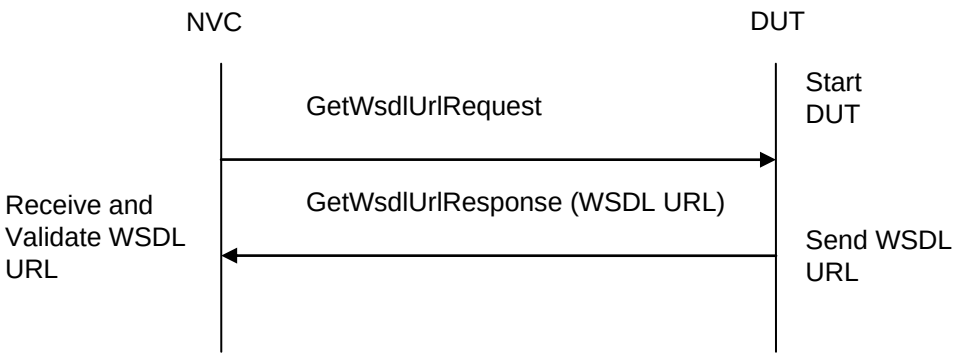
WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: None.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.

3. NVC 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 –

DUT passes 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

Requirement Level: MUST

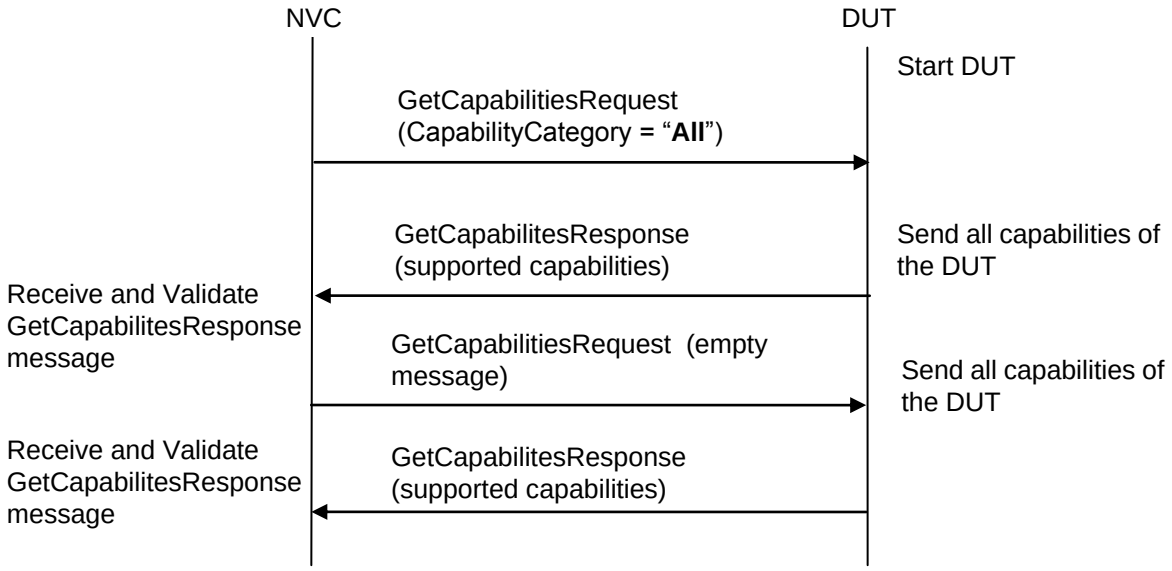
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify all Capabilities of the DUT.

Pre-Requisite: None.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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, Media and Events capabilities.
5. NVC will invoke GetCapabilitiesRequest message (empty message) to retrieve all capabilities of the DUT.
6. Verify the Capabilities Response from the DUT and support for Device, Media and Events capabilities

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send GetCapabilitesResponse message at step 4 and step 5.

The DUT did not support Device, Media and Events capabilities.

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.

Requirement Level: MUST

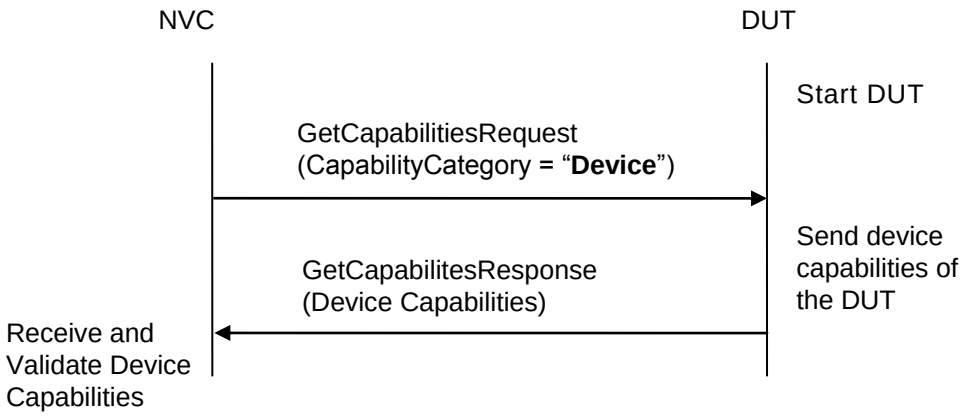
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify Device Capabilities of the DUT.

Pre-Requisite: None.

Test Configuration: NVC and DUT.

Test Sequence:



Test Procedure:

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

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send GetCapabilitesResponse 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

Requirement Level: MUST

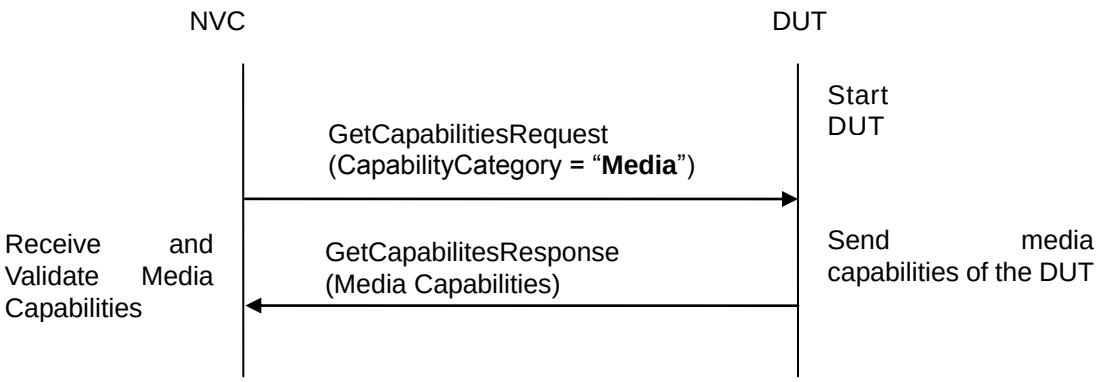
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify Media Capabilities of the DUT.

Pre-Requisite: None.

Test Configuration: NVC and DUT.

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke GetCapabilitiesRequest message (CapabilityCategory = "Media") to retrieve Media Capabilities of the DUT.
4. DUT sends its media capabilities in the GetCapabilitesResponse message.
5. Verify the address of the media service in the GetCapabilitesResponse message.
6. Verify Real time streaming capabilities if supported by the DUT.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send GetCapabilitesResponse message.

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

6.1.5 EVENT CAPABALITIES

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

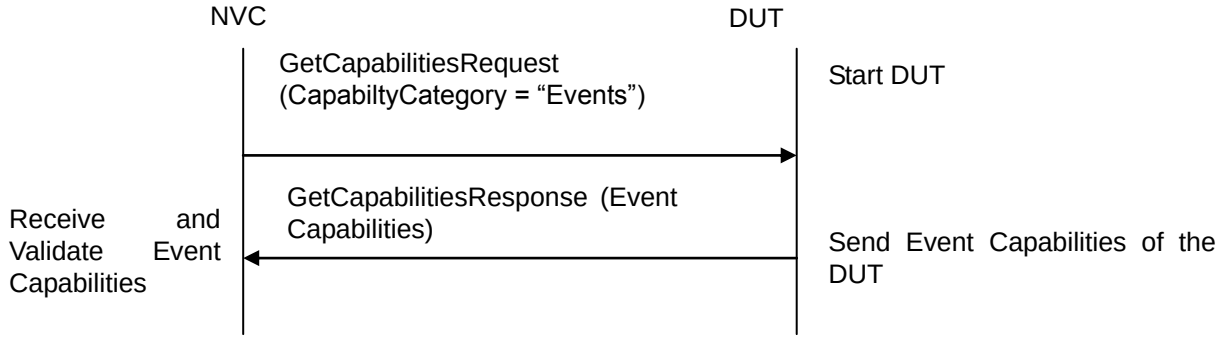
Requirement Level: MUST.

Test Purpose: To verify event capabilities of the DUT.

Pre-Requisite: None.

Test Configuration: NVC and DUT.

Test Sequence:



Test Procedure:

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

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send GetCapabilitesResponse message.

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

6.1.6 PTZ CAPABALITIES

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

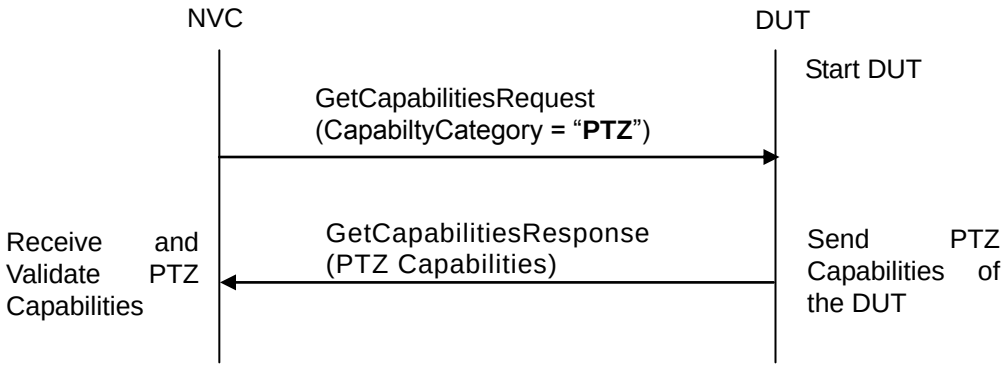
Requirement Level: MUST

Test Purpose: To verify PTZ capabilities of the DUT.

Pre-Requisite: None

Test Configuration: NVC and DUT.

Test Sequence:



Test Procedure:

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

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send GetCapabilitesResponse 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, if the PTZ is not supported.

6.1.7 SERVICE CATEGORY CAPABILITIES

Test Case ID: DEVICE-1-1-7

Note: DUT Service Category Capabilities Test to be repeated for Analytics and Imaging service

If a specific service category is not supported by the DUT, it MUST generate SOAP 1.2 fault response (ActionNotSupported/NoSuchService).

6.1.8 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.

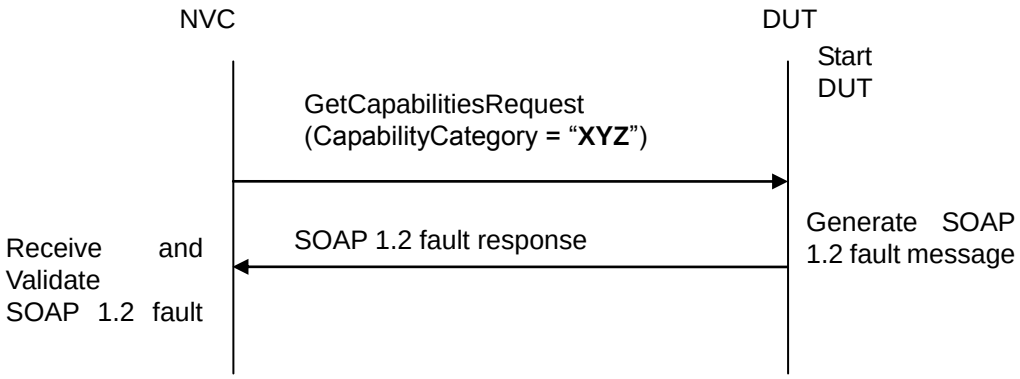
Requirement Level: SHOULD.

WSDL Reference: devicemgmt.wsdl

Pre-Requisite: None

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

Test Configuration: NVC and DUT.



Test Procedure:

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

Test Result:

PASS –

DUT passes 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.

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

Requirement Level: MUST

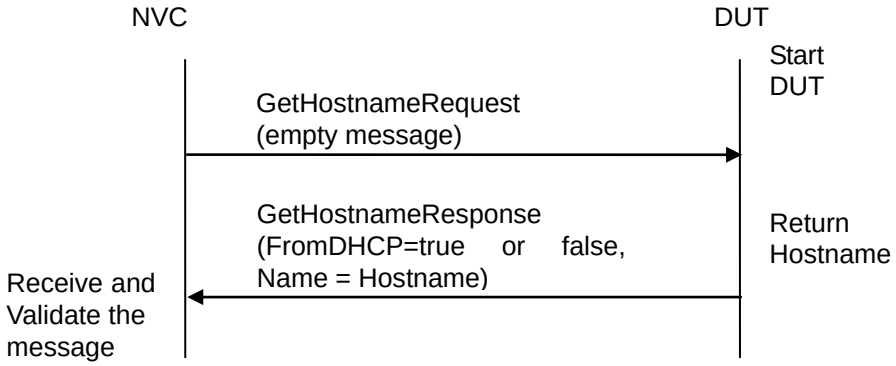
WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: None.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

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

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send GetHostnameResponse message.

6.2.2 NETWORK COMMAND SETHOSTNAME TEST

Test Label: Device Management Network Command **SetHostname** Test.

Test Case ID: DEVICE-2-1-2

ONVIF Core Specification Coverage: Set hostname

Command Under Test: SetHostname

Requirement Level: MUST

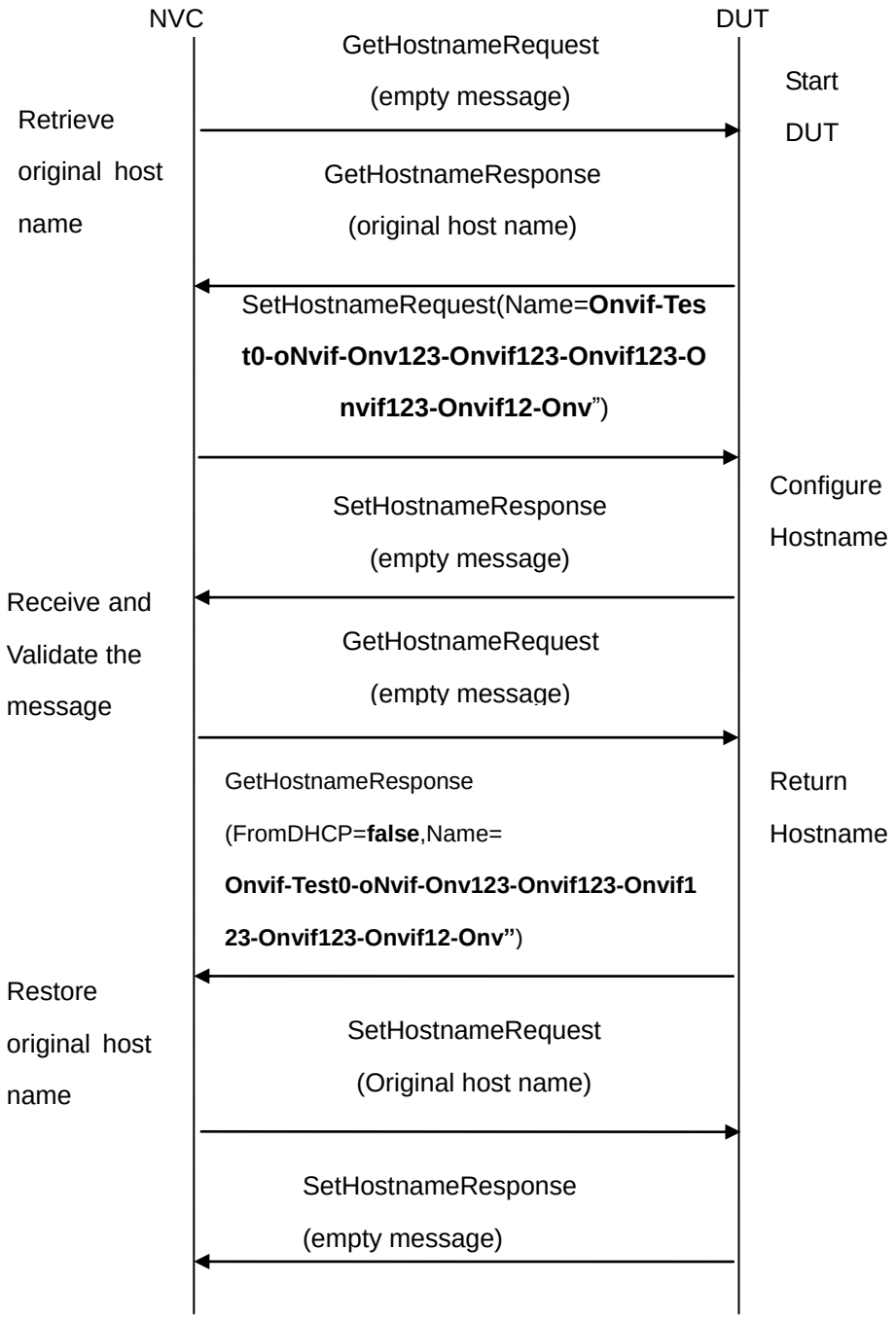
WSDL Reference: devicemgmt.wsdl

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

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

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC invokes GetHostnameRequest to retrieve original settings of DUT.

4. NVC will invoke SetHostnameRequest message (Name =" Onvif-Test0-oNvif-Onv123-Onvif123-Onvif123-Onvif123-Onvif12-Onv", whose length is equal to 63 bytes) to configure the hostname.
5. Verify that DUT sends SetHostnameResponse (empty message).
6. Verify the hostname configurations in DUT through GetHostnameRequest.
7. DUT sends hostname configuration in the GetHostnameResponse message (FromDHCP = false, Name = "Onvif-Test0-oNvif-Onv123-Onvif123-Onvif123-Onvif123-Onvif12-Onv").

13. NVC invokes SetHostnameRequest to restore original settings of DUT.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send SetHostnameResponse message.

The DUT did not send GetHostnameResponse message.

The DUT did not send correct hostname (i.e. "Onvif-Test0-oNvif-Onv123-Onvif123-Onvif123-Onvif123-Onvif12-Onv") in the GetHostnameResponse message.

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

6.2.3 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.

Requirement Level: SHOULD

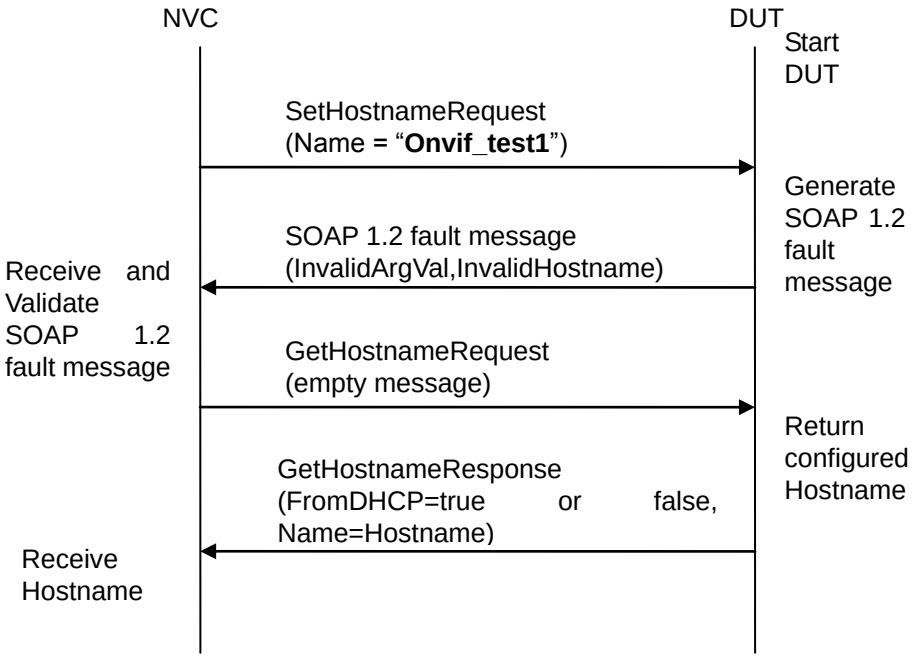
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify behaviour 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: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke SetHostnameRequest message (Name = "Onvif_test1") to configure the hostname.
4. Verify that DUT generates SOAP 1.2 fault message (InvalidArgVal/InvalidHostname).
5. Verify hostname from DUT through GetHostnameRequest.
6. DUT sends valid hostname in GetHostnameResponse message

   (FromDHCP=true or false, Name=Hostname).

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send SOAP 1.2 fault message.

The DUT did not send 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.

6.2.4 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

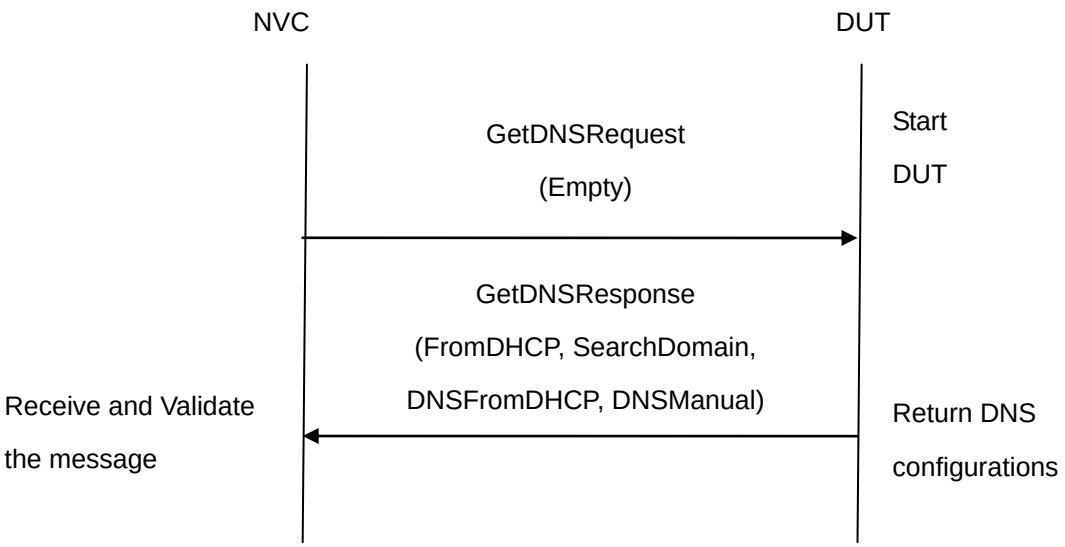
Requirement Level: MUST

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke GetDNSRequest message to retrieve DNS configurations of the DUT.

4. Verify the GetDNSResponse from 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 –

DUT passes 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 empty IP address.

6.2.5 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

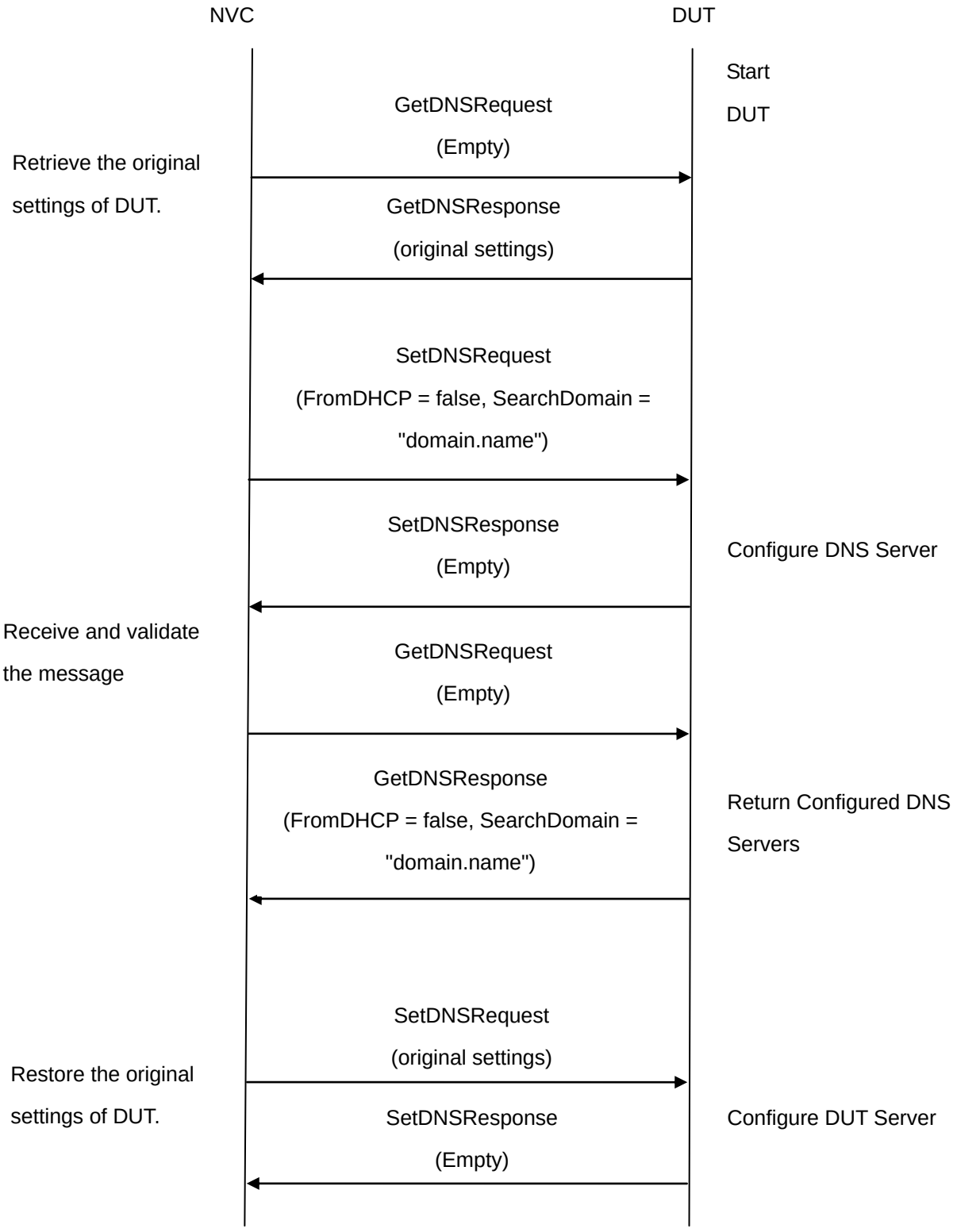
Requirement Level: MUST

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:


NVC
DUT
Start
DUT
GetDNSRequest
(Empty)
Retrieve the original
settings of DUT.
GetDNSResponse
(original settings)
SetDNSRequest
(FromDHCP = false, SearchDomain =
"domain.name")
SetDNSResponse
(Empty)
Configure DNS Server
Receive and validate
the message
GetDNSRequest
(Empty)
GetDNSResponse
(FromDHCP = false, SearchDomain =
"domain.name")
Return Configured DNS
Servers
SetDNSRequest
(original settings)
Restore the original
settings of DUT.
SetDNSResponse
(Empty)
Configure DUT Server

Test Procedure:

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

Test Result:

PASS –

DUT passes 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.

6.2.6 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

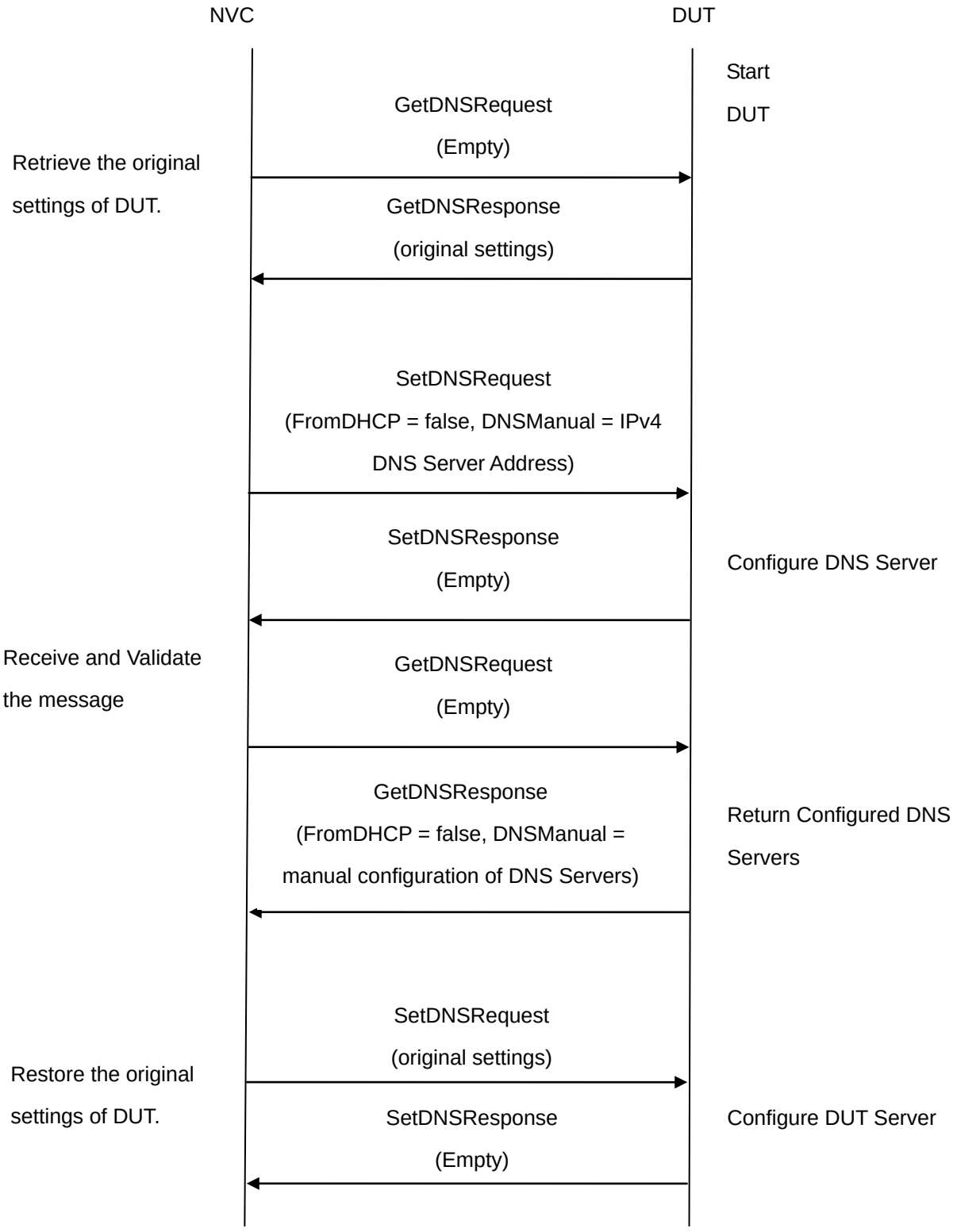
Requirement Level: MUST

Test Purpose: To configure IPv4 DNS server address setting in DUT through SetDNS command.

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:


NVC
DUT
Start
DUT
GetDNSRequest
(Empty)
Retrieve the original
settings of DUT.
GetDNSResponse
(original settings)
SetDNSRequest
(FromDHCP = false, DNSManual = IPv4
DNS Server Address)
SetDNSResponse
(Empty)
Configure DNS Server
Receive and Validate
the message
GetDNSRequest
(Empty)
GetDNSResponse
(FromDHCP = false, DNSManual =
manual configuration of DNS Servers)
Return Configured DNS
Servers
SetDNSRequest
(original settings)
Restore the original
settings of DUT.
SetDNSResponse
(Empty)
Configure DUT Server

Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke GetDNSRequest message to retrieve the original settings of DUT.
4. NVC 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. DUT sends its DNS configurations in the GetDNSResponse message (DNSInformation[FromDHCP = false, DNSManual = "IPv4", "10.1.1.1"]).
8. NVC will invoke SetDNSRequest message to restore the original settings of DUT.

Test Result:

PASS –

DUT passes 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.

6.2.7 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

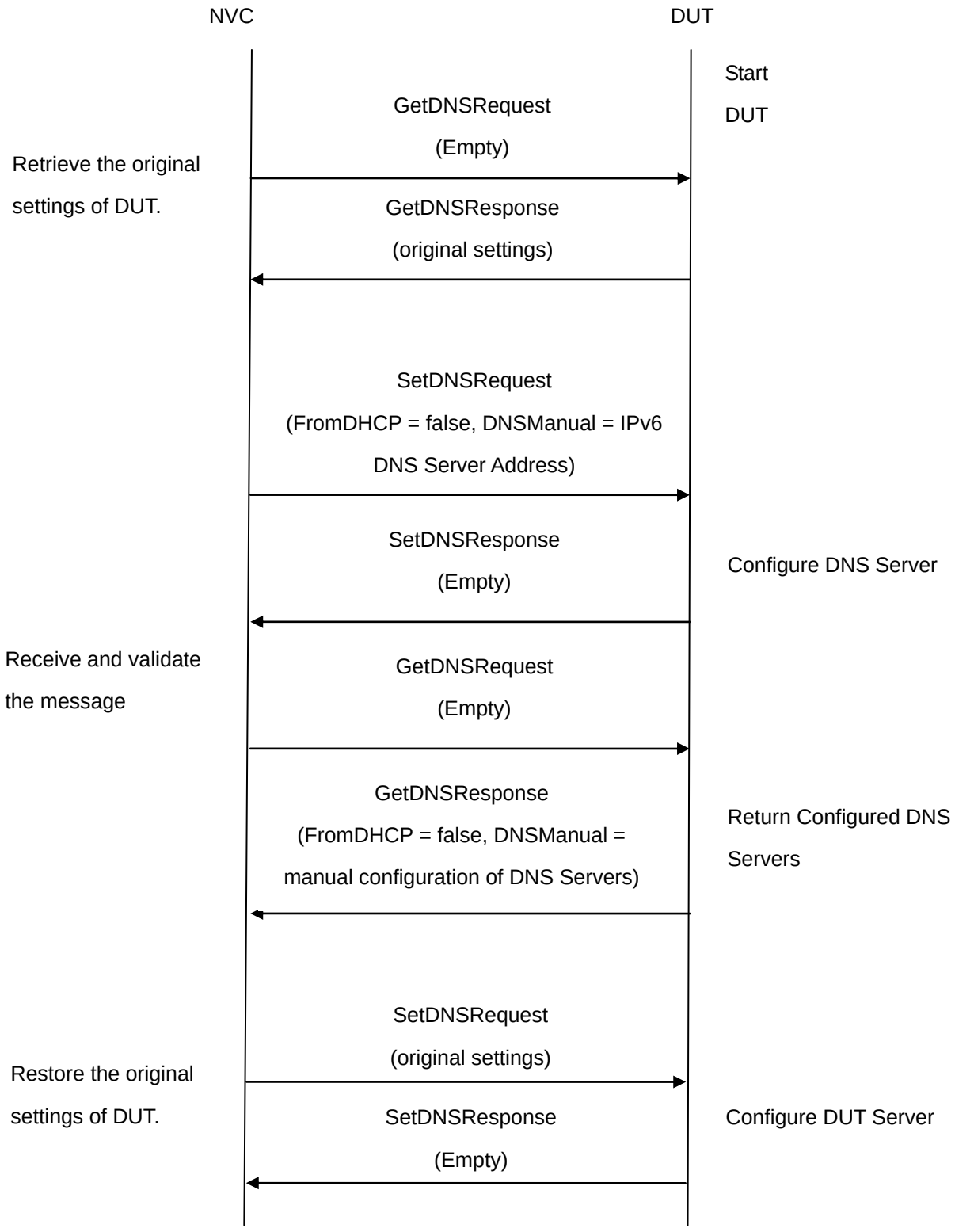
Requirement Level: MUST IF IMPLEMENTED (IPv6)

Test Purpose: To configure IPv6 DNS server address setting in DUT through SetDNS command.

Pre-Requisite: IPv6 is implemented by DUT.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke GetDNSRequest message to retrieve the original settings of DUT.
4. NVC 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 in DUT through GetDNSRequest.
7. DUT sends its DNS configurations in the GetDNSResponse message (DNSInformation[FromDHCP = false, DNSManual = "IPv6", "2001:1:1:1:1:1:1:1"]).
8. NVC will invoke SetDNSRequest message to restore the original settings of DUT.

Test Result:

PASS –

DUT passes 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.

6.2.8 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

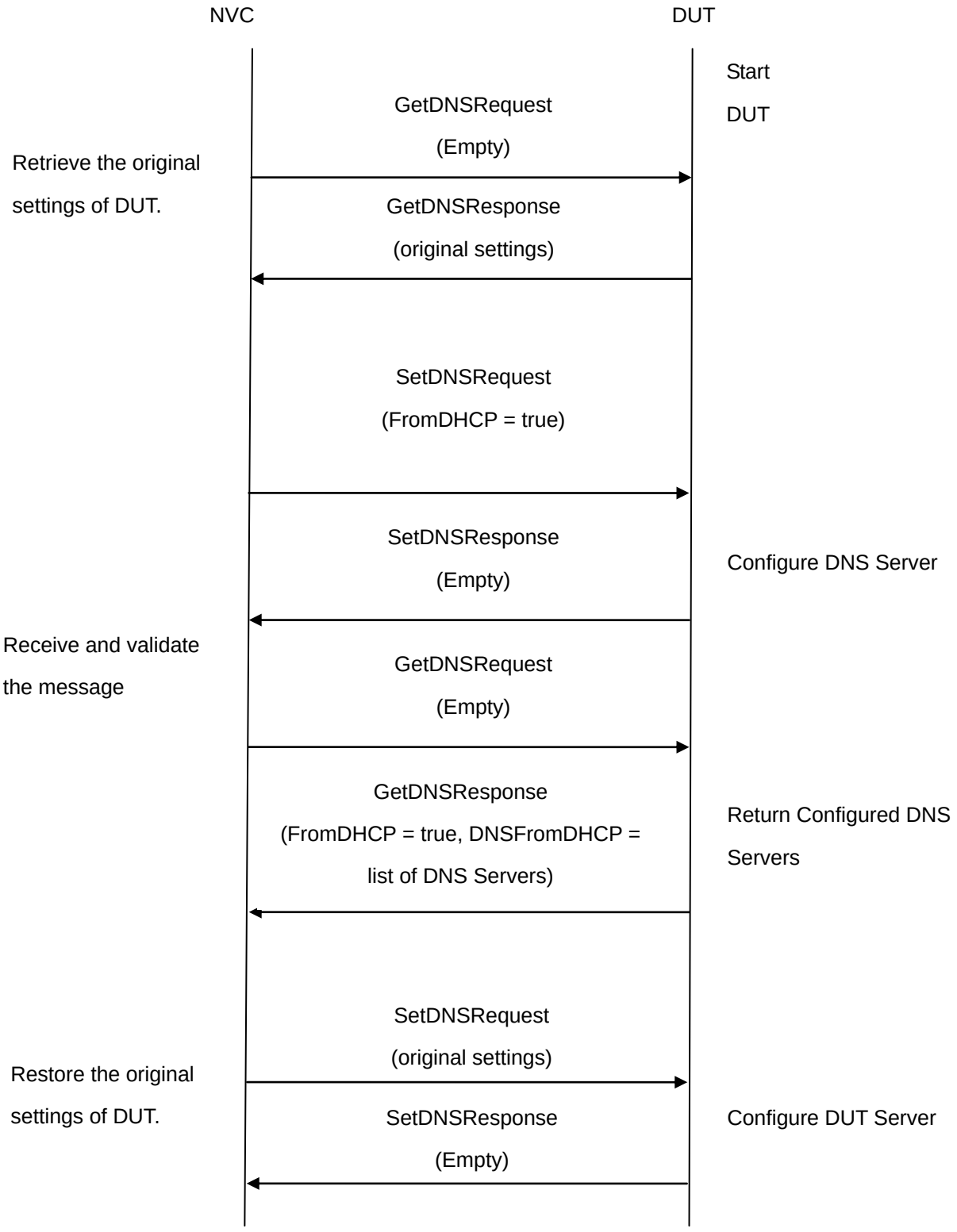
Requirement Level: MUST

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:


NVC
DUT
Start
DUT
GetDNSRequest
(Empty)
Retrieve the original
settings of DUT.
GetDNSResponse
(original settings)
SetDNSRequest
(FromDHCP = true)
SetDNSResponse
(Empty)
Configure DNS Server
Receive and validate
the message
GetDNSRequest
(Empty)
GetDNSResponse
(FromDHCP = true, DNSFromDHCP =
list of DNS Servers)
Return Configured DNS
Servers
SetDNSRequest
(original settings)
Restore the original
settings of DUT.
SetDNSResponse
(Empty)
Configure DUT Server

Test Procedure:

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

Test Result:

PASS –

DUT passes 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]) in the GetDNSResponse message.

6.2.9 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

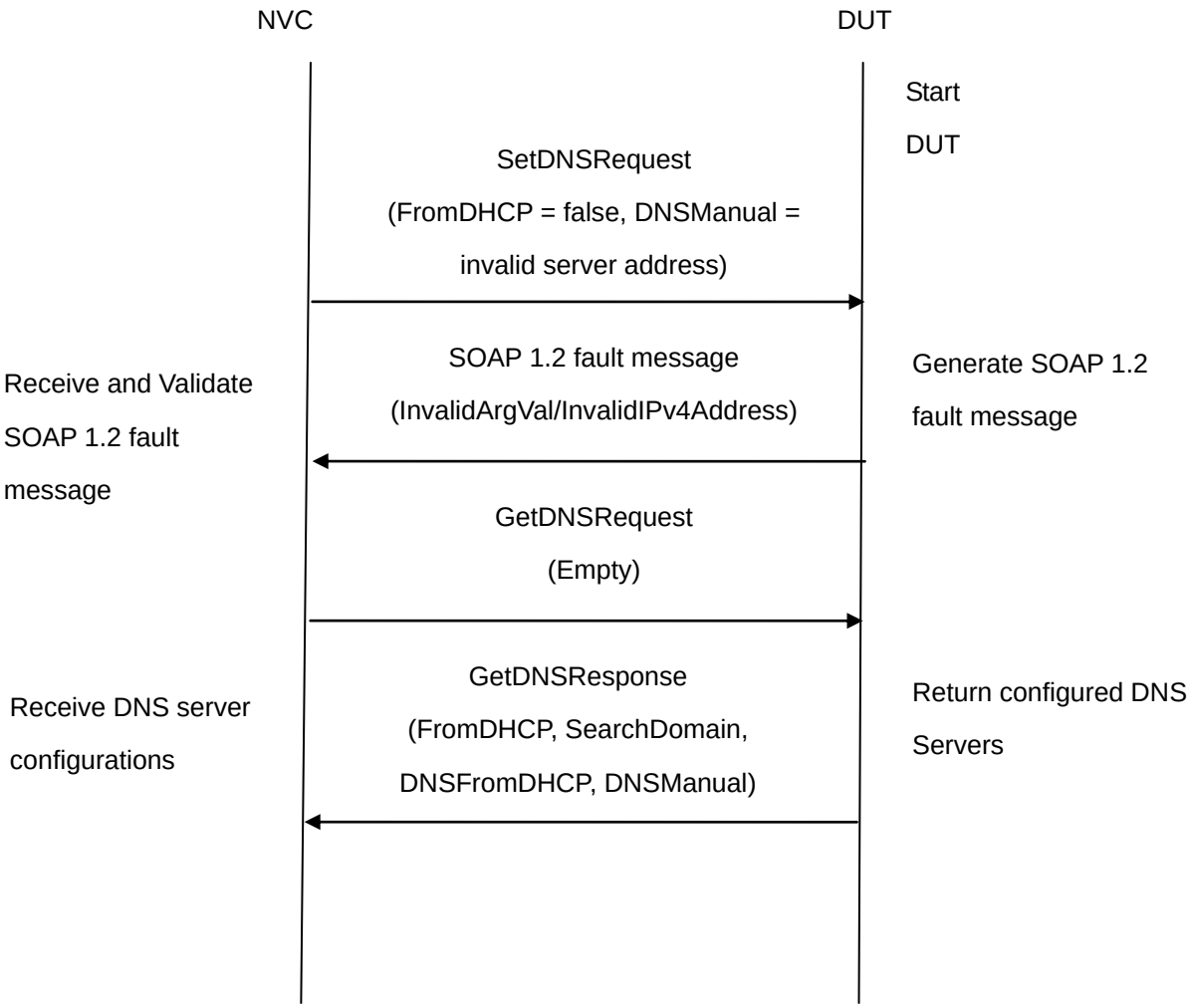
Requirement Level: SHOULD

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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 –

DUT passes 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.

6.2.10 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

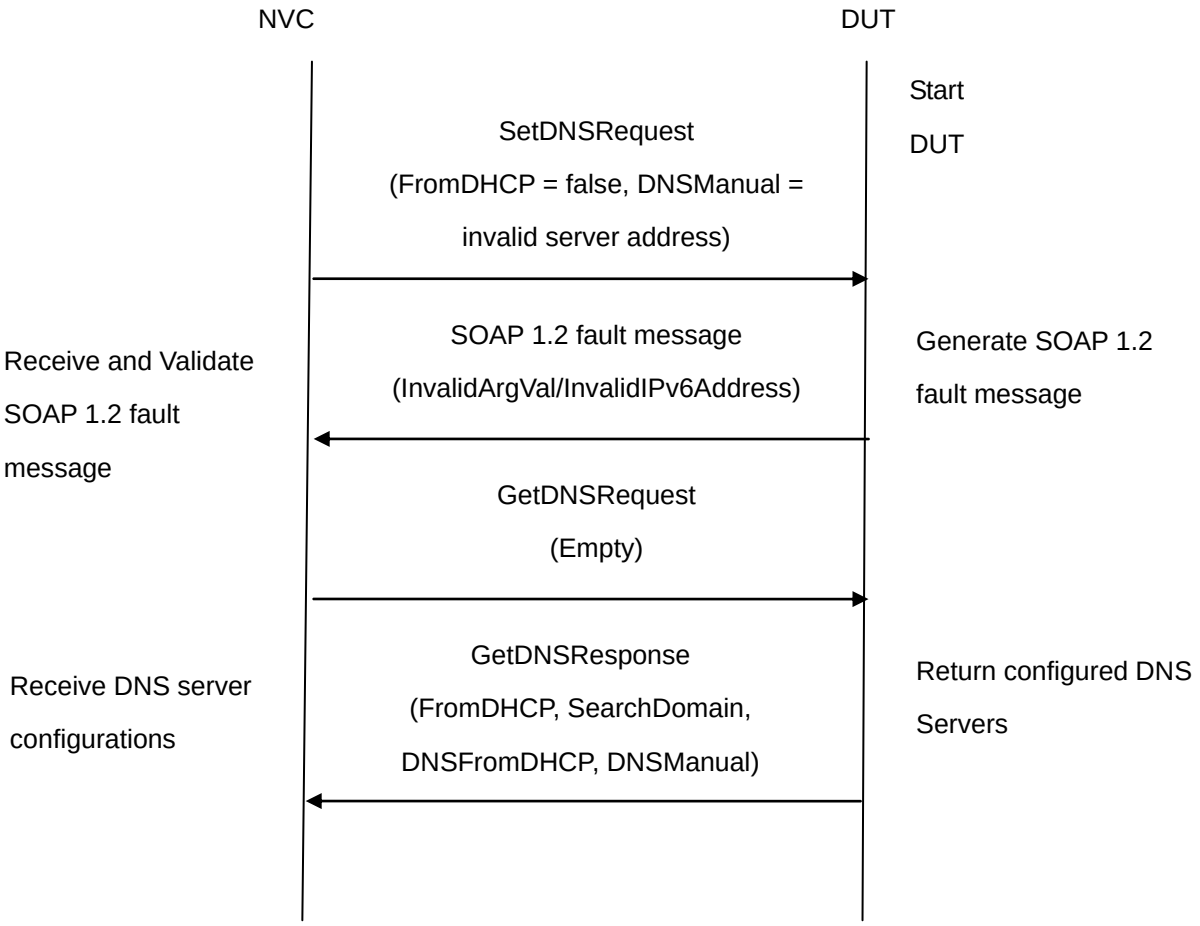
Requirement Level: SHOULD IF IMPLEMENTED (IPv6)

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

Pre-Requisite: IPv6 is implemented by DUT.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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. 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 –

DUT passes 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.

6.2.11 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

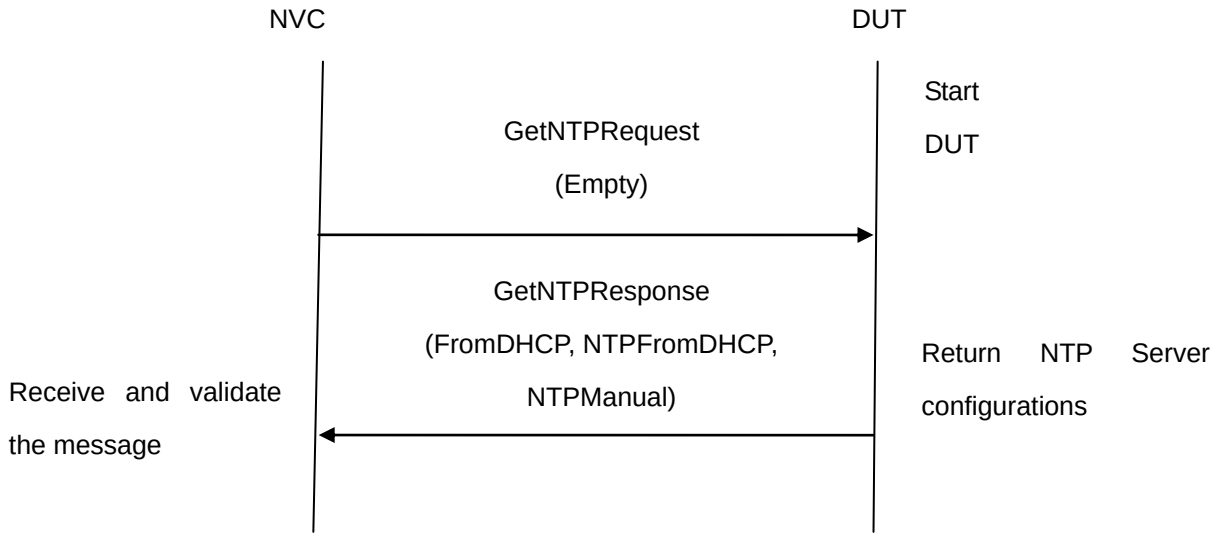
Requirement Level: MUST IF SUPPORTED (NTP)

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

Pre-Requisite: NTP is supported by DUT.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke GetNTPRequest message to retrieve NTP Server settings of the DUT.
4. Verify the GetNTPResponse from DUT (NTPInformation[FromDHCP = true or false, NTPFromDHCP = list of NTP Servers obtained from DHCP, NTPManual = list of NTP Servers manually configured]).

Test Result:

PASS –

DUT passes 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

6.2.12 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

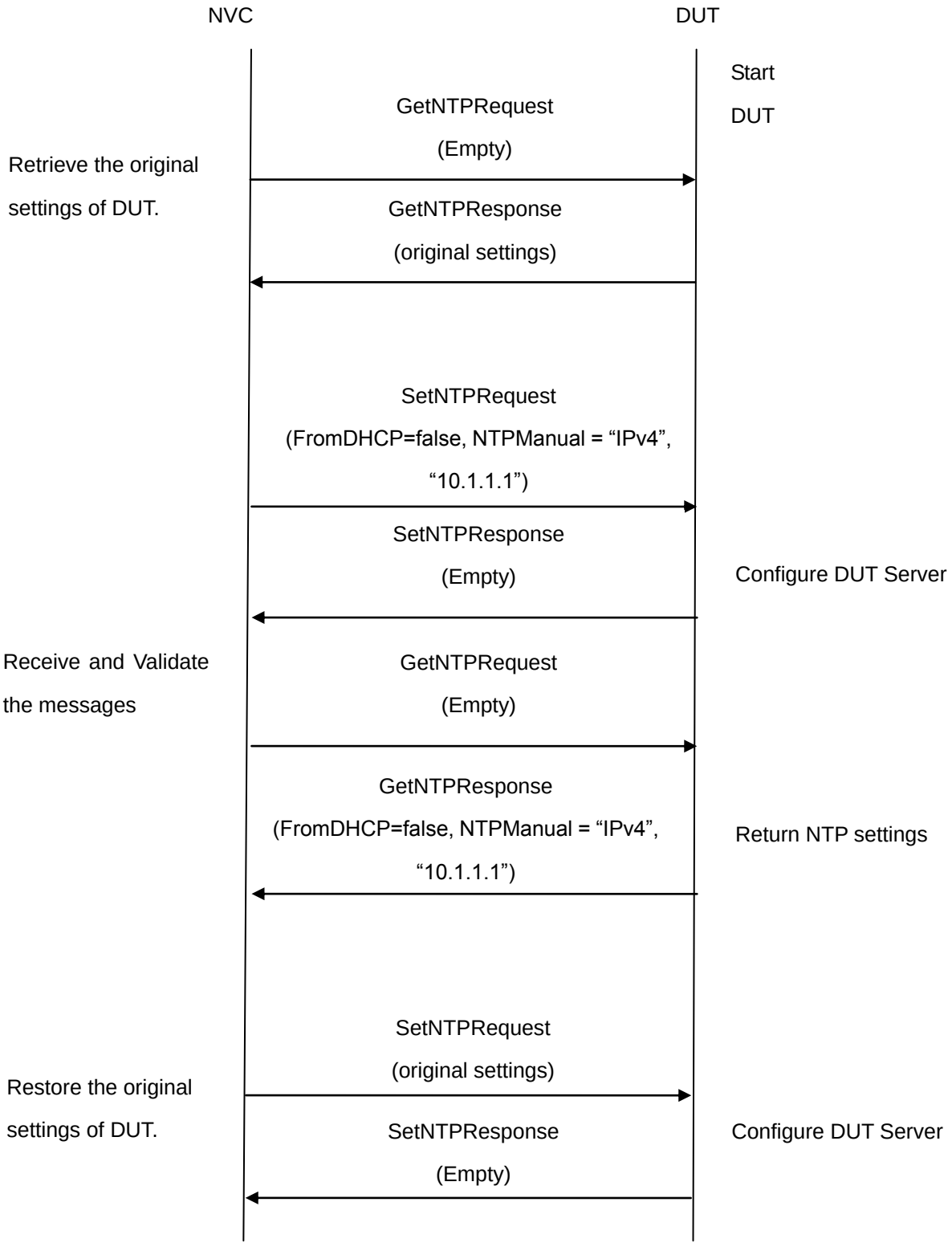
Requirement Level: MUST IF SUPPORTED (NTP)

Test Purpose: To configure NTP IPv4 address settings on an DUT through SetNTP command.

Pre-Requisite: NTP is supported by DUT.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.

2. Start the DUT.
3. NVC will invoke GetNTPRequest message to retrieve the original settings of DUT.
4. NVC 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. NVC will invoke SetNTPRequest message to restore the original settings of DUT.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

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

The DUT did not send GetNTPResponse message in 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 in step-6.

Note: See Annex A.6 for Valid IPv4 Address definition.

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

6.2.13 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

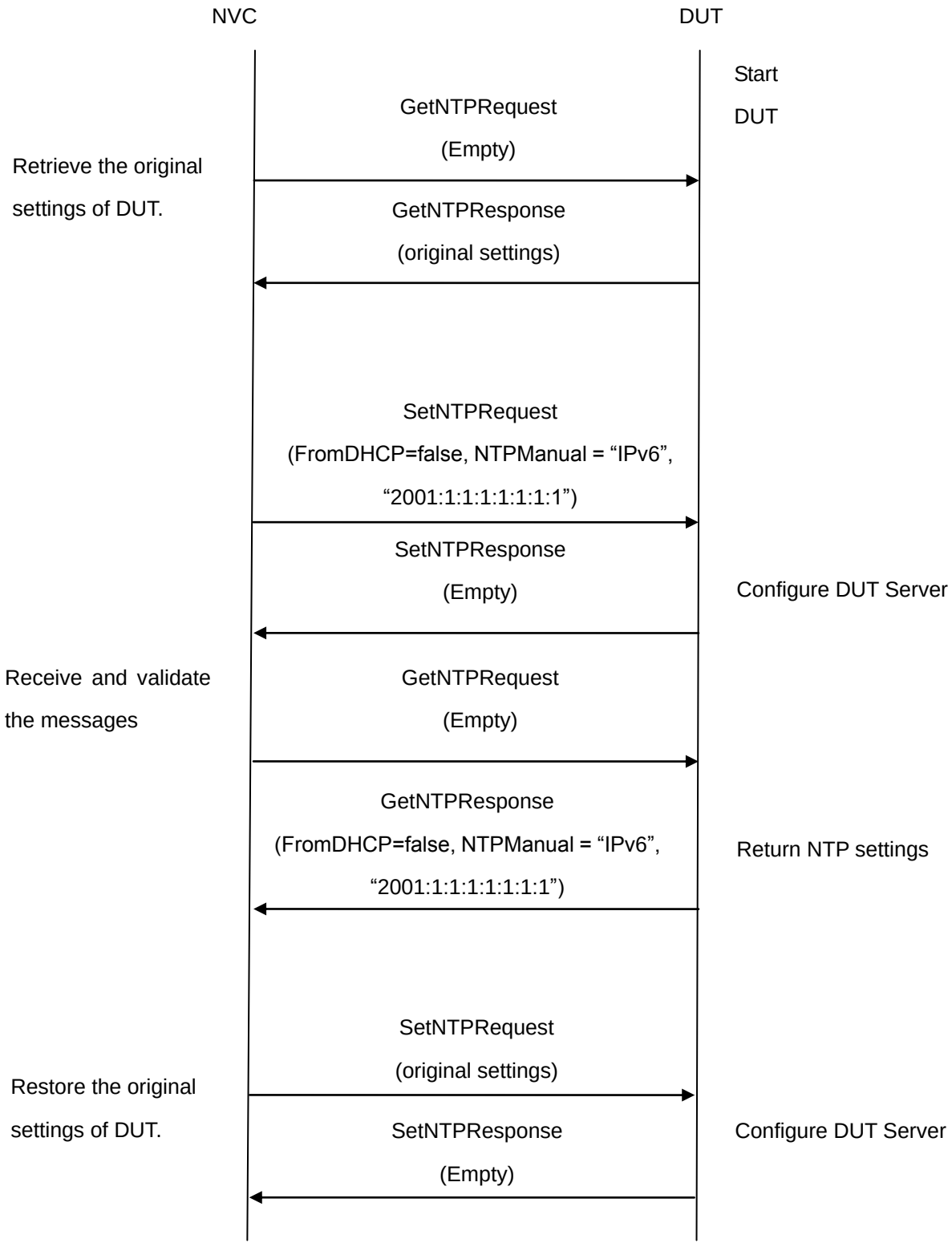
Requirement Level: MUST IF SUPPORTED (NTP) & IMPLEMENTED (IPv6)

Test Purpose: To configure NTP IPv6 address settings on an DUT through SetNTP command.

Pre-Requisite: NTP is supported by DUT. IPv6 is implemented by DUT.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke GetNTPRequest message to retrieve the original settings of DUT.
4. NVC 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. 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. NVC will invoke SetNTPRequest message to restore the original settings of DUT.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

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

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

6.2.14 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

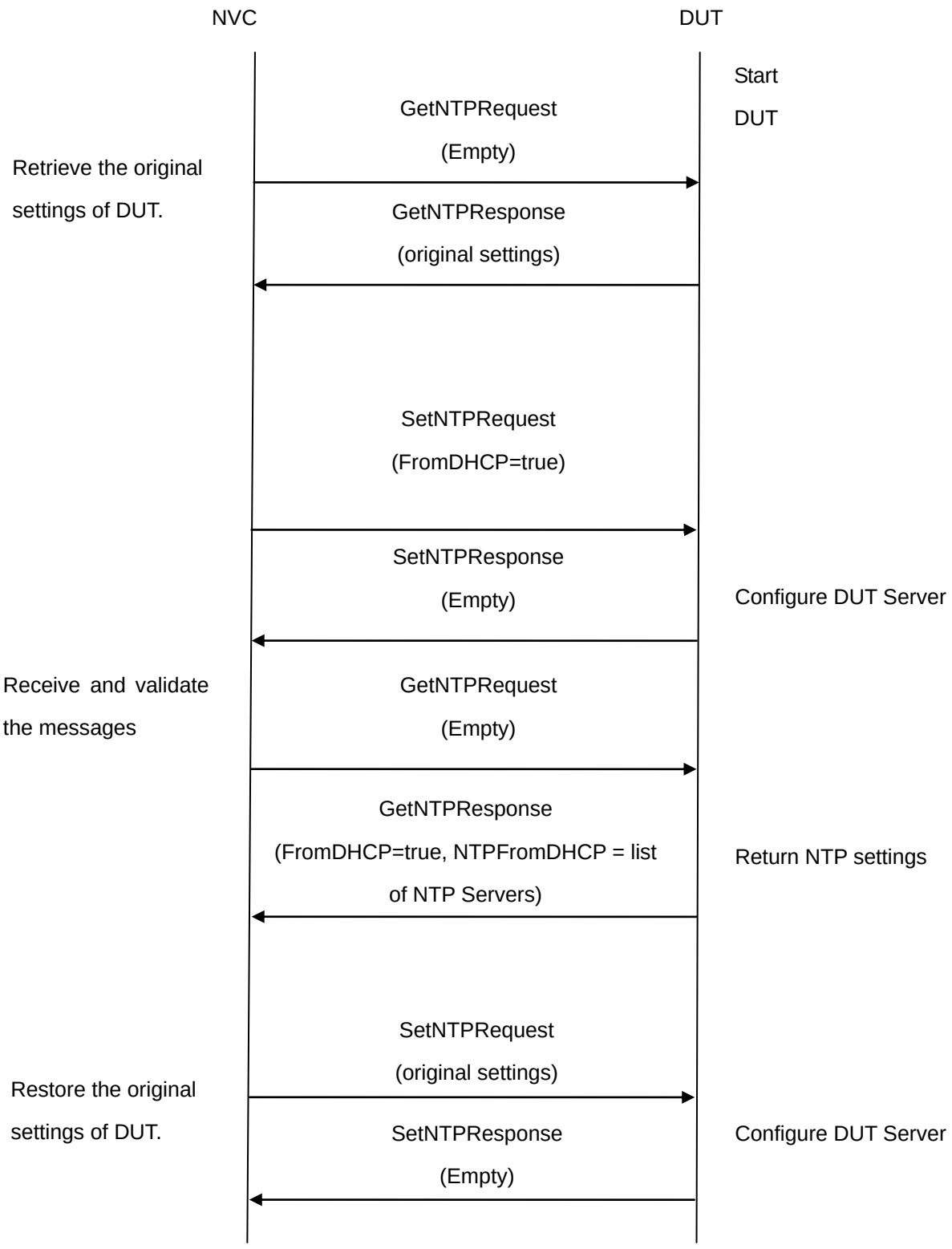
Requirement Level: MUST IF SUPPORTED (NTP)

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

Pre-Requisite: NTP is supported by DUT.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.

2. Start the DUT.
3. NVC will invoke GetNTPRequest message to retrieve the original settings of DUT.
4. NVC 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. DUT sends its NTP Server settings in the GetNTPResponse message (NTPInformation[FromDHCP= true, NTPFromDHCP = list of NTP Servers obtained from DHCP]).
8. NVC will invoke SetNTPRequest message to restore the original settings of DUT.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

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

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

The DUT did not send correct NTP Server information (i.e. NTPInformation[FromDHCP=true, NTPFromDHCP = list of NTP Servers obtained from DHCP]) in GetNTPResponse message in step-7.

6.2.15 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

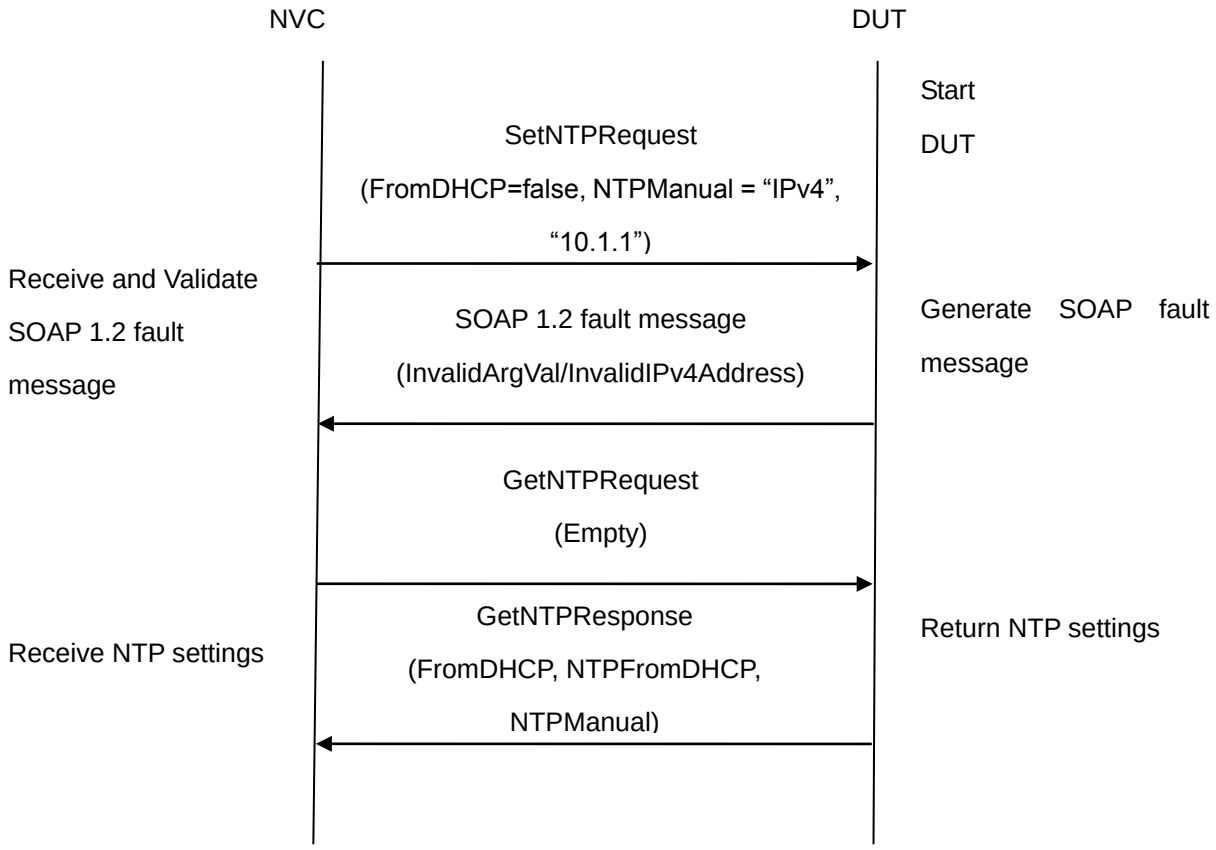
Requirement Level: SHOULD IF SUPPORTED (NTP)

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

Pre-Requisite: NTP is supported by DUT.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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. 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 –

DUT passes 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.

6.2.16 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

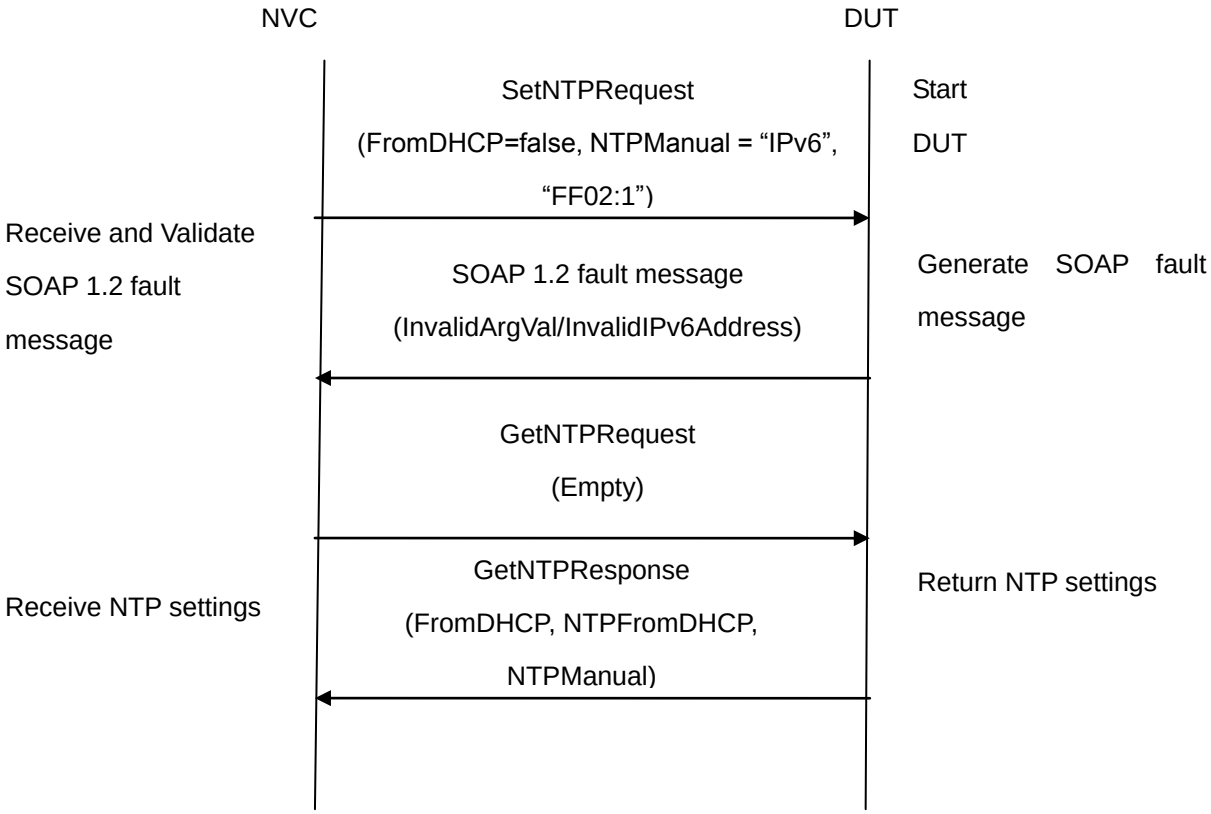
Requirement Level: SHOULD IF SUPPORTED (NTP) & IMPLEMENTED (IPv6)

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

Pre-Requisite: NTP is supported by DUT. IPv6 is implemented by DUT.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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. 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 –

DUT passes 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.

6.2.17 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

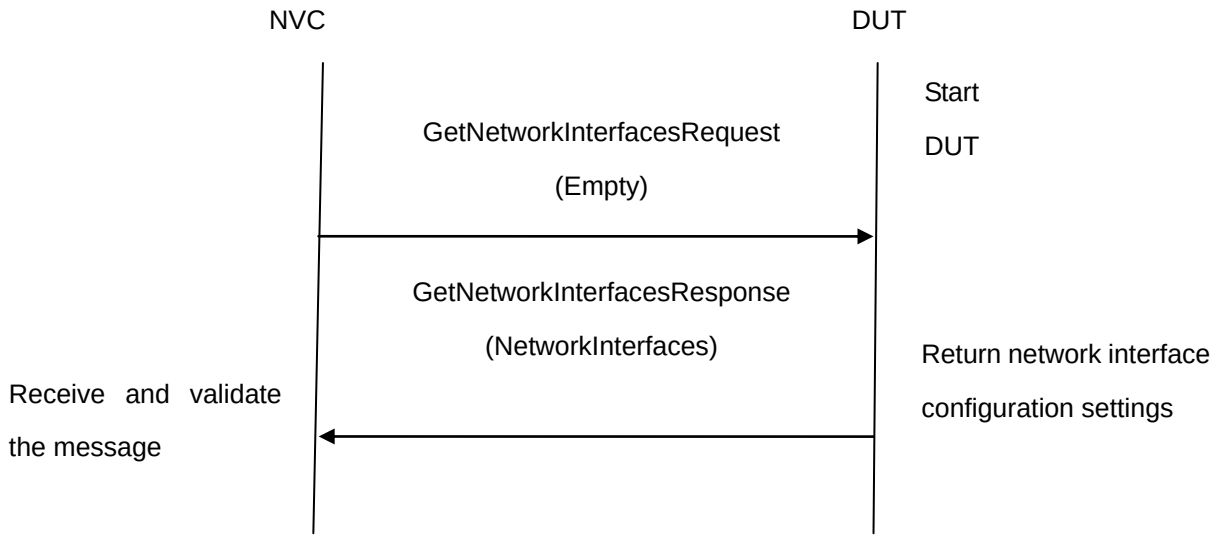
Requirement Level: MUST

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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 –

DUT passes 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.

6.2.18 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

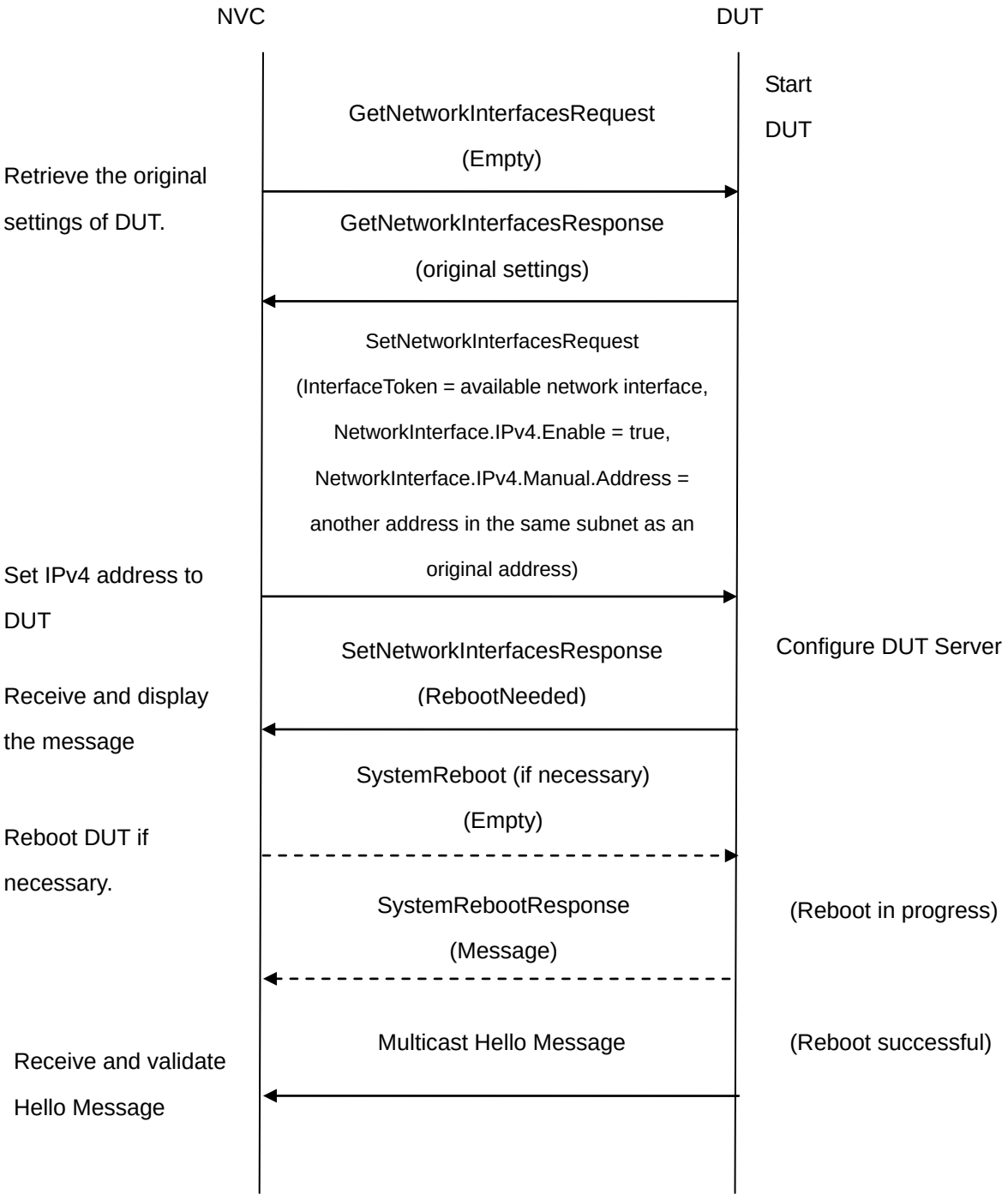
Requirement Level: MUST

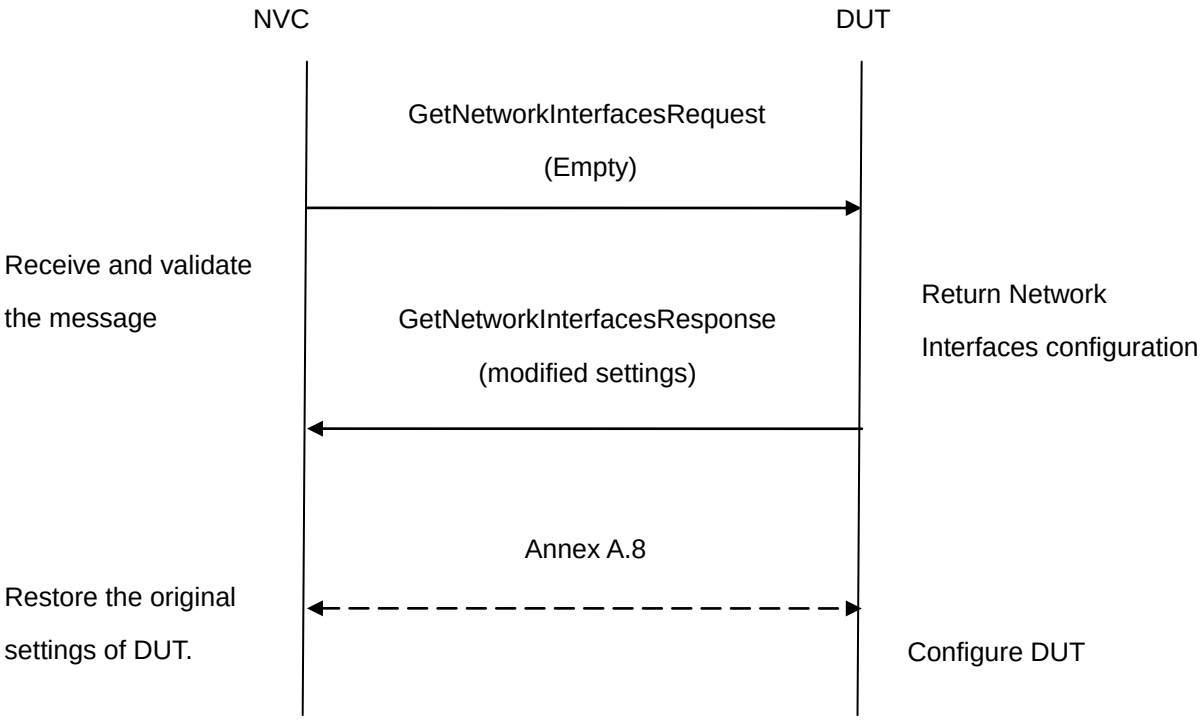
Test Purpose: To configure IPv4 address setting on an DUT through SetNetworkInterfaces command.

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

Test Configuration: NVC and DUT

Test Sequence:


NVC
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 NVC.
2. Start the DUT.
3. NVC will invoke GetNetworkInterfacesRequest message to retrieve the original settings of DUT.
4. NVC 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. DUT will return SetNetworkInterfacesResponse.
6. If necessary, NVC will invoke SystemReboot message to restart DUT. Otherwise, continue to step-8.
7. DUT will return SystemRebootResponse message.
8. DUT will send Multicast Hello message from newly configured address.
9. NVC will receive and validate Hello message sent from newly configured address by DUT.
10. Verify the network interfaces settings in DUT through GetNetworkInterfacesRequest message.
11. 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. NVC will restore the original settings by following the procedure mentioned in Annex A.8.

Test Result:

PASS –

DUT passes 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.

6.2.19 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

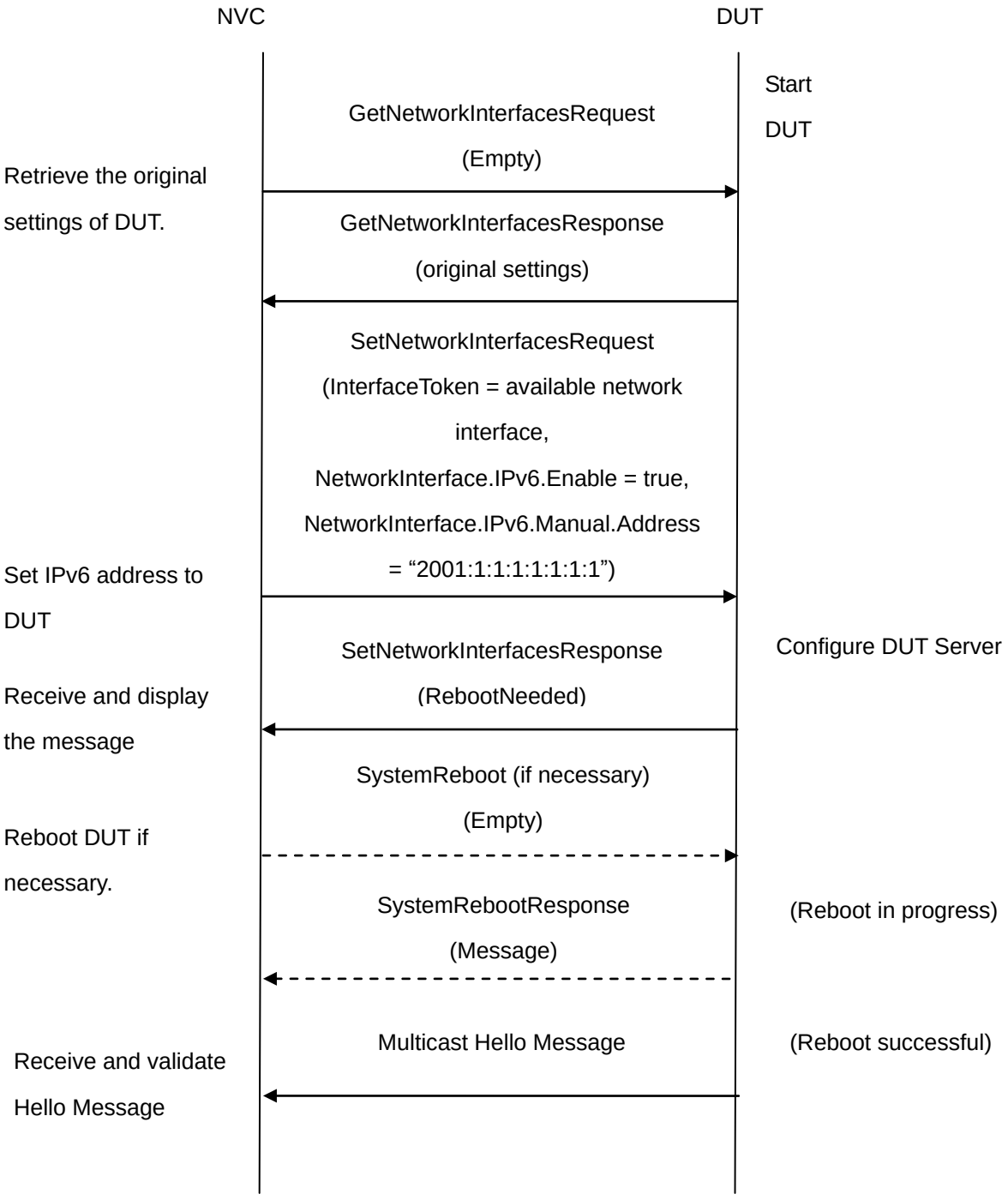
Requirement Level: MUST IF IMPLEMENTED (IPv6)

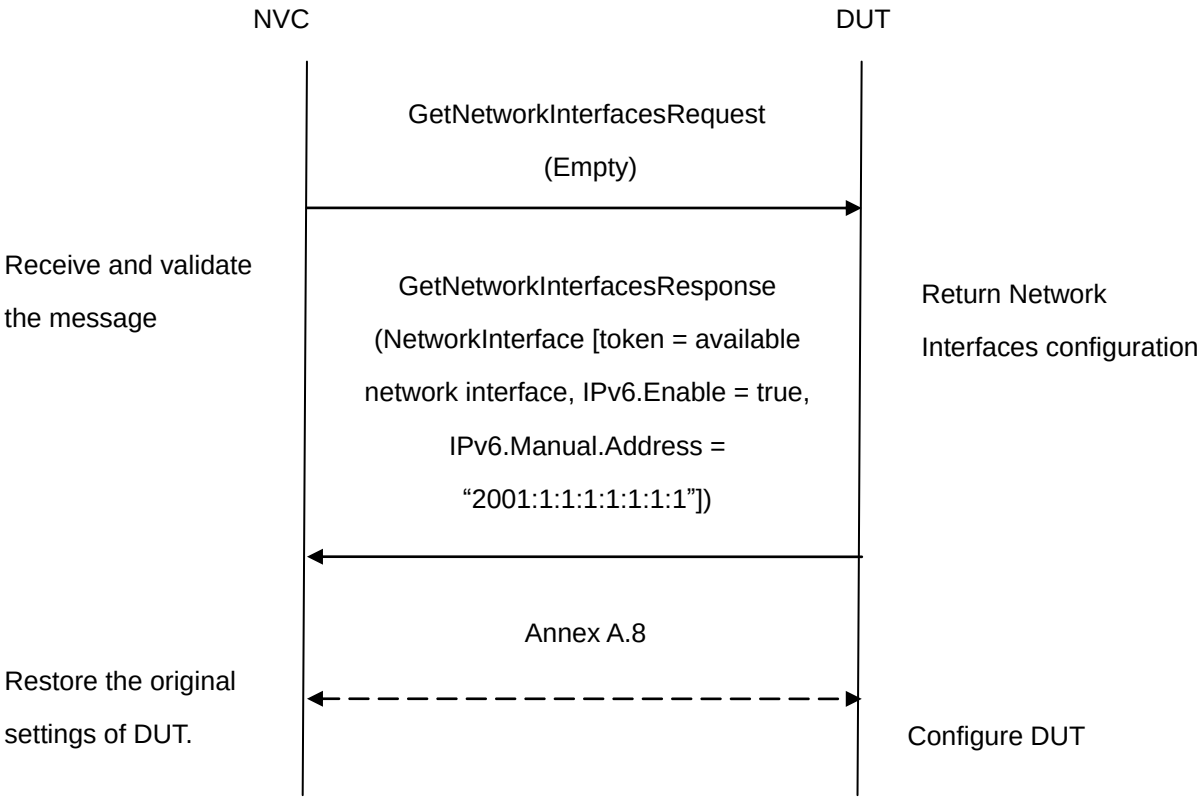
Test Purpose: To configure IPv6 address setting on an DUT through SetNetworkInterfaces command.

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:


NVC
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 NVC.
2. Start the DUT.
3. NVC will invoke GetNetworkInterfacesRequest message to retrieve the original settings of DUT.
4. NVC 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. DUT will return SetNetworkInterfacesResponse.
6. If necessary, NVC will invoke SystemReboot message to restart DUT. Otherwise, continue to step-8.
7. DUT will return SystemRebootResponse message.
8. DUT will send Multicast Hello message.
9. NVC will receive and validate Hello message sent by DUT.
10. Verify the network interfaces settings in DUT through GetNetworkInterfacesRequest message.

11. 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. NVC will restore the original settings by following the procedure mentioned in Annex A.8.

Test Result:

PASS –

DUT passes 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.

6.2.20 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

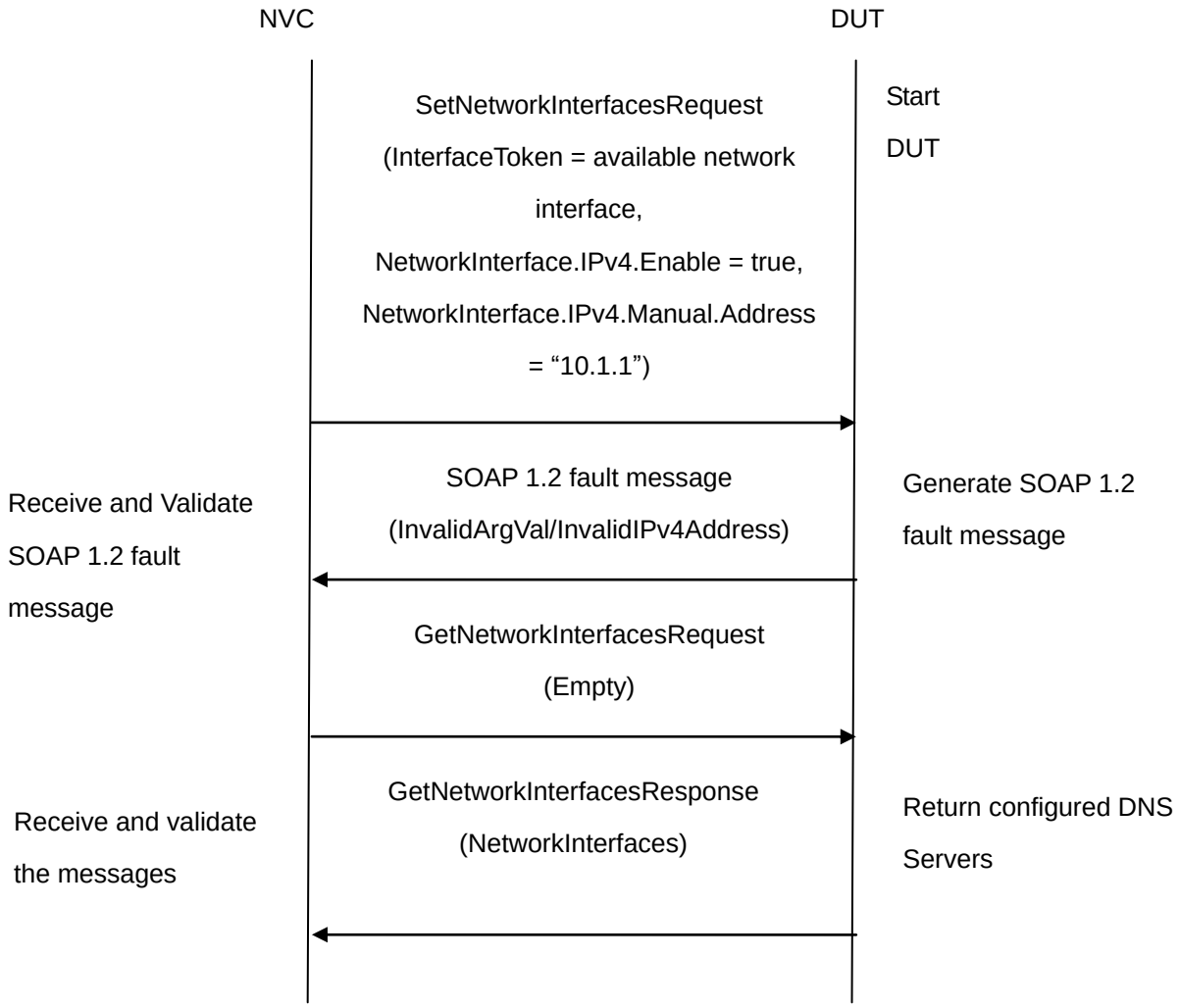
Requirement Level: SHOULD

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke SetDNSRequest message (InterfaceToken = available network interface, NetworkInterface.IPv4.Enable = true, NetworkInterface.IPv4.Manual.Address = “10.1.1”).
4. Verify that the DUT generates SOAP 1.2 fault message (InvalidArgVal/InvalidIPv4Address).
5. Retrieve network interface configurations from DUT through GetNetworkInterfacesRequest message.
6. DUT sends valid network interface configurations in the GetNetworkInterfacesResponse message.

Test Result:

PASS –

DUT passes 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.

6.2.21 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

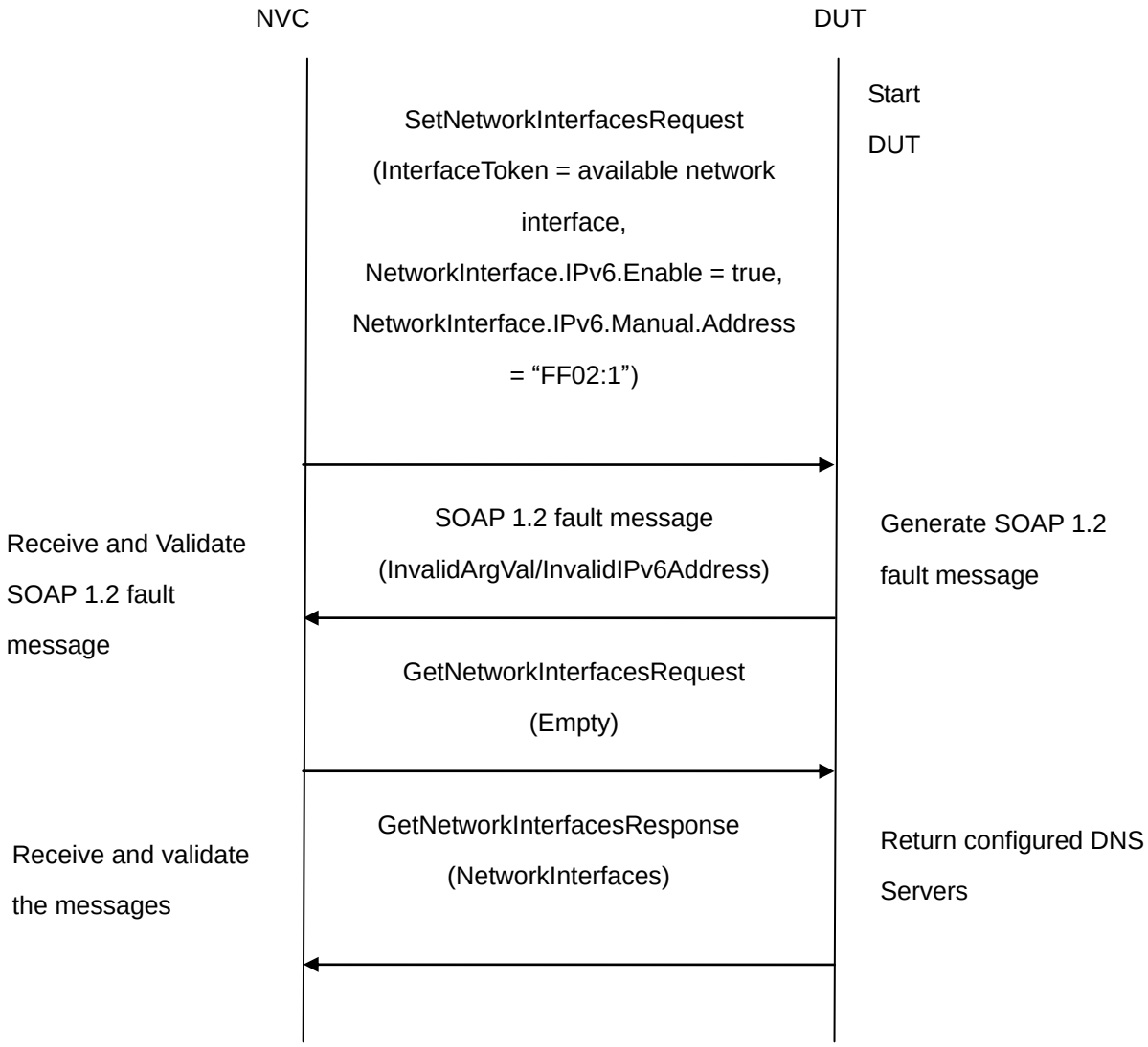
Requirement Level: SHOULD IF IMPLEMENTED (IPv6)

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke SetDNSRequest message (InterfaceToken = available network interface, NetworkInterface.IPv6.Enable = true, NetworkInterface.IPv6.Manual.Address = “FF02:1”).
4. Verify that the DUT generates SOAP 1.2 fault message (InvalidArgVal/InvalidIPv6Address).
5. Retrieve network interface configurations from DUT through GetNetworkInterfacesRequest message.
6. DUT sends valid network interface configurations in the GetNetworkInterfacesResponse message.

Test Result:

PASS –

DUT passes 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.

6.2.22 GET NETWORK PROTOCOLS CONFIGURATION

Test Label: Device Management Network Command GetNetworkProtocols Test.

Test Case ID: DEVICE-2-1-22

ONVIF Core Specification Coverage: Get network protocols

Command Under Test: GetNetworkProtocols

WSDL Reference: devicemgmt.wsdl

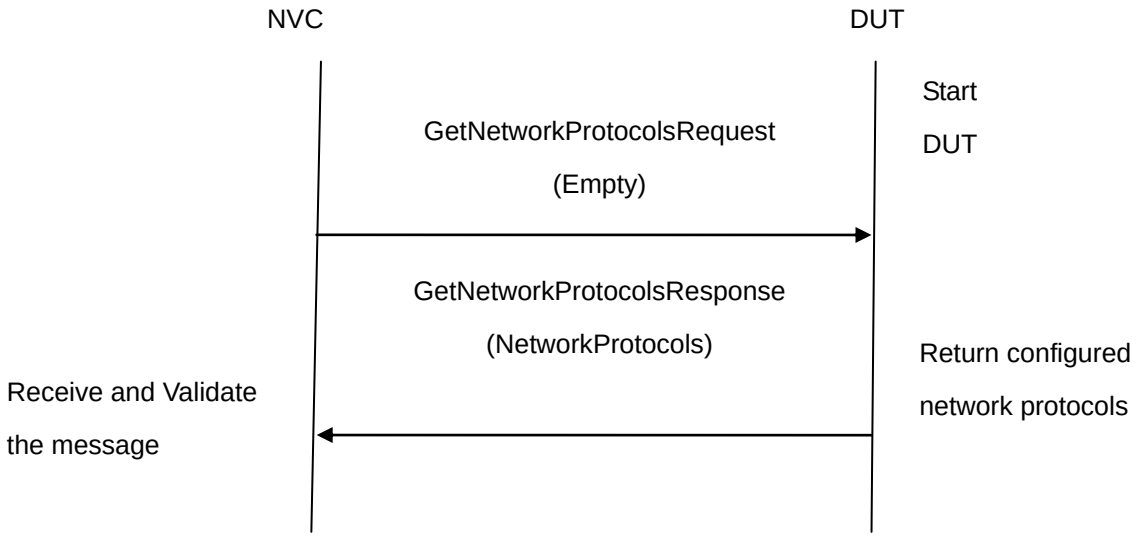
Requirement Level: MUST

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke GetNetworkProtocolsRequest message to retrieve configured network protocols of the DUT.
4. Verify the GetNetworkProtocolsResponse from DUT (NetworkProtocols = list of configured network protocols) and check that the mandatory protocols (RTSP & HTTP) are present in the list.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send GetNetworkProtocolsResponse message.

The DUT did not send correct information (i.e. NetworkProtocols = list of configured network protocols, the mandatory protocols (RTSP & HTTP) are present in the list) in the GetNetworkProtocolsResponse message.

6.2.23 SET NETWORK PROTOCOLS CONFIGURATION

Test Label: Device Management Network Command SetNetworkProtocols Test.

Test Case ID: DEVICE-2-1-23

ONVIF Core Specification Coverage: Set network protocols

Command Under Test: SetNetworkProtocols

WSDL Reference: devicemgmt.wsdl

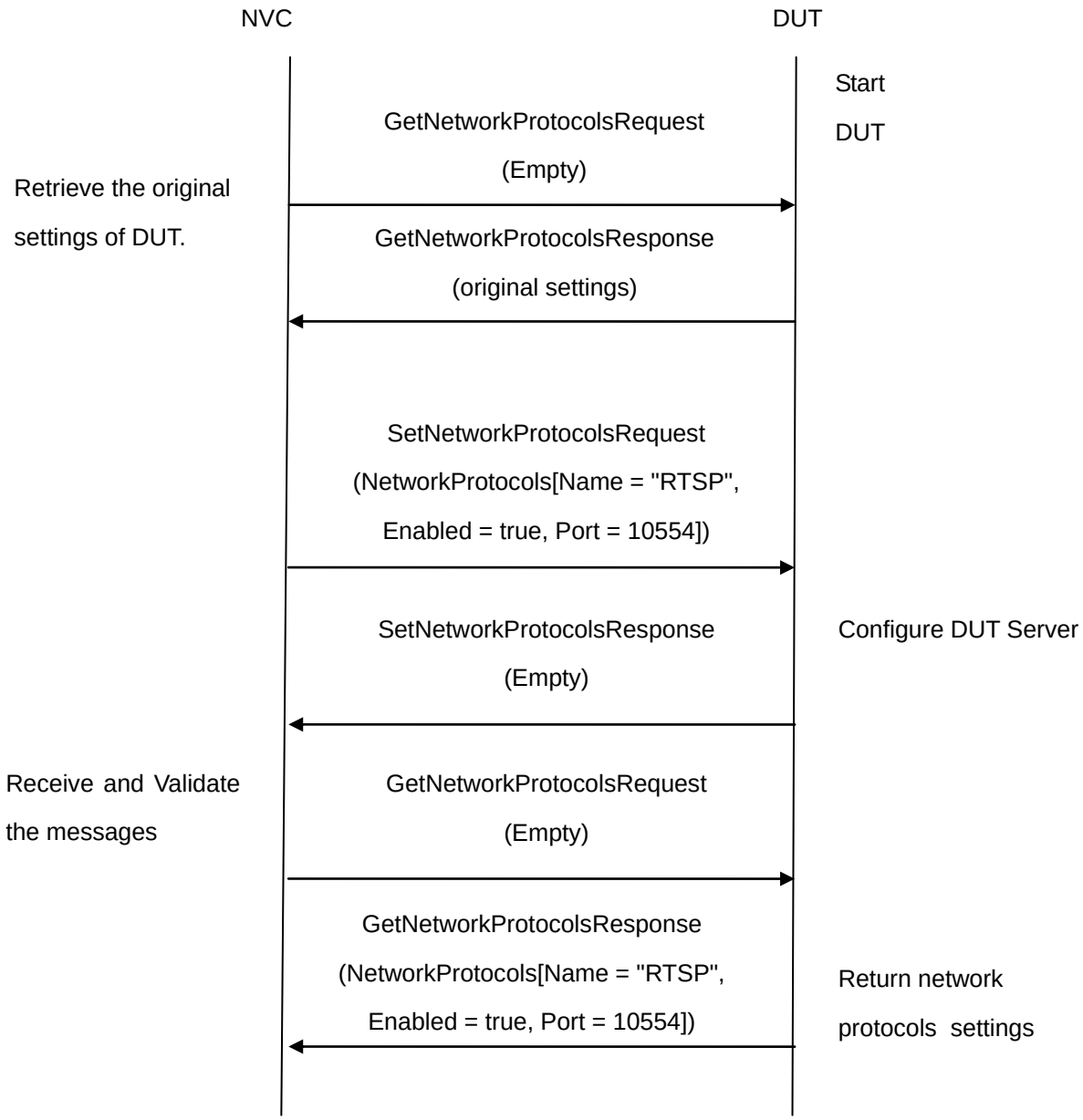
Requirement Level: MUST

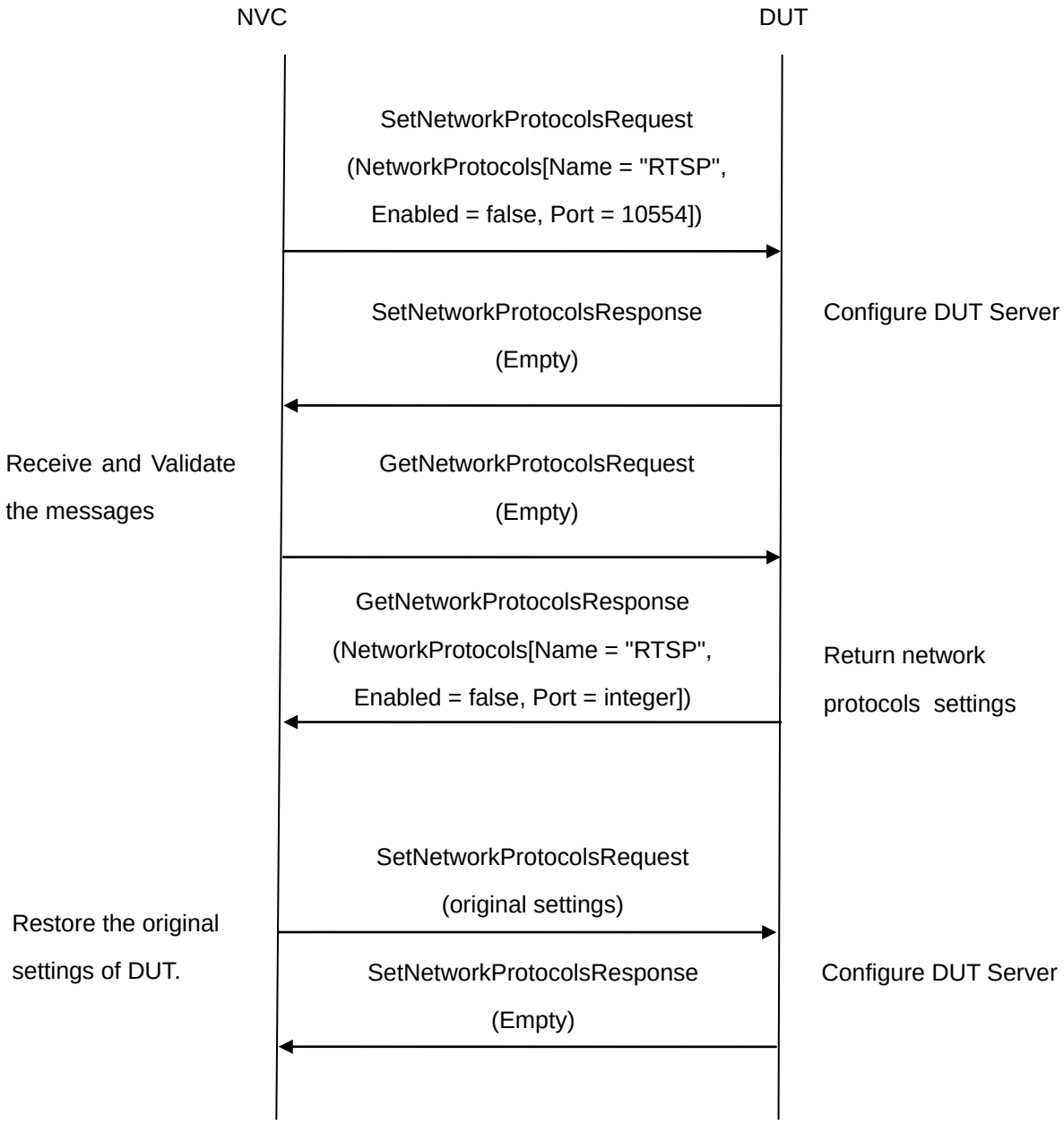
Test Purpose: To configure network protocols setting on an DUT through SetNetworkProtocols command.

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:


NVC
DUT
Start
DUT
GetNetworkProtocolsRequest
(Empty)
Retrieve the original
settings of DUT.
GetNetworkProtocolsResponse
(original settings)
SetNetworkProtocolsRequest
(NetworkProtocols[Name = "RTSP",
Enabled = true, Port = 10554])
SetNetworkProtocolsResponse
(Empty)
Configure DUT Server
Receive and Validate
the messages
GetNetworkProtocolsRequest
(Empty)
GetNetworkProtocolsResponse
(NetworkProtocols[Name = "RTSP",
Enabled = true, Port = 10554])
Return network
protocols settings



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke GetNetworkProtocolsRequest message to retrieve the original settings of DUT.
4. NVC will invoke SetNetworkProtocolsRequest message (NetworkProtocols[Name = "RTSP", Enabled = true, Port = 10554]).
5. Verify that the DUT sends SetNetworkProtocolsResponse (empty message).

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

Test Result:

PASS –

DUT passes 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 in step-7.

The DUT did not send correct network protocols information (i.e. NetworkProtocols[Name = "RTSP", Enabled = false, Port = integer]) in GetNetworkProtocolsResponse message in step-11.

6.2.24 SET NETWORK PROTOCOLS CONFIGURATION - UNSUPPORTED PROTOCOLS

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

Test Case ID: DEVICE-2-1-24

ONVIF Core Specification Coverage: Set network protocols

Command Under Test: SetNetworkProtocols

WSDL Reference: devicemgmt.wsdl

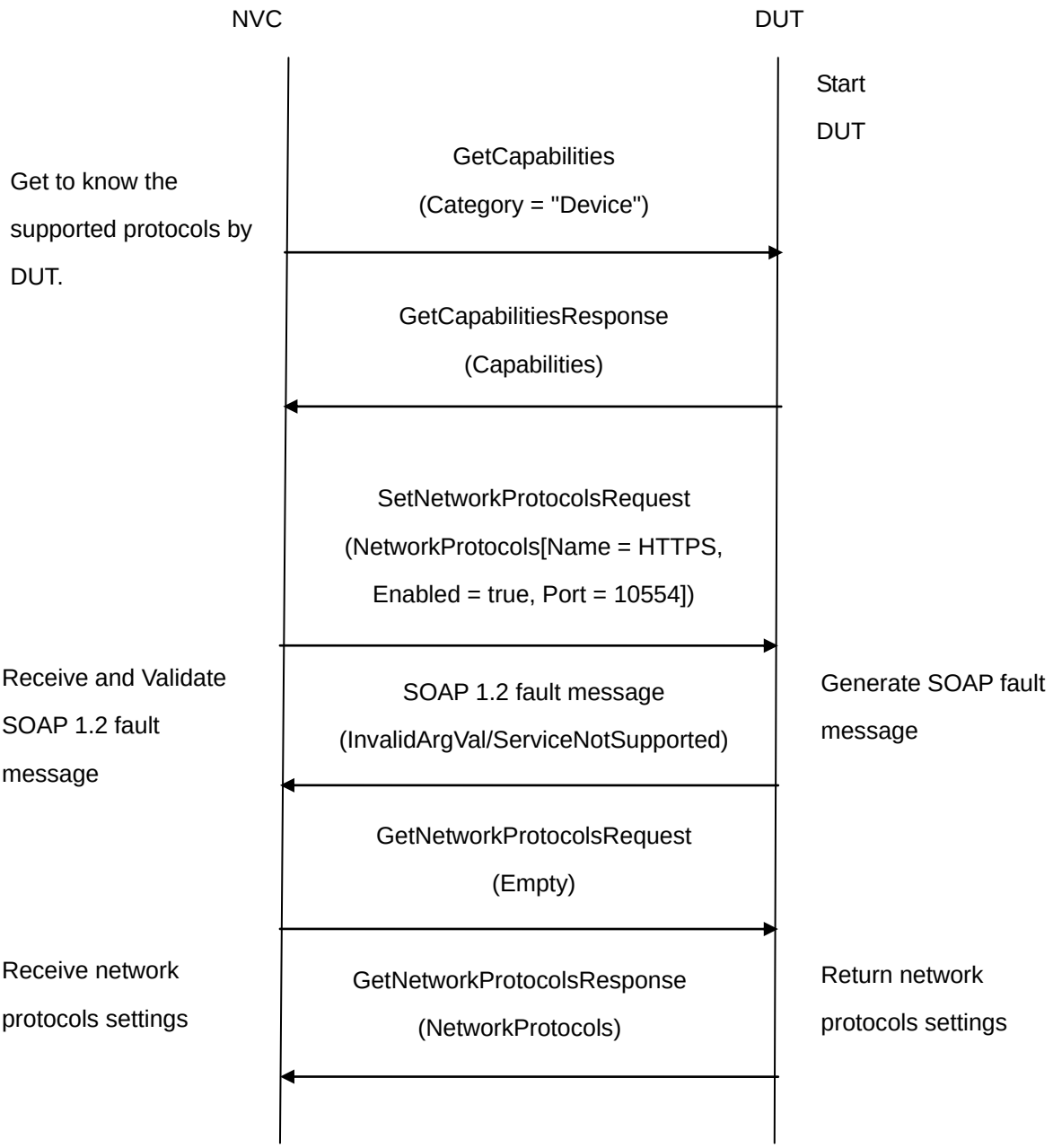
Requirement Level: SHOULD

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

Pre-Requisite: DUT does not support all protocols.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke GetCapabilities message (Category = "Device") to get to know the supported protocols by DUT.

4. DUT will return GetCapabilitiesResponse message.
5. If DUT supports HTTPS (i.e. GetCapabilitiesResponse message includes Capabilities.Device.Security.TLS1.1 = true, Capabilities.Device.Security.TLS1.2 = true or Capabilities.Device.Security.Extension.TLS1.0 = true), then continue to next test case.
6. NVC will invoke SetNetworkProtocolsRequest message (NetworkProtocols[Name = HTTPS, 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. DUT sends valid network protocol configurations in the GetNetworkProtocolsResponse message (NetworkProtocols = supported protocols).

Test Result:

PASS –

DUT passes 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.

The DUT did not send GetCapabilitiesResponse message.

6.2.25 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

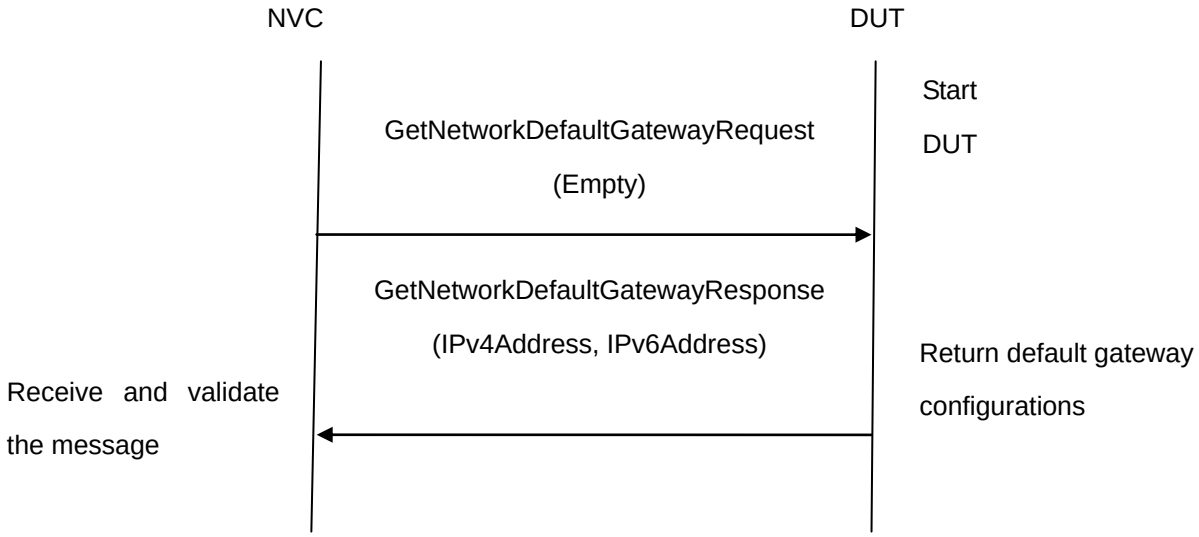
Requirement Level: MUST

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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 –

DUT passes 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.

6.2.26 SET NETWORK DEFAULT GATEWAY CONFIGURATION - IPV4

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

Test Case ID: DEVICE-2-1-26

ONVIF Core Specification Coverage: Set default gateway

Command Under Test: SetNetworkDefaultGateway

WSDL Reference: devicemgmt.wsdl

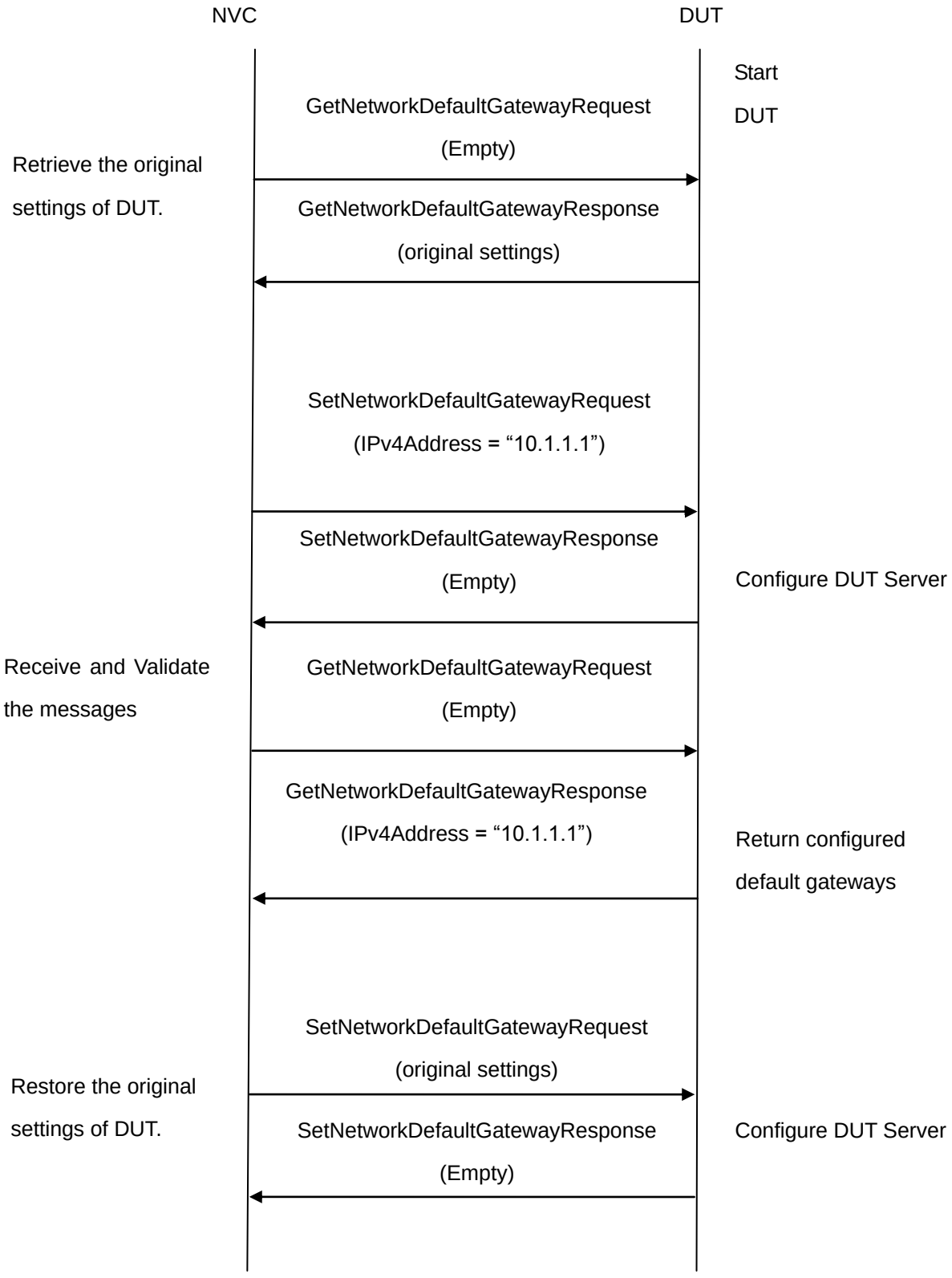
Requirement Level: MUST

Test Purpose: To configure default gateway IPv4 address setting on an DUT through SetNetworkDefaultGateway command.

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.

2. Start the DUT.
3. NVC will invoke GetNetworkDefaultGatewayRequest message to retrieve the original settings of DUT.
4. NVC will invoke SetNetworkDefaultGatewayRequest message (IPv4Address = “10.1.1.1”).
5. Verify that the DUT sends SetNetworkDefaultGatewayResponse (empty message).
6. Verify the configured default gateways settings in DUT through GetNetworkDefaultGatewayRequest message.
7. DUT sends its configured default gateways settings in the GetNetworkDefaultGatewayResponse message (IPv4Address = “10.1.1.1”).
8. NVC will invoke SetNetworkDefaultGatewayRequest message to restore the original settings of DUT.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send SetNetworkDefaultGatewayResponse message in step-5.

The DUT did not send GetNetworkDefaultGatewayResponse message in step-7.

The DUT did not send correct default gateway information (i.e. IPv4Address = “10.1.1.1”) in GetNetworkDefaultGatewayResponse message in step-7.

Note: See Annex A.6 for Valid IPv4 Address definition.

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

6.2.27 SET NETWORK DEFAULT GATEWAY CONFIGURATION - IPV6

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

Test Case ID: DEVICE-2-1-27

ONVIF Core Specification Coverage: Set default gateway

Command Under Test: SetNetworkDefaultGateway

WSDL Reference: devicemgmt.wsdl

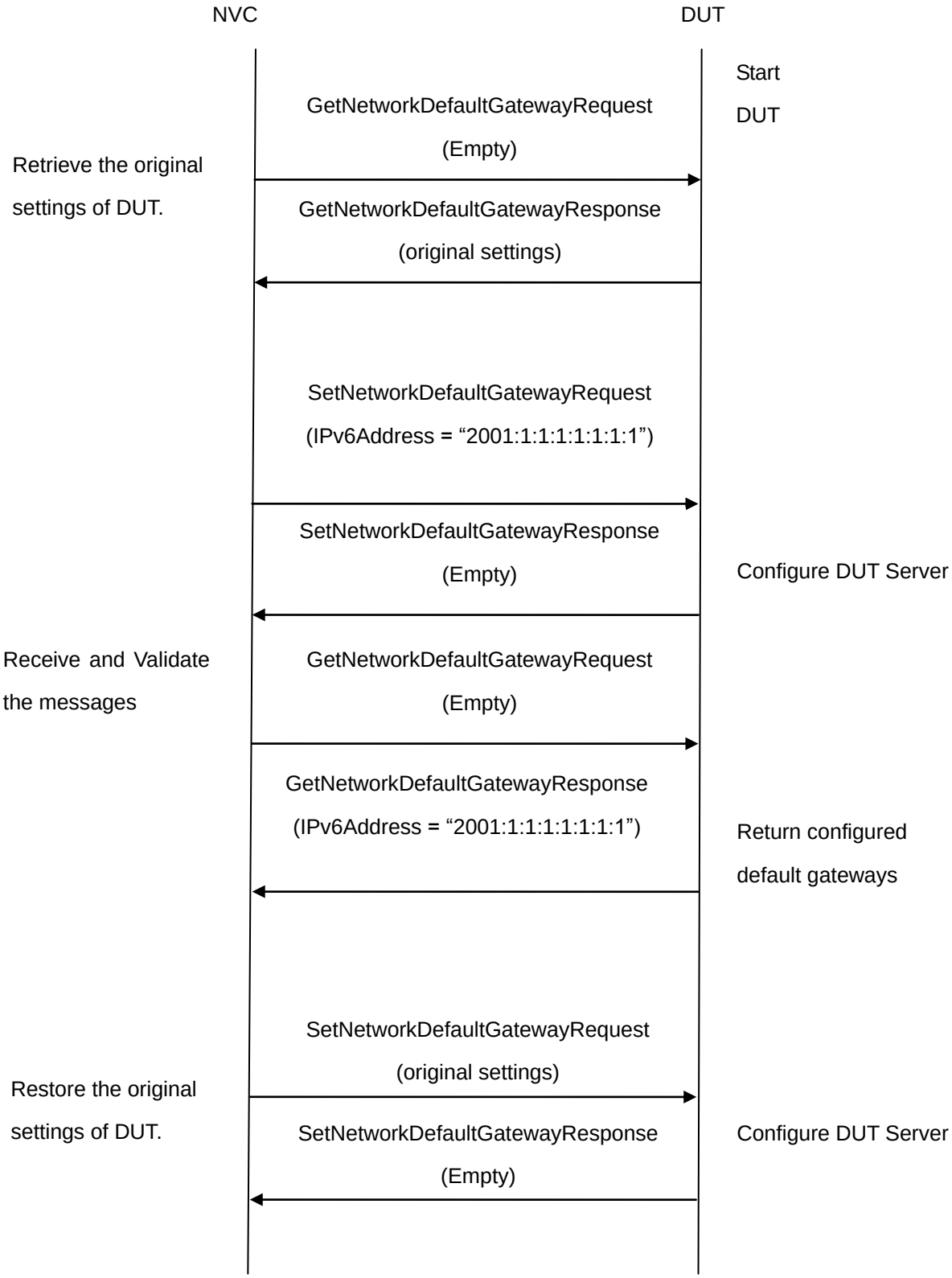
Requirement Level: SHOULD IF IMPLEMENTED (IPv6)

Test Purpose: To configure default gateway IPv6 address setting on an DUT through SetNetworkDefaultGateway command.

Pre-Requisite: IPv6 is implemented by DUT.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke GetNetworkDefaultGatewayRequest message to retrieve the original settings of DUT.
4. NVC will invoke SetNetworkDefaultGatewayRequest message (IPv6Address = “2001:1:1:1:1:1:1:1”).
5. Verify that the DUT sends SetNetworkDefaultGatewayResponse (empty message).
6. Verify the configured default gateways settings in DUT through GetNetworkDefaultGagewayRequest message.
7. DUT sends its configured default gateways settings in the GetNetworkDefaultGatewayResponse message (IPv6Address = “2001:1:1:1:1:1:1:1”).
8. NVC will invoke SetNetworkDefaultGatewayRequest message to restore the original settings of DUT.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send SetNetworkDefaultGatewayResponse message in step-5.

The DUT did not send GetNetworkDefaultGatewayResponse message in step-7.

The DUT did not send correct default gateway information (i.e. IPv6Address = “2001:1:1:1:1:1:1:1”) in GetNetworkDefaultGatewayResponse message in step-7.

6.2.28 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

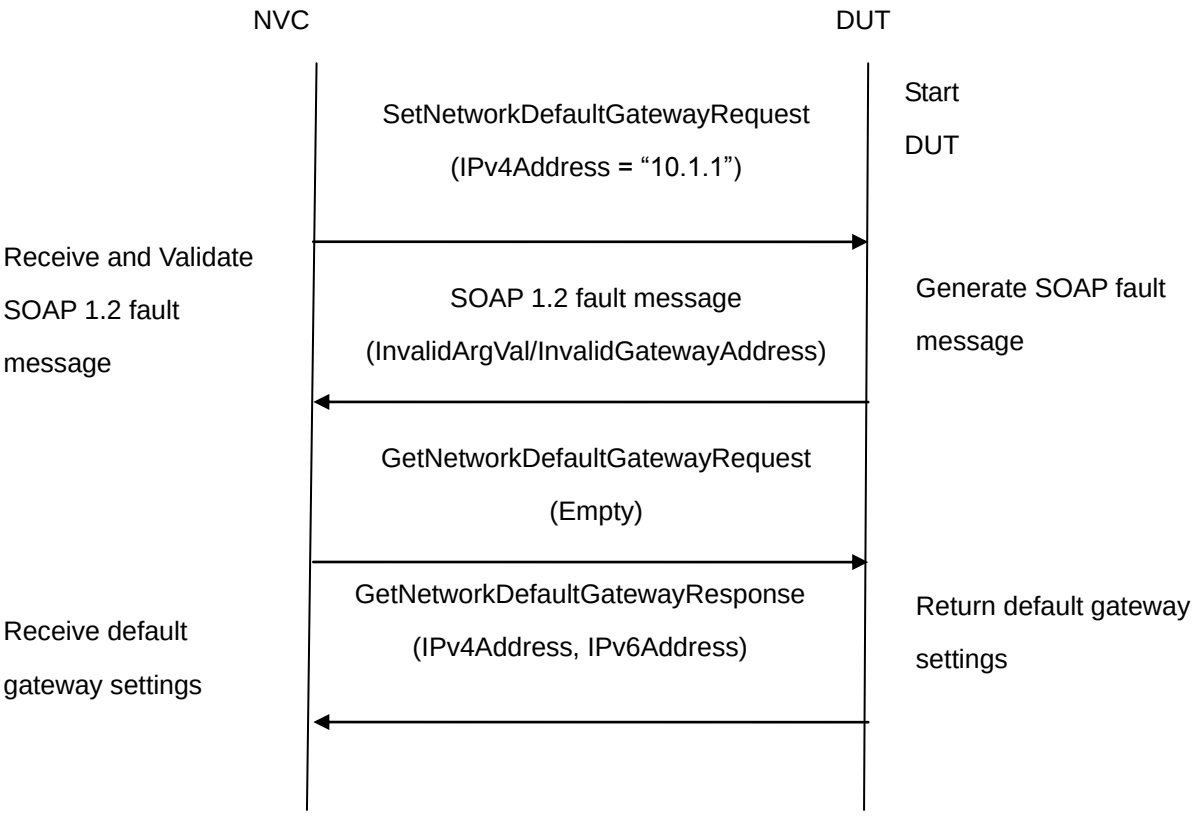
Requirement Level: SHOULD

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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. 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 –

DUT passes 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 “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.

6.2.29 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

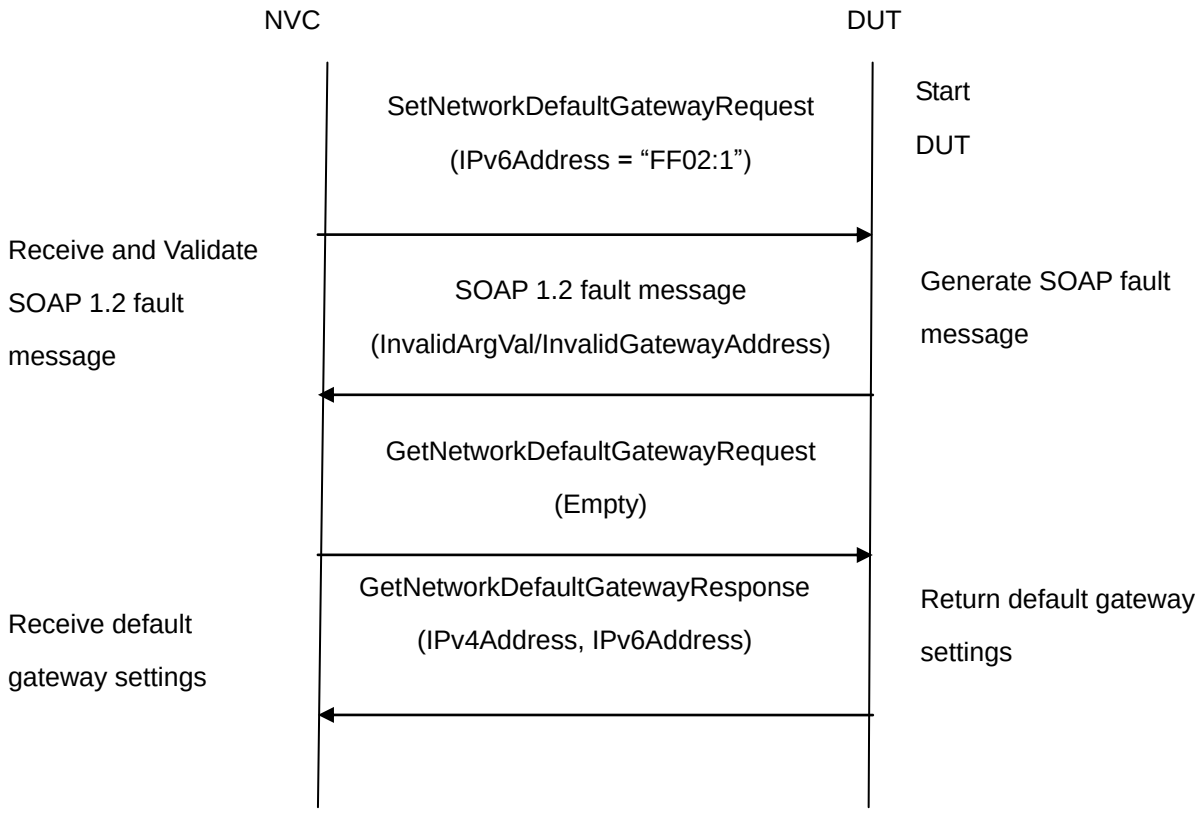
Requirement Level: SHOULD IMPLEMENTED (IPv6)

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

Pre-Requisite: IPv6 is implemented by DUT.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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. 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 –

DUT passes 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.

6.3 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

Requirement Level: MUST

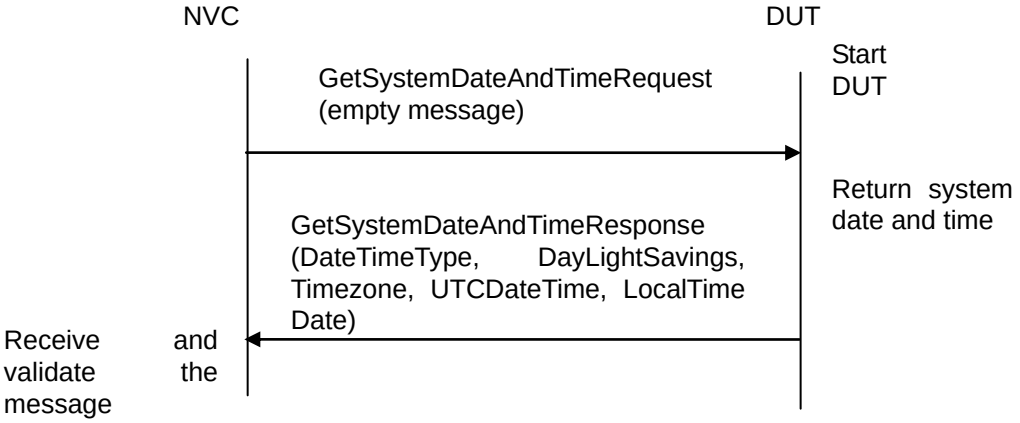
WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: None.

Test Configuration: NVC and DUT.

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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 LocalTimeDate = Hour:Min:Sec, Year:Month:Day).

Test Result:

PASS –

DUT passes 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 MUST 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

Requirement Level: MUST

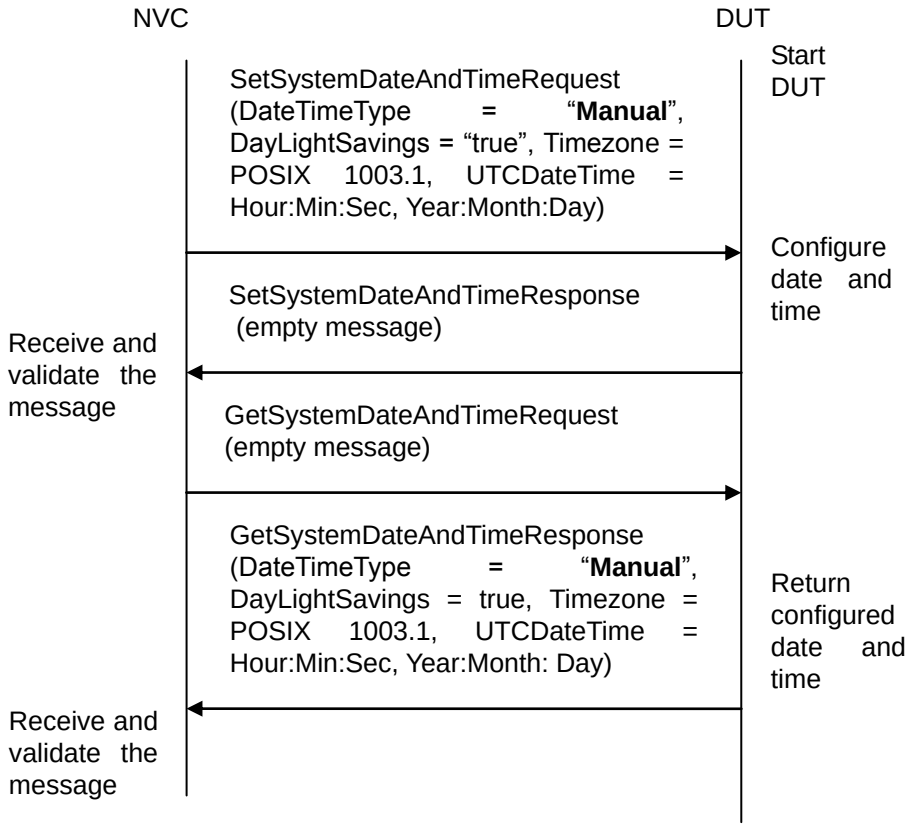
WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: None.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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 –

DUT passes 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 USING NTP

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

Test Case ID: DEVICE-3-1-3

ONVIF Core Specification Coverage: Set system date and time

Command Under Test: SetSystemDateAndTime

Requirement Level: MUST IF SUPPORTED(NTP).

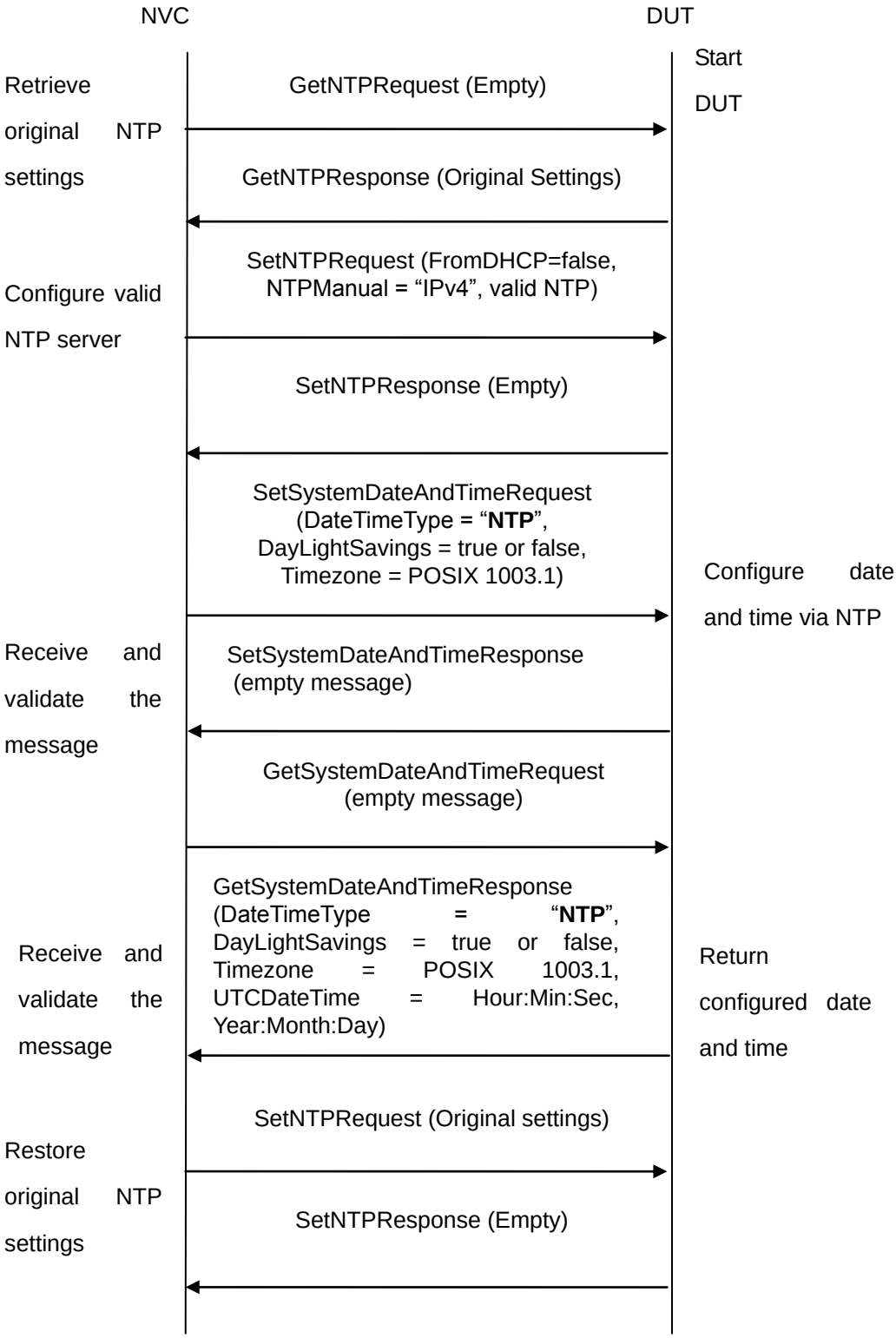
WSDL Reference: devicemgmt.wsdl

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

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

Test Configuration: NVC, DUT and NTP.

Test Sequence:



Test Procedure:

1. Start an NVC.

2. Start the DUT.
3. NVC invokes GetNTPRequest to retrieve original NTP settings of DUT.
4. NVC invokes SetNTPRequest (FromDHCP=false, NTPManual = “IPv4”, valid NTP server address) to configure DUT with proper NTP server.
5. NVC will invoke SetSystemDateAndTimeRequest message (DateTimeType = “NTP”, DayLightSavings = true or false, Timezone = POSIX 1003.1) to configure the time in the DUT.
6. DUT shall obtain and configure time via NTP.
7. Verify that the DUT sends SetSystemDateAndTimeResponse (empty message).
8. Verify the DUT date and time configurations through GetSystemDateAndTimeRequest message.
9. 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).
10. NVC invokes SetNTPRequest (original NTP Settings) to restore NTP settings of DUT.

Test Result:

PASS –

DUT passes 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.

6.3.4 SYSTEM COMMAND SETSYSTEMDATEANDTIME TEST FOR INVALID TIMEZONE

Test Label: Device Management Network Command **SetSystemDateAndTime** Test, for invalid timezone.

Test Case ID: DEVICE-3-1-4

ONVIF Core Specification Coverage: Set system date and time.

Command Under Test: SetSystemDateAndTime

Requirement Level: SHOULD

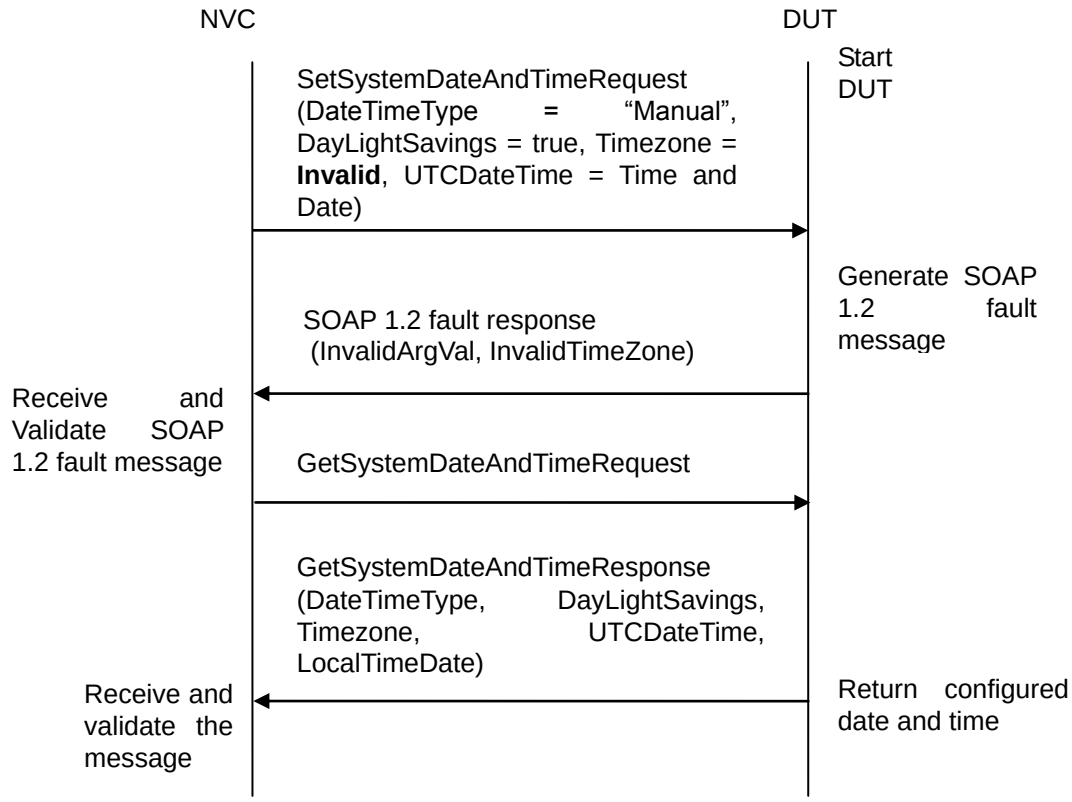
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify the behaviour of the DUT for invalid Timezone configuration.

Pre-Requisite: None.

Test Configuration: NVC and DUT.

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke SetSystemDateAndTimeRequest message with invalid Timezone (DateTimeType "Manual", DayLightSavings = "True", Timezone = Invalid, UTCDateTime = Hour:Min:Sec, Year:Month:Day).
4. Verify that 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 LocalTimeDate = Hour:Min:Sec, Year:Month:Day).

Test Result:

PASS –

DUT passes 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 MUST return UTCDateTime or LocalDateTime in the GetSystemDateAndTimeResponse message.

See Annex A.3 for Invalid TimeZone and SOAP 1.2 fault message definitions.

6.3.5 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

Requirement Level: SHOULD

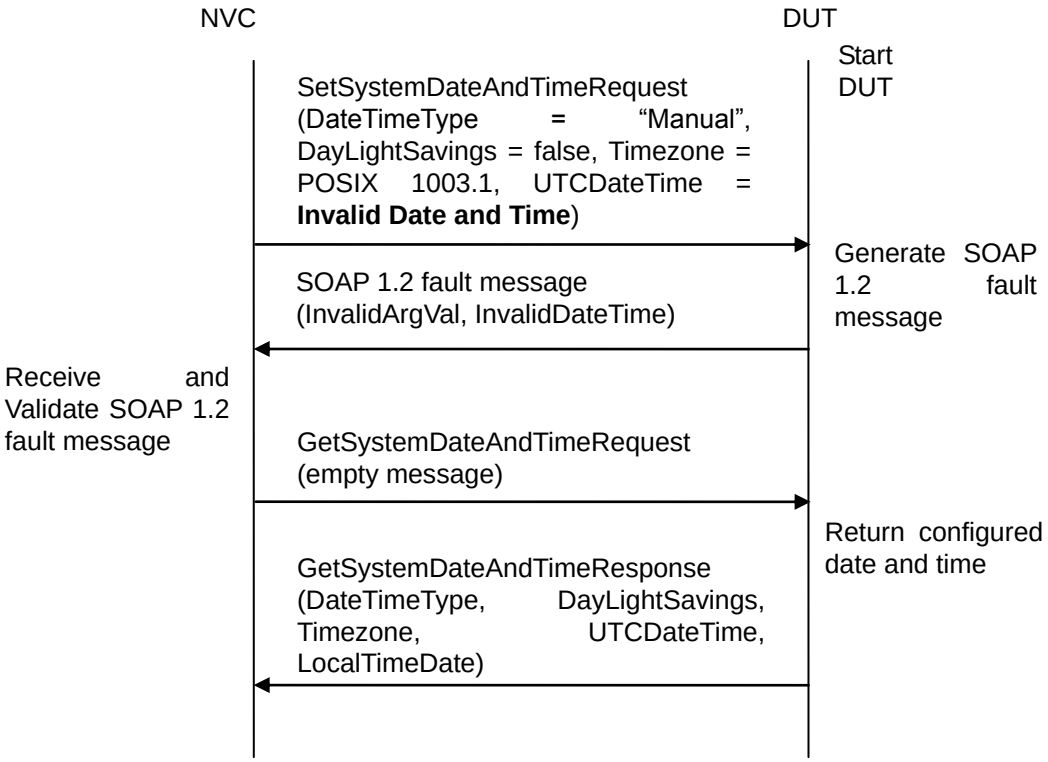
WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: None.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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 DUT generates SOAP 1.2 fault message (InvalidArgVal, InvalidDateTime).
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 LocalTimeDate = Hour:Min:Sec, Year:Month:Day).

Test Result:

PASS –

DUT passes 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 DUT MUST return UTCDateTime or LocalDateTime in the GetSystemDateAndTimeResponse message.

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

6.3.6 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

Requirement Level: MUST

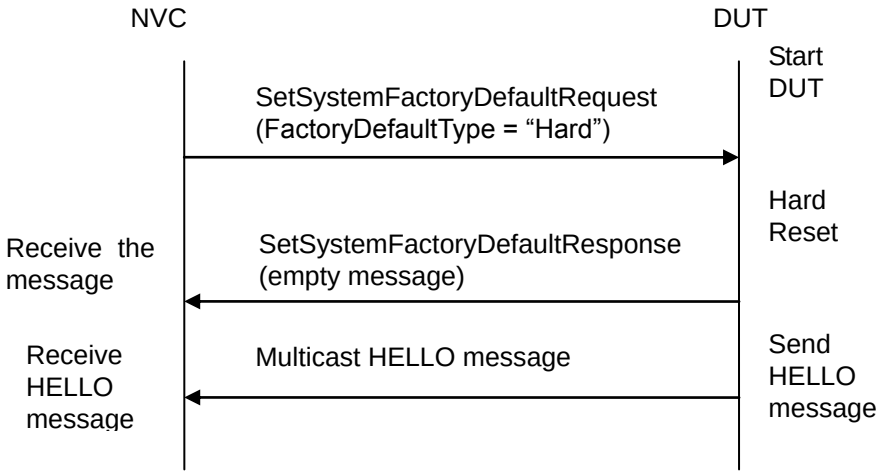
WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: None.

Test Configuration: NVC and DUT.

Test Sequence:



Test Procedure:

1. Start an NVC.

2. Start the DUT.
3. NVC 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 –

DUT passes 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 must be configured with an IPv4 address, must be IP reachable in the test network and other relevant configurations to be done for further tests.

6.3.7 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

Requirement Level: MUST

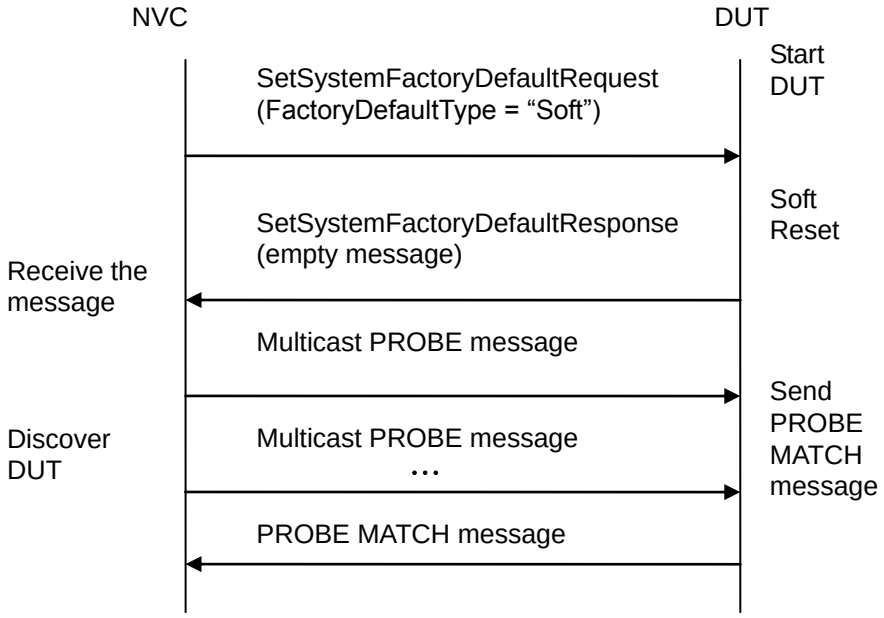
WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: None.

Test Configuration: NVC and DUT.

Test Sequence:



Test Procedure:

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

Test Result:

PASS –

DUT passes 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.

6.3.8 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

Requirement Level: MUST

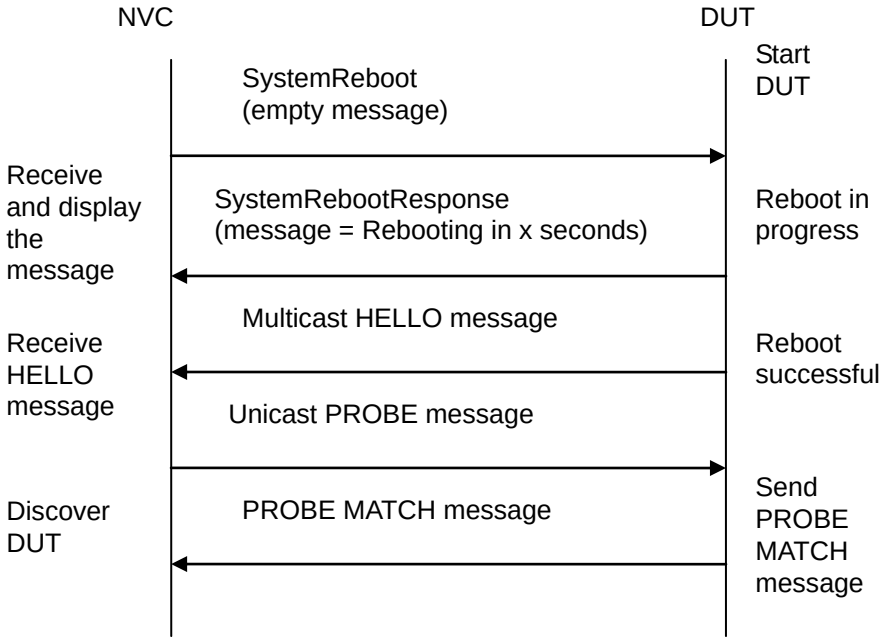
WSDL Reference: devicemgmt.wsdl

Test Purpose: To reboot the DUT through SystemReboot command.

Pre-Requisite: None.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke SystemReboot message to reset the DUT.
4. Verify that DUT sends SystemRebootResponse message (example message string = “Rebooting in x seconds”).
5. DUT will send Multicast HELLO message after it is successfully rebooted.
6. NVC will verify the HELLO message sent by DUT.

7. NVC will send Unicast PROBE message to discover the DUT.
8. DUT will send a PROBE MATCH message.
9. NVC will verify the PROBE MATCH message sent by DUT.

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

Test Result:

PASS –

DUT passes 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.

6.3.9 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

Requirement Level: MUST

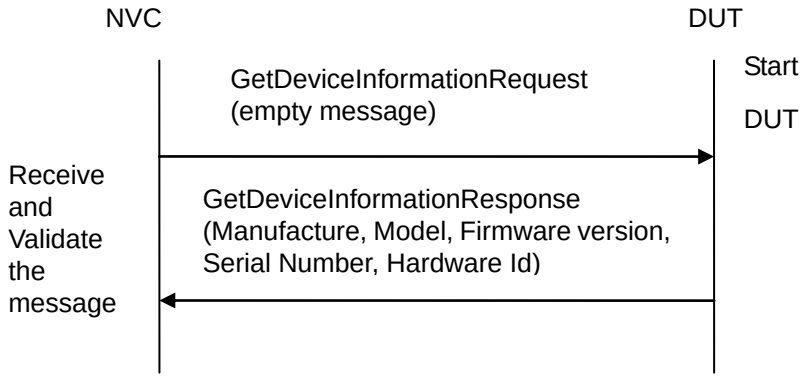
WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: None.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

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

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send GetDeviceInformationResponse message.

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

6.3.10 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

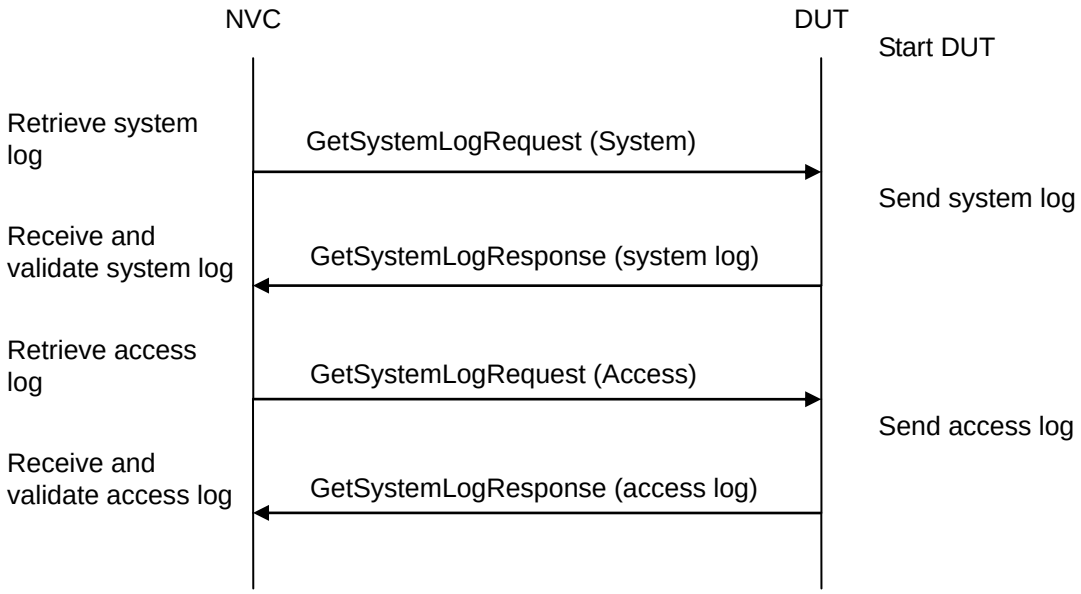
Requirement Level: SHOULD

Test Porpose: To retrieve DUT system and access logs through GetSystemLog command.

Pre-Requisite: None.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start a DUT.
3. NVC 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. NVC 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 –

DUT passes 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.).

6.4 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.

Requirement Level: MUST

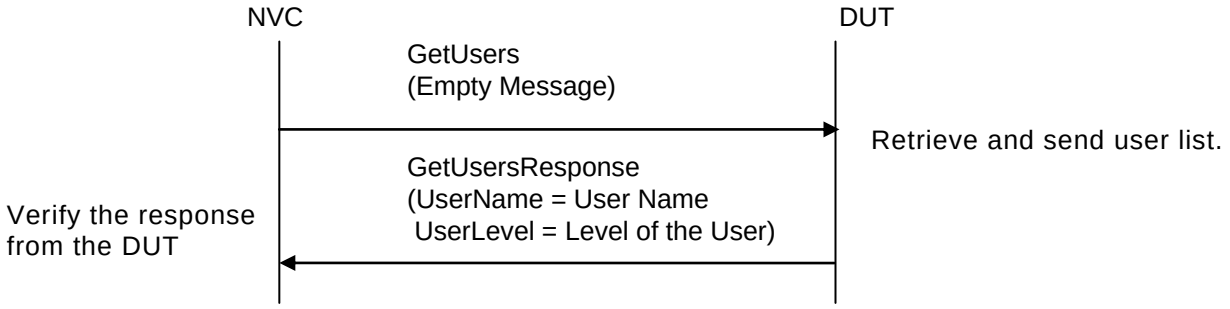
WSDL Reference: devicemgmt.wsdl

Test Purpose: To verify the behaviour of GetUsers command.

Pre-Requisite: It might be necessary that, NVC may need to operate in administrator mode to execute this test case.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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 –

DUT passes 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

Test Label: Device Management Security Command **CreateUsers** Verification.

Test Case ID: DEVICE-4-1-2

ONVIF Core Specification Coverage: Create users.

Command Under Test: CreateUsers

Requirement Level: MUST

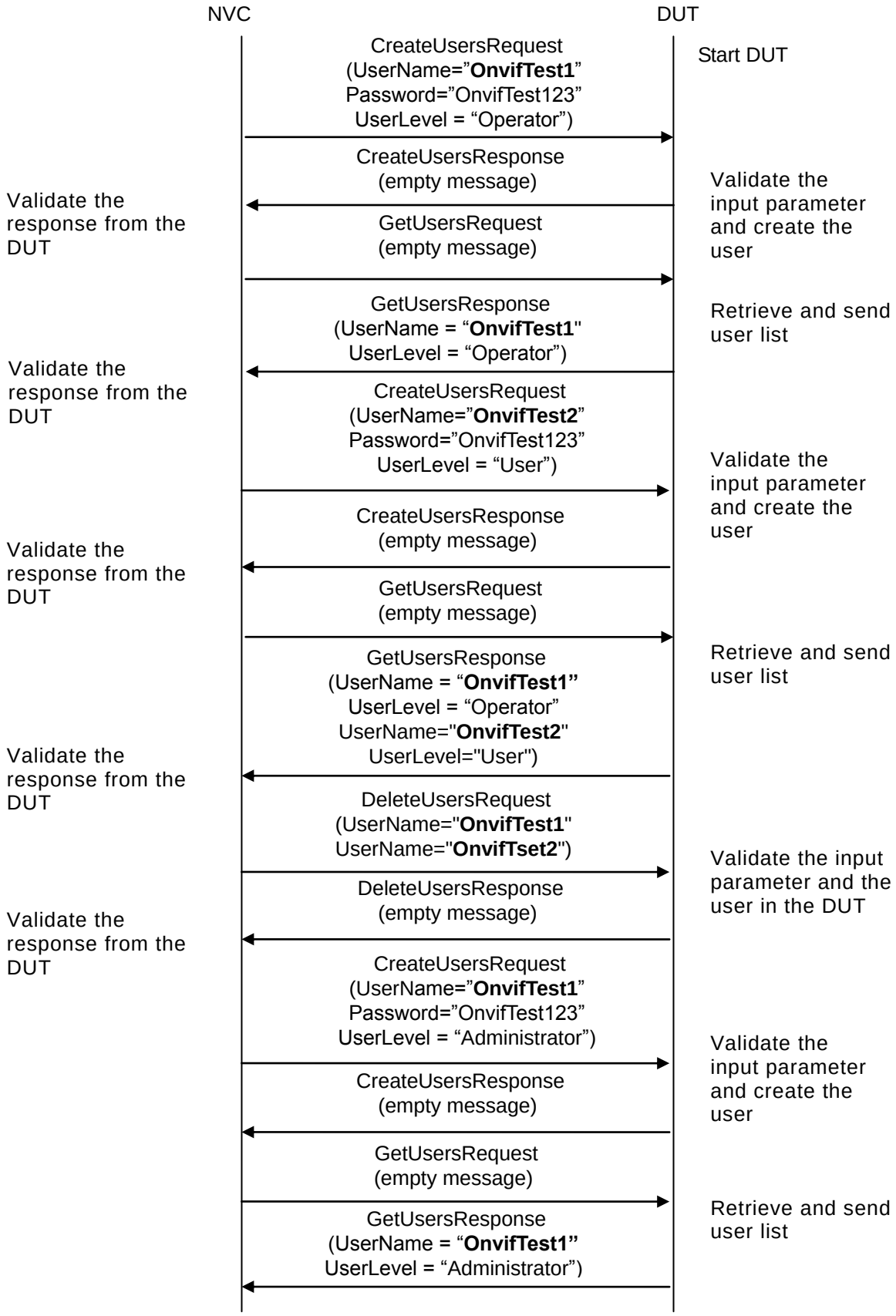
WSDL Reference: devicemgmt.wsdl

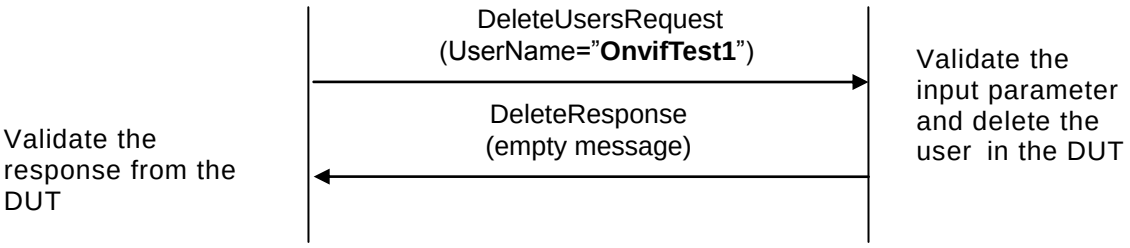
Test Purpose: To verify the behaviour of CreateUsers command.

Pre-Requisite: It might be necessary that, NVC may need to operate in administrator mode to execute this test case.

Test Configuration: NVC and DUT.

Test Sequence:


NVC
DUT
CreateUsersRequest (UserName="OnvifTest1" Password="OnvifTest123" UserLevel = "Operator")
Start DUT
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
CreateUsersRequest (UserName="OnvifTest1" Password="OnvifTest123" UserLevel = "Administrator")
Validate the input parameter and create the user
CreateUsersResponse (empty message)
GetUsersRequest (empty message)
GetUsersResponse (UserName = "OnvifTest1" UserLevel = "Administrator")
Retrieve and send user list



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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. NVC 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").
7. NVC will invoke CreateUsersRequest message (UserName = "OnvifTest2" Password = "OnvifTest123" UserLevel = "User") to create the user in the DUT.
8. Verify that DUT sends CreateUsersResponse message (empty message).
9. NVC will invoke GetUsersRequest message (empty message), to retrieve the user list from the DUT.
10. Verify that DUT sends the GetUsersResponse message (UserName = "OnvifTest1" UserLevel = "Operator", UserName ="OnvifTest2" UserLevel = "User").
11. NVC will invoke DeleteUsersRequest message (UserName = "OnvifTest1", UserName = "OnvifTest2") to delete the users in the DUT.
12. Verify that DUT sends DeleteUsersResponse message (empty Message).
13. NVC will invoke CreateUsersRequest message (UserName="OnvifTest1" Password="OnvifTest123" UserLevel="Administrator"), to create the user in the DUT.
14. Verify that DUT sends CreateUsersResponse message (empty message).
15. NVC will invoke GetUsersRequest message (empty message) to retrieve the user list from the DUT.
16. Verify that DUT sends GetUsersResponse message (UserName="OnvifTest1", UserLevel="Administrator").
17. NVC will invoke DeleteUsersRequest message (UserName="OnvifTest1") to delete the users in the DUT.
18. Verify that DUT sends DeleteUsersResponse message (empty message).

Test Result:

PASS –

DUT passes all assertions

DUT creates user either at step 3 or step 10 or step 16 successfully or DUT creates users at step 3, 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:

1 It may be possible that 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.

2 It may be possible that DUT may return more number of users than actually created in this test case i.e. default users if any might be returned.

3 In some DUT's it might not be possible to create user with all levels. So if the DUT successfully creates the user either at step 3 or step 10 or step 16 successfully then this test case shall be treated as PASS case.

6.4.3 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.

Requirement Level: SHOULD

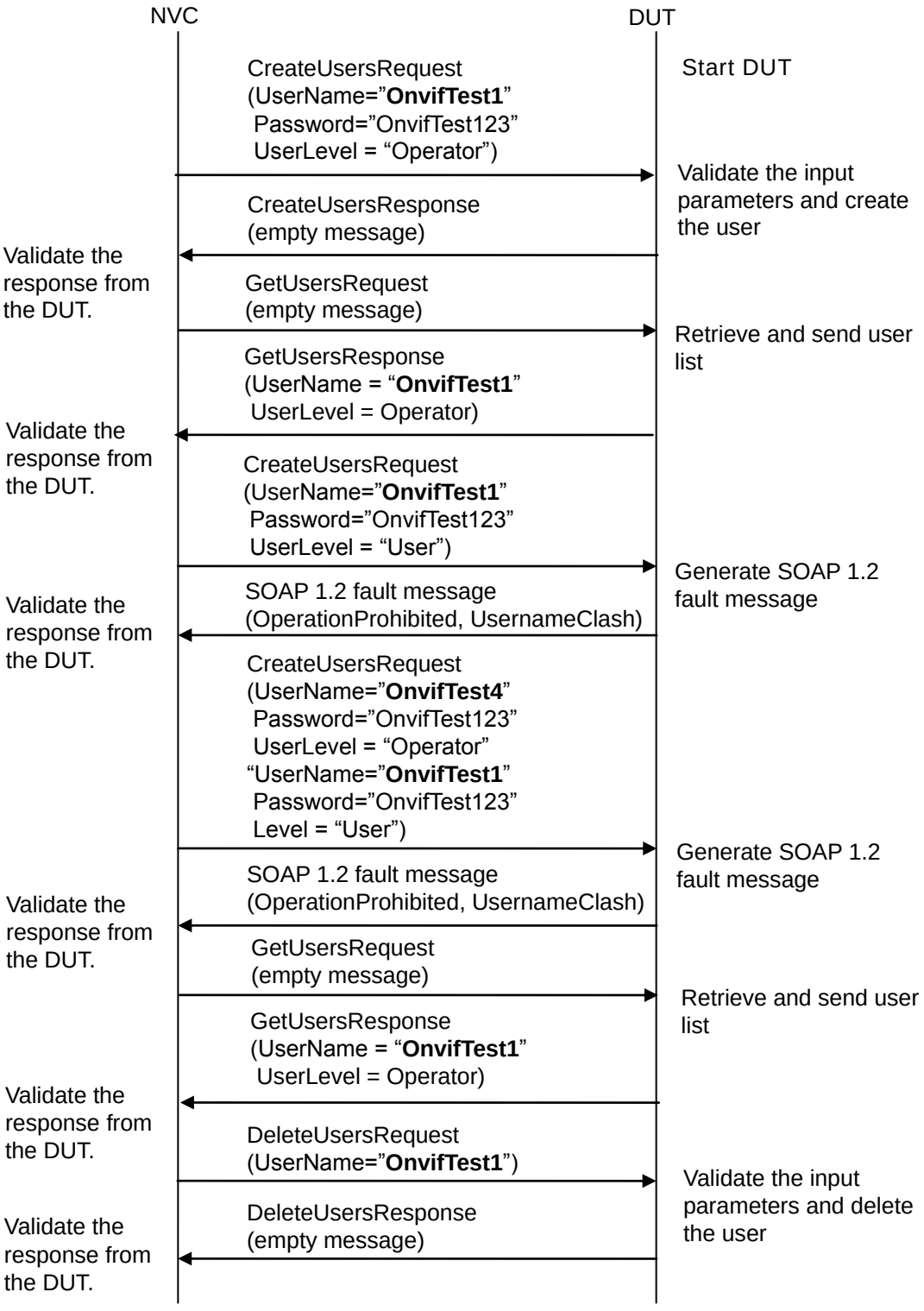
WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: It might be necessary that, NVC may need to operate in administrator mode to execute this test case.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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. NVC 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)
7. NVC will invoke CreateUsersRequest message (UserName = ”OnvifTest1” Password = OnvifTest123 UserLevel = User) to create the user in the DUT.
8. Verify that NVC generates SOAP 1.2 fault message (OperationProhibited, UsernameClash).
9. NVC 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 NVC generates SOAP 1.2 fault message (OperationProhibited, UsernameClash).
11. NVC 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. NVC will invoke DeleteUsersRequest message (UserName = ”OnvifTest1”) to delete the user in the DUT.
14. Verify that DUT sends DeleteUsersResponse message (empty message).

Test Result:

PASS –

DUT passes 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:

1 It may be possible that 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.

2 It may be possible that DUT may return more number of users than actually created in this test case i.e. default users if any might be returned.

6.4.4 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

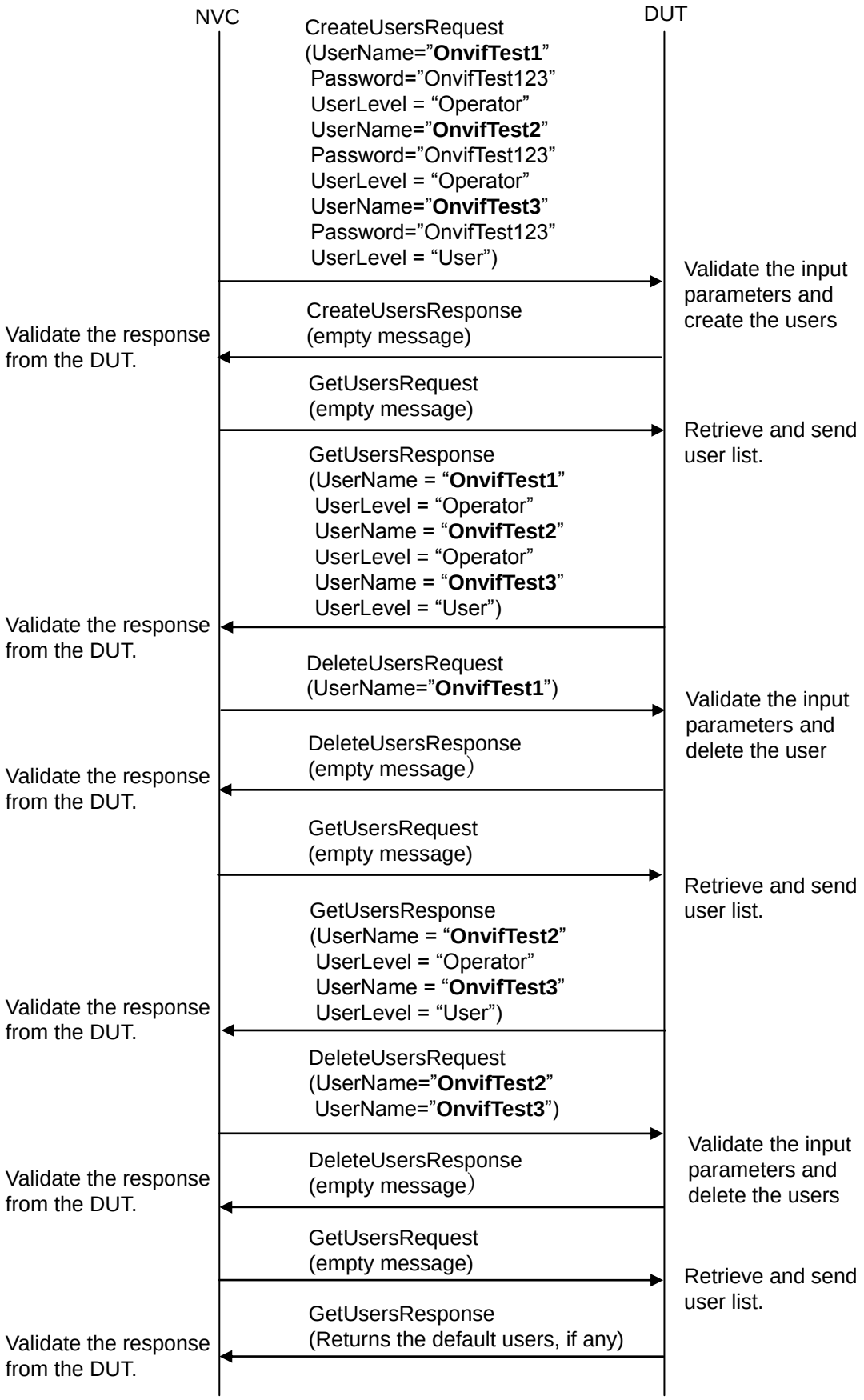
Requirement Level: MUST

WSDL Reference: devicemgmt.wsdl

Pre-Requisite: It might be necessary that, NVC may need to operate in administrator mode to execute this test case.

Test Configuration: NVC and DUT.

Test Sequence:


NVC
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 NVC
2. Start the DUT.
3. NVC 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. NVC 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. NVC will invoke DeleteUsersRequest message (UserName=”OnvifTest1”), to delete the user.
8. Verify that DUT sends DeleteUsersResponse message (empty message).
9. NVC 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. NVC will invoke DeleteUsersRequest message (UserName = “OnvifTest2”, UserName = “OnvifTest3”) to delete the users.
12. Verify that DUT sends the DeleteUserResponse message (empty message).
13. NVC 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 –

DUT passes 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:

1 It may be possible that 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.

2 It may be possible that DUT may return more number of users than actually created in this test case i.e. default users if any might be returned.

6.4.5 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.

Requirement Level: SHOULD.

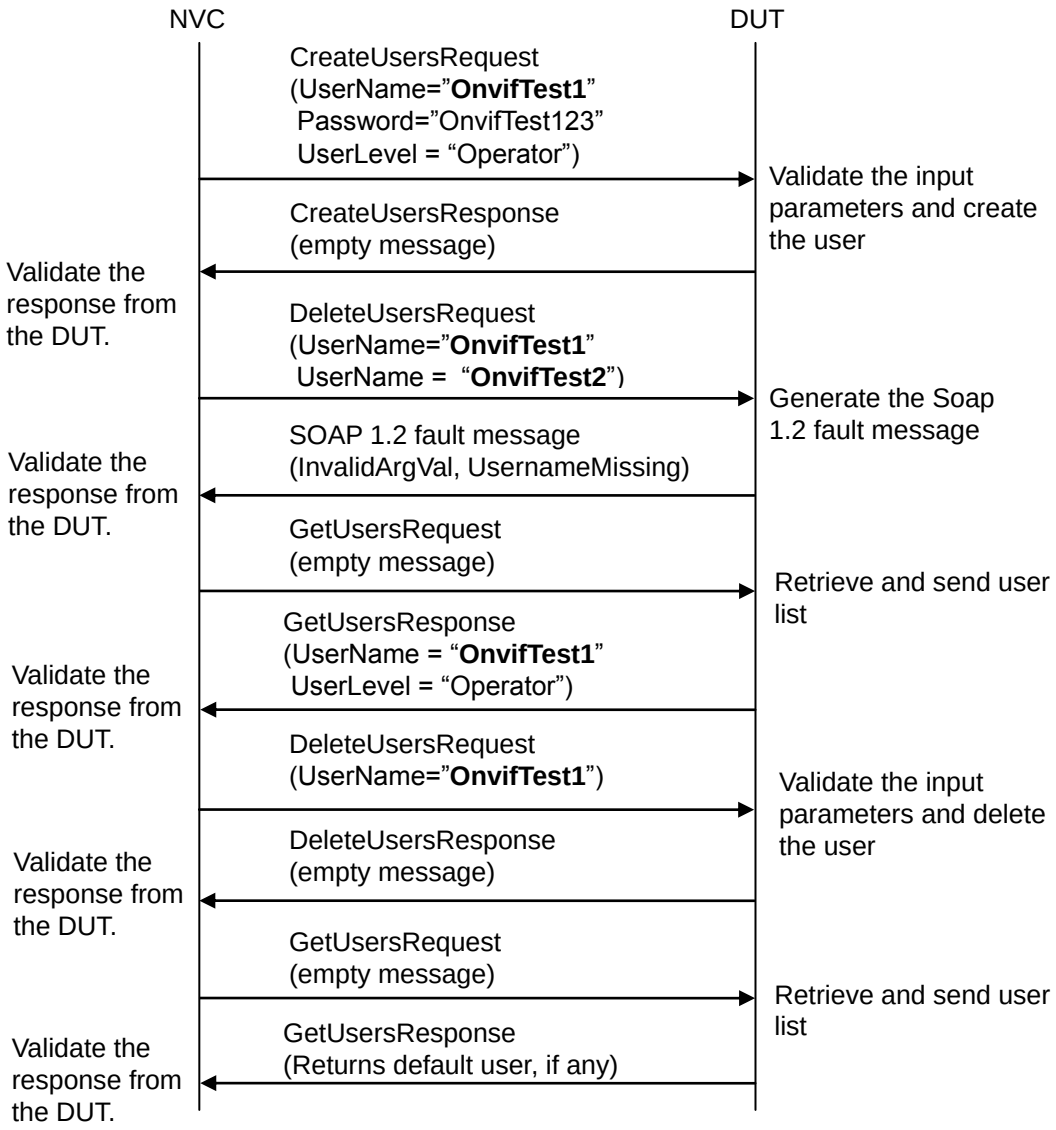
WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: It might be necessary that, NVC may need to operate in administrator mode to execute this test case.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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. NVC 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. NVC 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. NVC will invoke DeleteUsersRequest message (UserName = “OnvifTest1”), to delete the user.
10. Verify that DUT sends DeleteUsersResponse message (empty message).
11. NVC 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 –

DUT passes 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:

1 It may be possible that 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.

2 It may be possible that DUT may return more number of users than actually created in this test case i.e. default users if any might be returned.

6.4.6 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.

Requirement Level: MUST

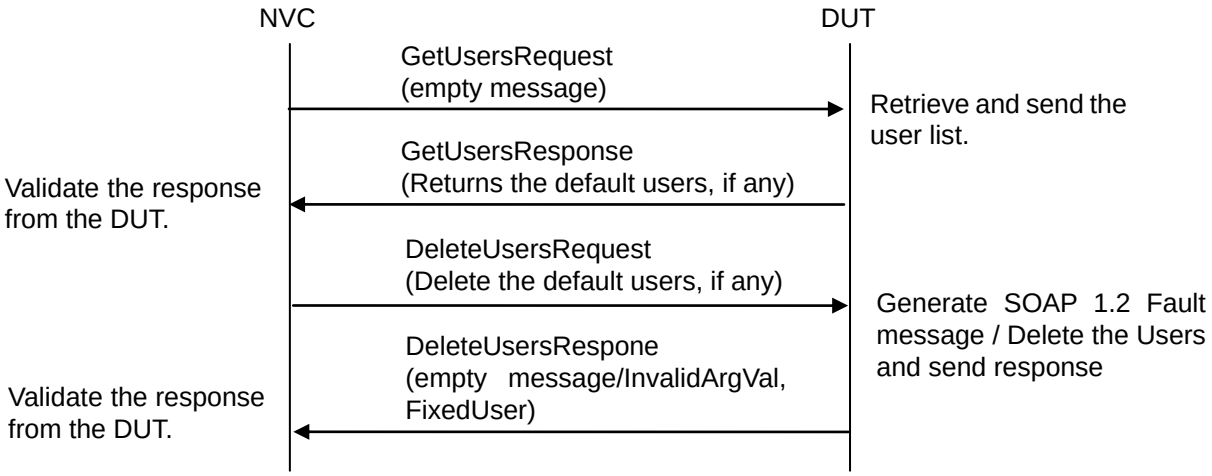
WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: It might be necessary that, NVC may need to operate in administrator mode to execute this test case.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke GetUsersRequest message (empty message), to retrieve the user list from the DUT.
4. Verify that DUT sends GetUsersResponse message (Returns default users, if any).
5. NVC will invoke DeleteUsersRequest message (UserName= Default Users), to delete the default users if there are any.
6. Verify that 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 –

DUT passes 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 DeleteUserResponse message.

Note:

1 DUT may return the default users, if any.

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

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

6.4.7 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.

Requirement Level: MUST

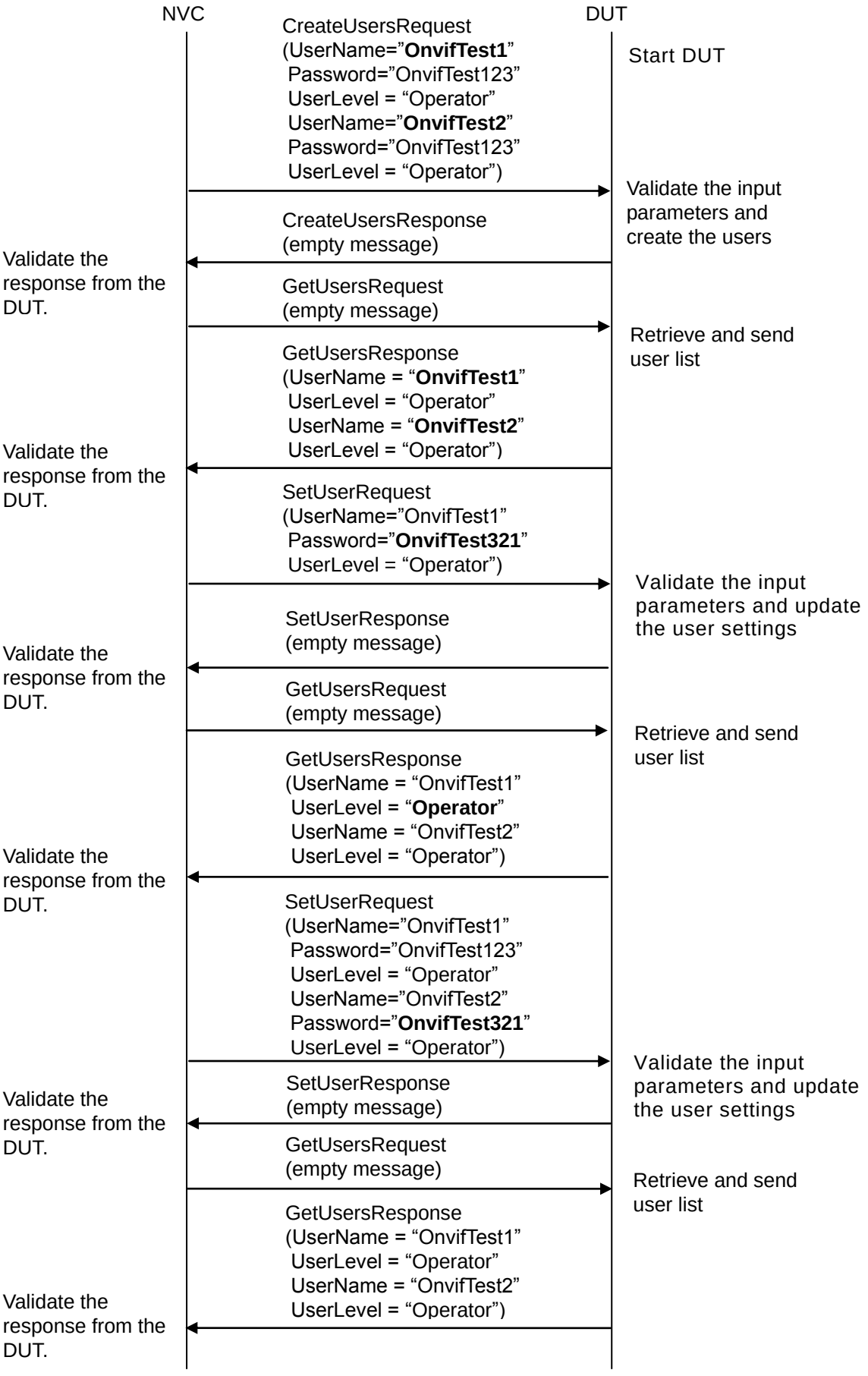
WSDL Reference: devicemgmt.wsdl

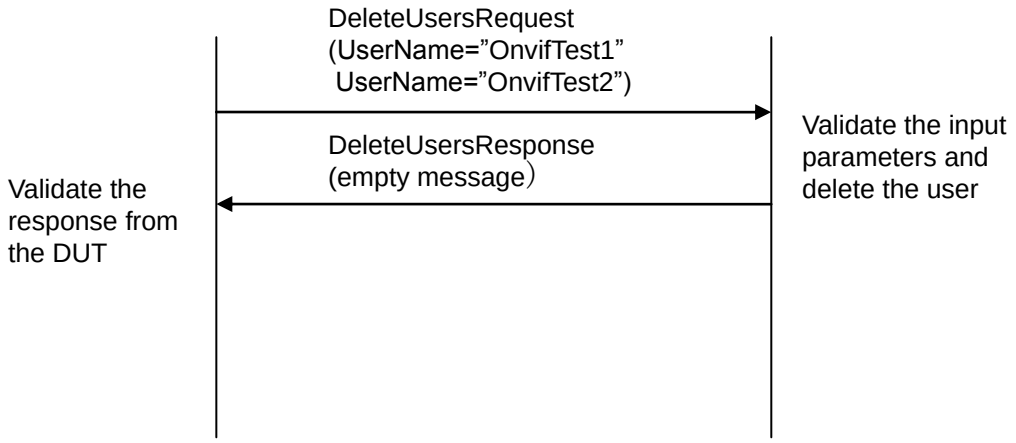
Test Purpose: To verify the behaviour of SetUser command.

Pre-Requisite: It might be necessary that, NVC may need to operate in administrator mode to execute this test case.

Test Configuration: NVC and DUT

Test Sequence:


NVC
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 NVC.
2. Start the DUT.
3. NVC 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 DUT sends CreateUsersResponse message (empty message).
5. NVC 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”).
7. NVC will invoke SetUserRequest message (UserName = “OnvifTest1” Password = “OnvifTest321” UserLevel = “Operator”), to update the user settings in the DUT.
8. Verify that DUT sends SetUserResponse message (empty message).
9. NVC will invoke GetUsersRequest message (empty message), to retrieve the user list from the DUT.
10. Verify that DUT sends GetUsersResponse message (UserName= “OnvifTest1” UserLevel = “Operator”, UserName = “OnvifTest2” UserLevel = “Operator”).
11. NVC 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 DUT sends SetUserResponse message (empty message).
13. NVC will invoke GetUsersRequest (empty message), to retrieve the user list from the DUT.
14. Verify that DUT sends GetUsersResponse message (UserName = OnvifTest1 UserLevel = “Operator”, UserName = OnvifTest2 UserLevel = “User”).
15. NVC will invoke DeleteUsersRequest message (UserName = ”OnvifTest1”, UserName = ”OnvifTest2”) to delete the user in the DUT.
16. Verify that DUT sends DeleteUsersResponse message (Empty Message).

Test Result:

PASS –

DUT passes 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:

1 It may be possible that DUT may return SOAP Fault 1.2 "TooManyUsers" for the CreateUsers command. In this case, test must be executed with the default users if any.

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

6.4.8 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.

Requirement Level: SHOULD

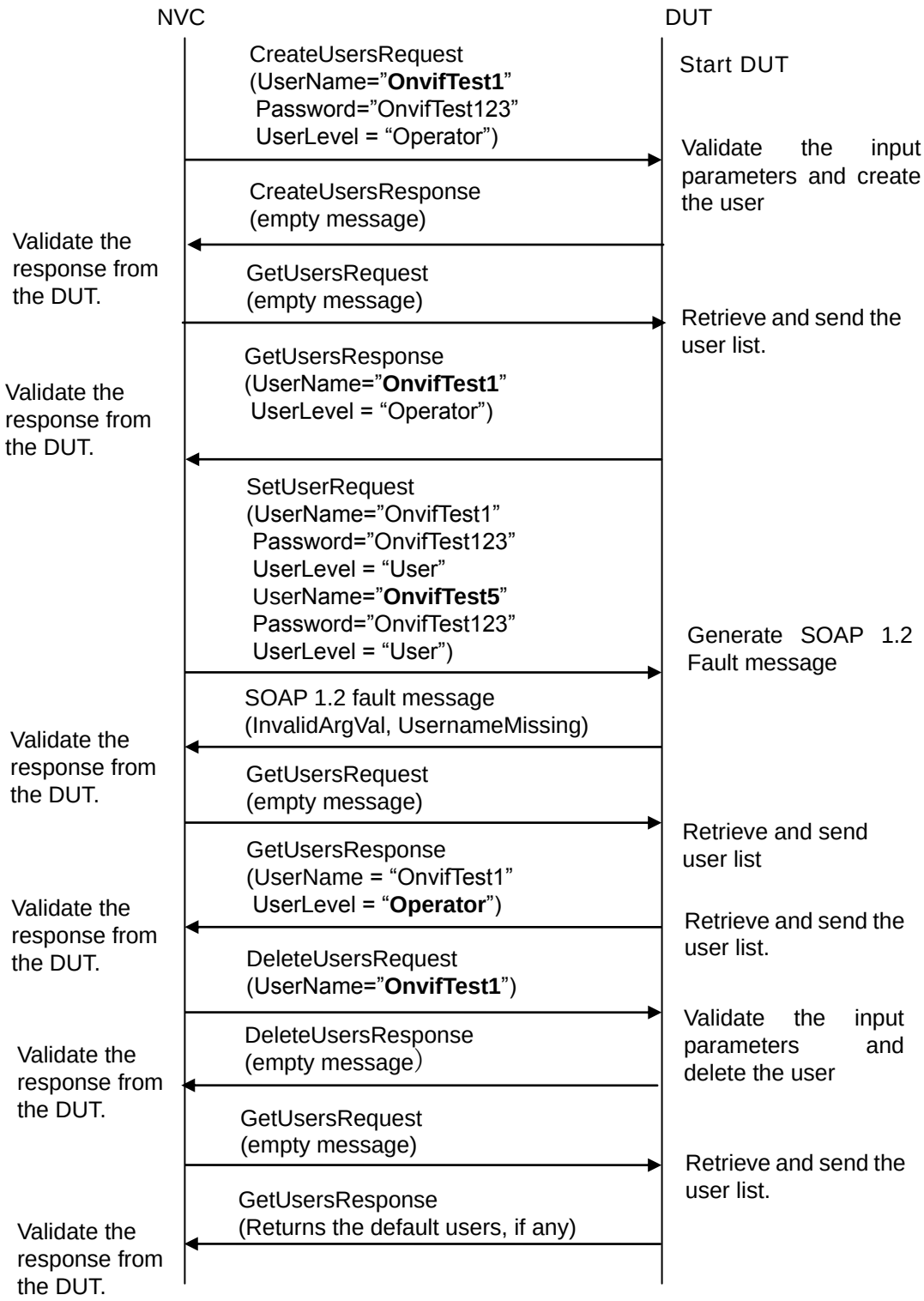
WSDL Reference: devicemgmt.wsdl

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

Pre-Requisite: It might be necessary that, NVC may need to operate in administrator mode to execute this test case.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.

2. Start the DUT.
3. NVC 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. NVC will invoke GetUsersRequest message (empty message) to retrieve the user list form the DUT
6. Verify that DUT sends the GetUsersResponse message (UserName = “OnvifTest1” UserLevel = “Operator”).
7. NVC 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 DUT generates SOAP 1.2 fault message (InvalidArgVal, UsernameMissing).
9. NVC will invoke GetUsersRequest message (empty message), to retrieve the user list from the DUT.
10. Verify that DUT sends the GetUsersResponse message (UserName = “OnvifTest1” UserLevel = “Operator”).
11. NVC will invoke DeleteUsersRequest message (UserName = ”OnvifTest1”), to delete the user in the DUT.
12. Verify that DUT sends DeleteUsersResponse message (Empty Message).
13. NVC will invoke GetUsersRequest message (empty message) to retrieve the user list form the DUT
14. Verify that DUT sends the GetUsersReponse message (returns default users, if any).

Test Result:

PASS –

DUT passes 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:

1 It may be possible that DUT may return SOAP Fault 1.2 “TooManyUsers” for the CreateUsers command. In this case, test must be executed with the default users if any.

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

6.5 I/O

6.5.1 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

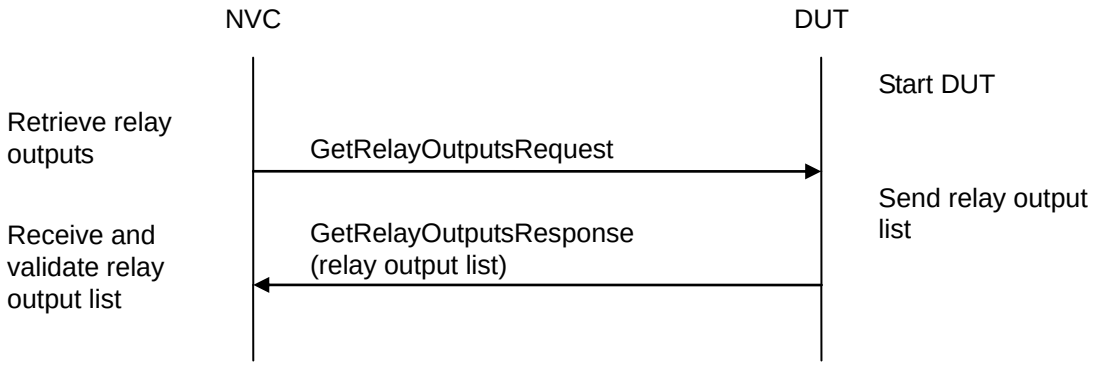
Requirement Level: MUST IF SUPPORTED (Relay Outputs)

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

Pre-Requisite: Relay Outputs supported by DUT.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

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

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send GetRelayOutputsResponse message.

The DUT did not send valid GetRelayOutputsResponse message.

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

The DUT sent an empty list of RelayOutputs.

6.5.2 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

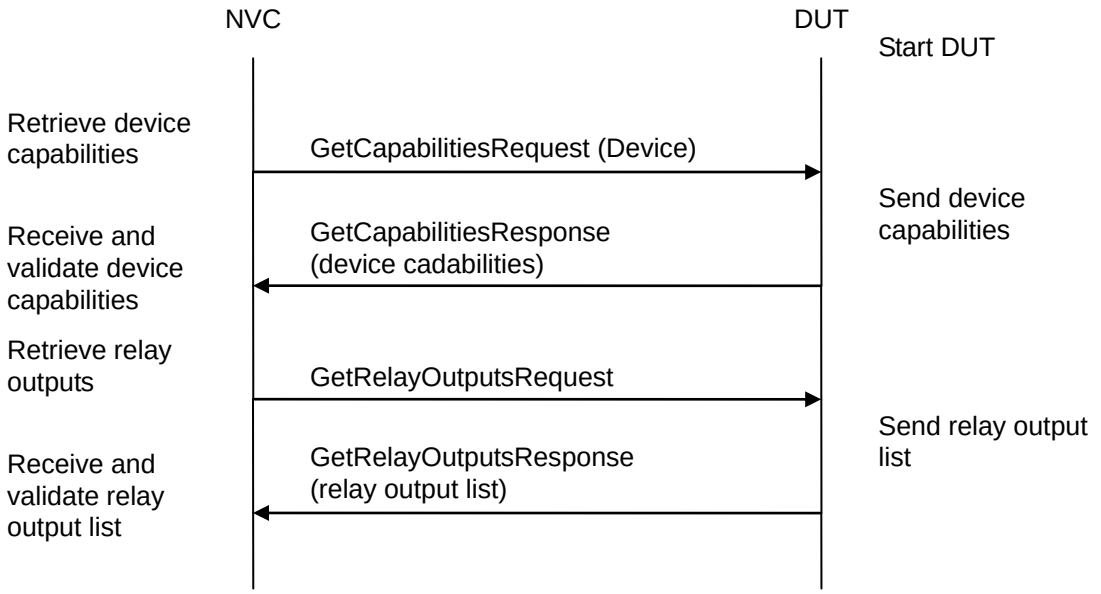
Requirement Level: SHOULD IF SUPPORTED (Relay Outputs)

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

Pre-Requisite: Relay Outputs supported by DUT.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start a DUT.
3. NVC invokes GetCapabilitiesRequest message (**Device**) to retrieve device capabilities.
4. Verify the GetCapabilitiesResponse message from the DUT.
5. NVC invokes GetRelayOutputsRequest message to retrieve relay outputs supported by the DUT.
6. Verify the GetRelayOutputsResponse message from the DUT.

Test Result:

PASS –

DUT passes 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 count in GetRelayOutputsResponse message that differ from Device.IO.RelayOutputs from GetCapabilitiesResponse message.

6.5.3 IO COMMAND SETRELAYOUTPUTSETTINGS – INVALID TOKEN

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

Test Case ID: DEVICE-5-1-4

ONVIF Core Specification Coverage: Set relay output settings

Command under Test: SetRelayOutputSettings

WSDL Reference: devicemgmt.wsdl

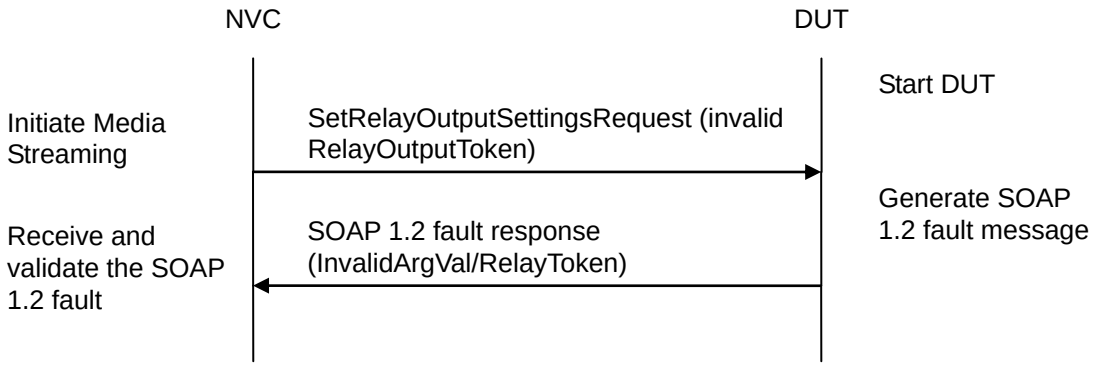
Requirement Level: MUST IF SUPPORTED (Relay Outputs)

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

Pre-Requisite: Relay Outputs supported by DUT.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start a DUT.
3. NVC invokes SetRelayOutputSettingsRequest message (**invalid RelayOutputToken**).
4. The DUT will generate SOAP 1.2 fault message (**InvalidArgVal/RelayToken**)

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send SOAP 1.2 fault message.

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

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

6.5.4 IO COMMAND SETRELAYOUTPUTSTATE – INVALID TOKEN

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

Test Case ID: DEVICE-5-1-10

ONVIF Core Specification Coverage: Trigger relay output

Command under Test: SetRelayOutputState

WSDL Reference: devicemgmt.wsdl

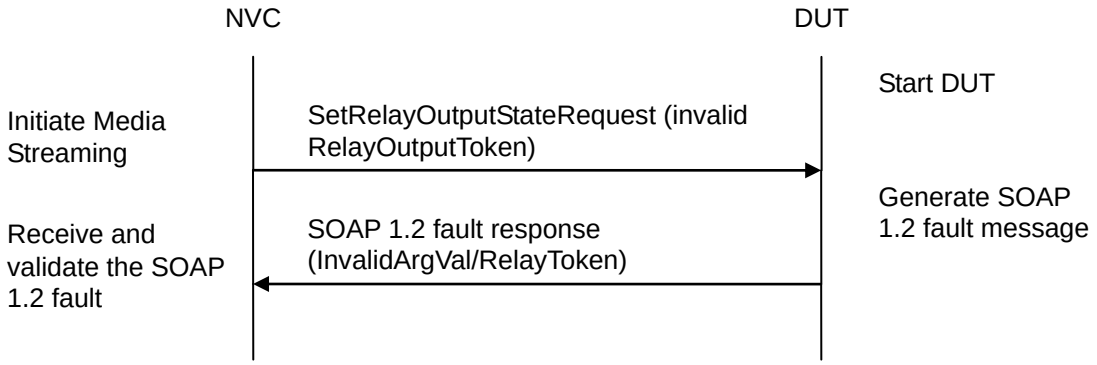
Requirement Level: MUST IF SUPPORTED (Relay Outputs)

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

Pre-Requisite: Relay Outputs supported by DUT.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start a DUT.
3. NVC invokes SetRelayOutputStateRequest message (**invalid RelayOutputToken**).
4. The DUT will generate SOAP 1.2 fault message (**InvalidArgVal/RelayToken**).

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send SOAP 1.2 fault message.

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

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

6.6 Namespace Handling

6.6.1 DEVICE MANAGEMENT - NAMESPACES (DEFAULT NAMESPASES 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

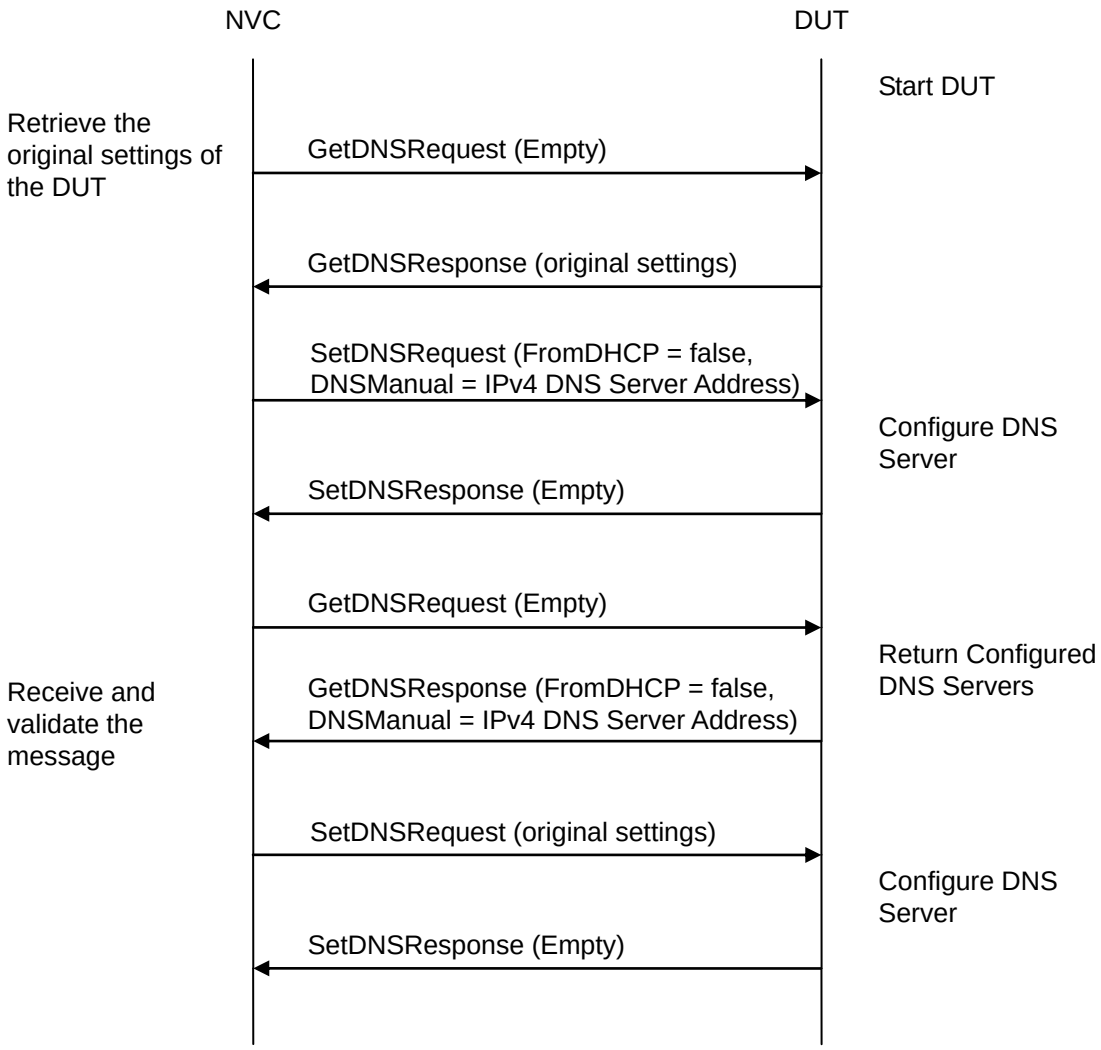
Requirement Level: MUST

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start a DUT.
3. NVC invokes GetDNSRequest message to retrieve the original settings of the DUT.
4. Verify the GetDNSResponse message from the DUT.
5. NVC 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. NVC 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. NVC will invoke SetDNS Request message to restore the original settings of the DUT.

Test Result:

PASS –

DUT passes 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 must have default namespaces definition in each tag (see examples in Annex A.11).

6.6.2 DEVICE MANAGEMENT - NAMESPACES (DEFAULT NAMESPASES 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

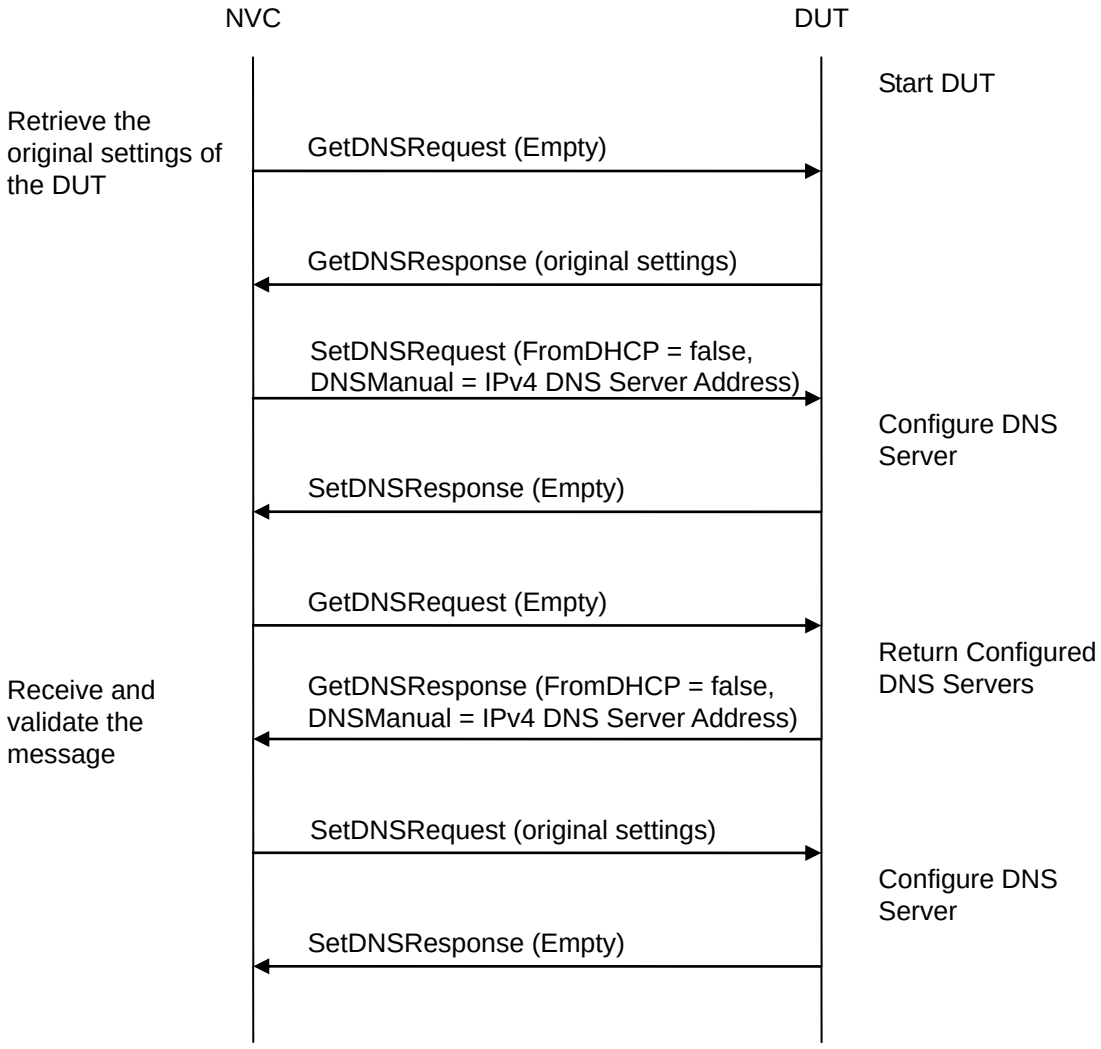
Requirement Level: MUST

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start a DUT.
3. NVC invokes GetDNSRequest message to retrieve the original settings of the DUT.
4. Verify the GetDNSResponse message from the DUT.
5. NVC 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. NVC 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. NVC will invoke SetDNS Request message to restore the original settings of the DUT.

Test Result:

PASS –

DUT passes 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 must have default namespaces definition in parent tag (see examples in Annex A.11).

6.6.3 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

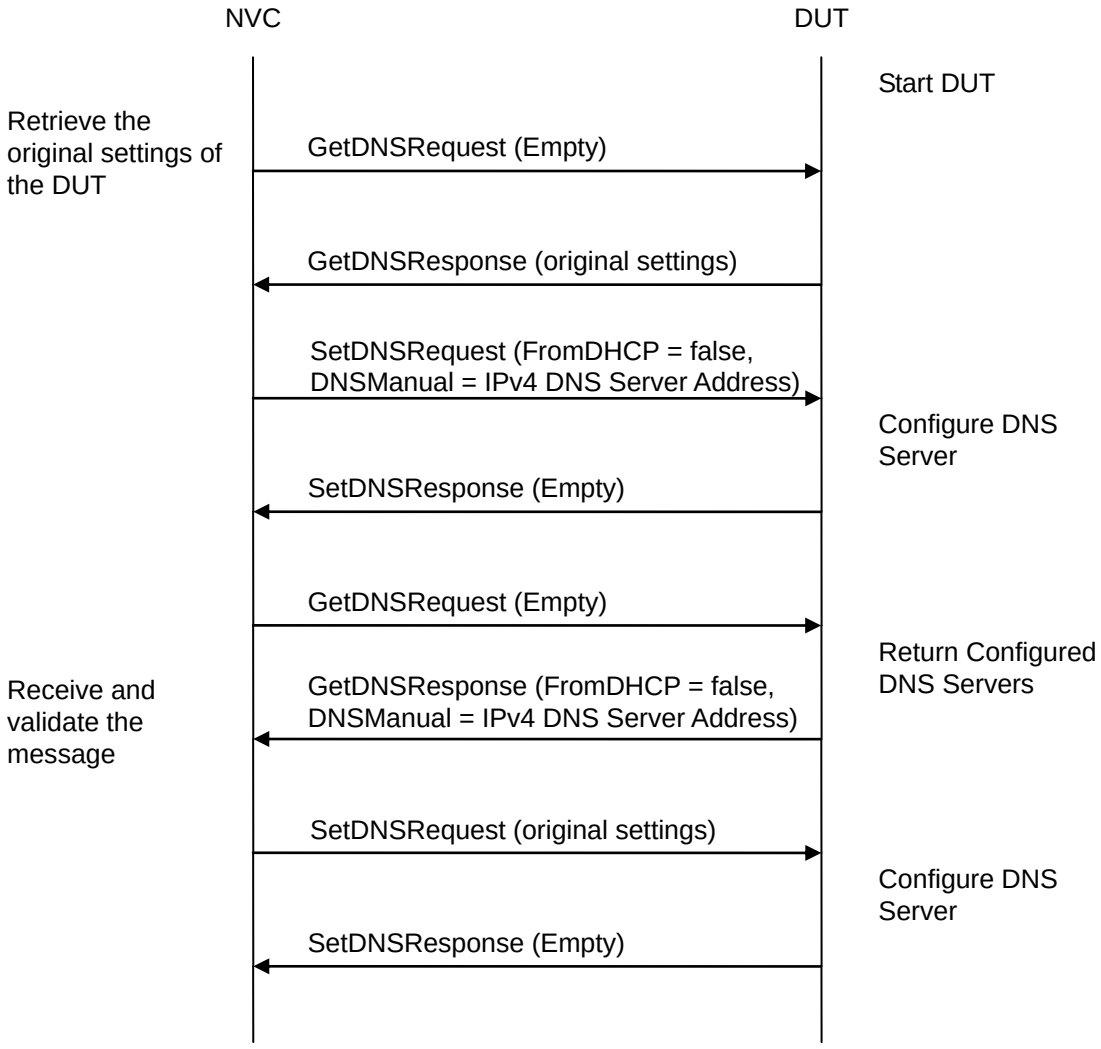
Requirement Level: MUST

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start a DUT.
3. NVC invokes GetDNSRequest message to retrieve the original settings of the DUT.
4. Verify the GetDNSResponse message from the DUT.
5. NVC 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. NVC 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. NVC will invoke SetDNS Request message to restore the original settings of the DUT.

Test Result:

PASS –

DUT passes 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 must have namespaces definition with not standard prefixes (see examples in Annex A.11).

6.6.4 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

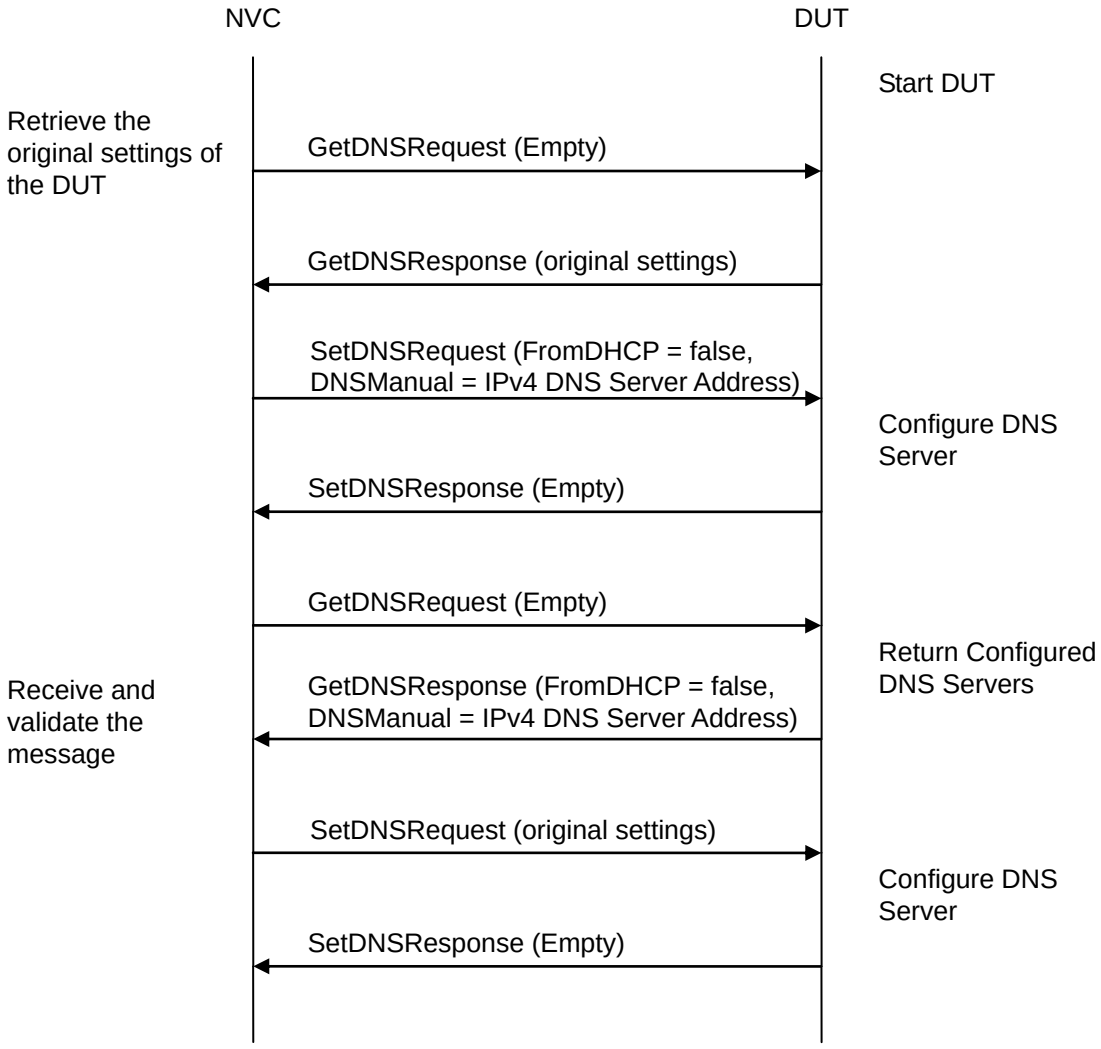
Requirement Level: MUST

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start a DUT.
3. NVC invokes GetDNSRequest message to retrieve the original settings of the DUT.
4. Verify the GetDNSResponse message from the DUT.
5. NVC 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. NVC 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. NVC will invoke SetDNS Request message to restore the original settings of the DUT.

Test Result:

PASS –

DUT passes 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 must have namespaces definition with different prefixes for the same namespace (see examples in Annex A.11).

6.6.5 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

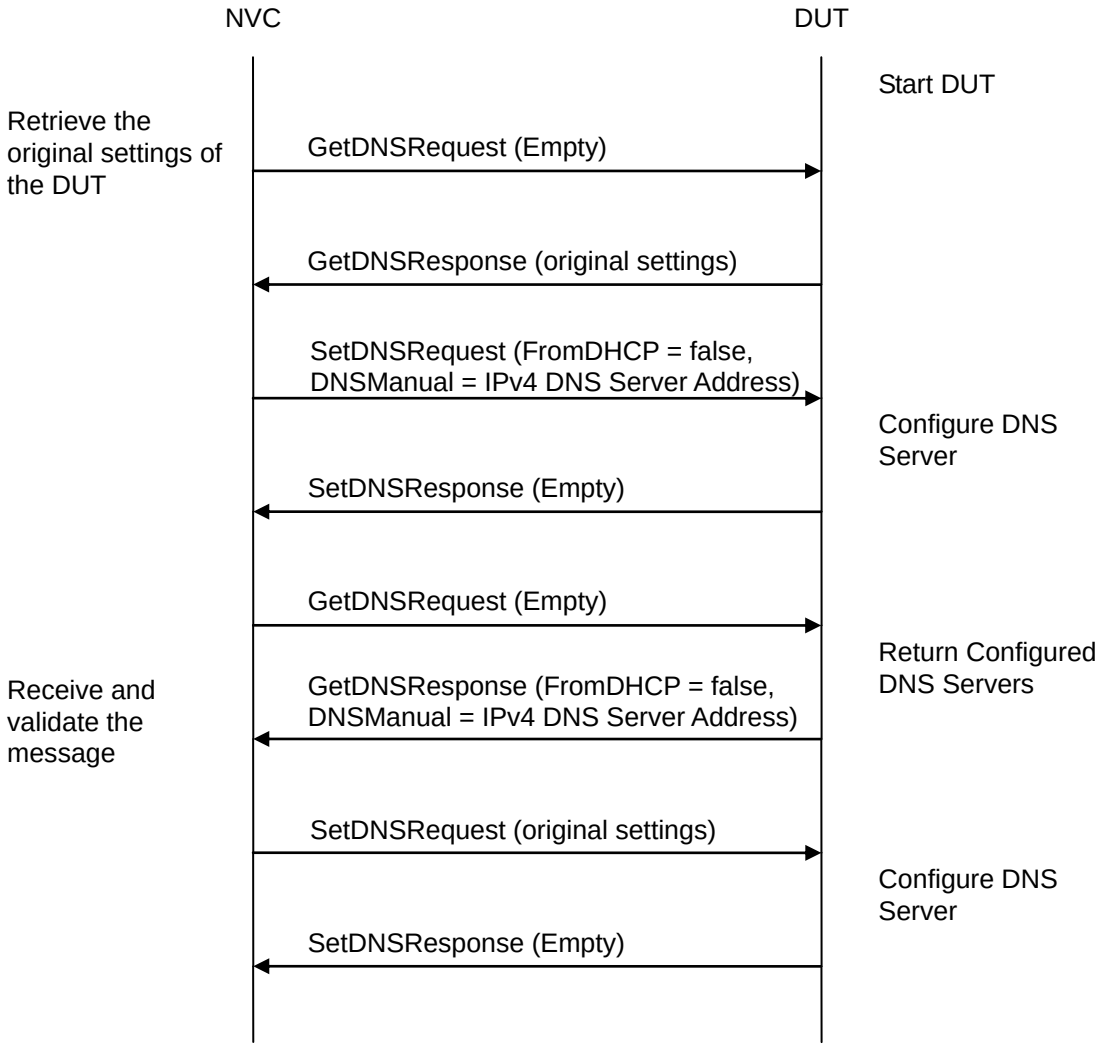
Requirement Level: MUST

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

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start a DUT.
3. NVC invokes GetDNSRequest message to retrieve the original settings of the DUT.
4. Verify the GetDNSResponse message from the DUT.
5. NVC 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. NVC 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. NVC will invoke SetDNS Request message to restore the original settings of the DUT.

Test Result:

PASS –

DUT passes 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 must have namespaces definition with the same prefixes for different namespaces (see examples in Annex A.11).

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-1-1-1

ONVIF Core Specification Coverage: GetEventProperties

Command Under Test: GetEventProperties

WSDL Reference: event.wsdl

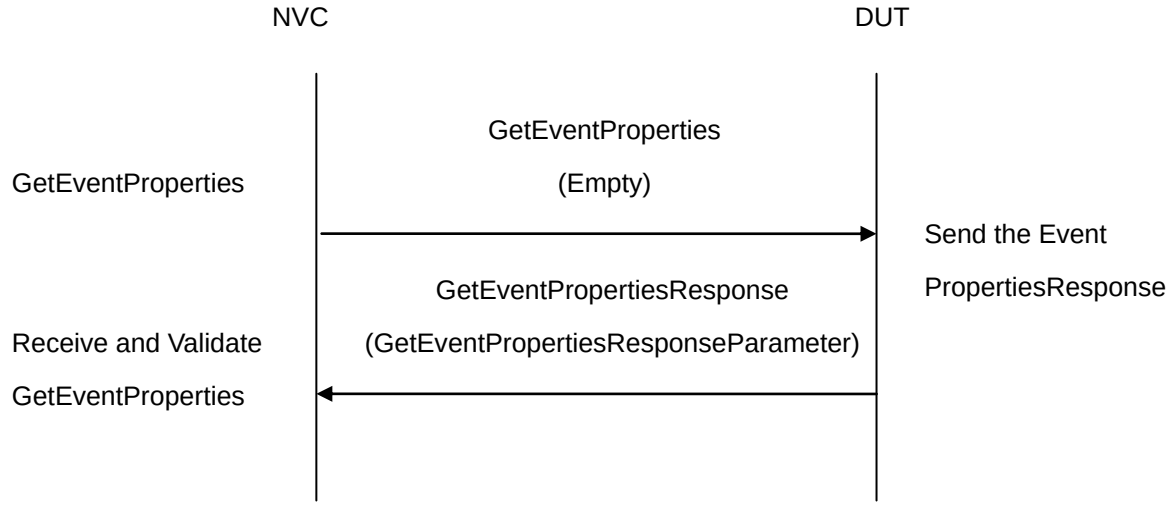
Requirement Level: MUST

Test Purpose: To verify DUT GetEventProperties command

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke GetEventProperties 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 –

DUT passes 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

7.2 Basic Notification Interface

7.2.1 BASIC NOTIFICATION INTERFACE - SUBSCRIBE

Test Label: Event handling SUBSCRIBE

Test Case ID: EVENT-2-1-1

ONVIF Core Specification Coverage: Basic Notification Interface

Command Under Test: Subscribe

WSDL Reference: event.wsdl

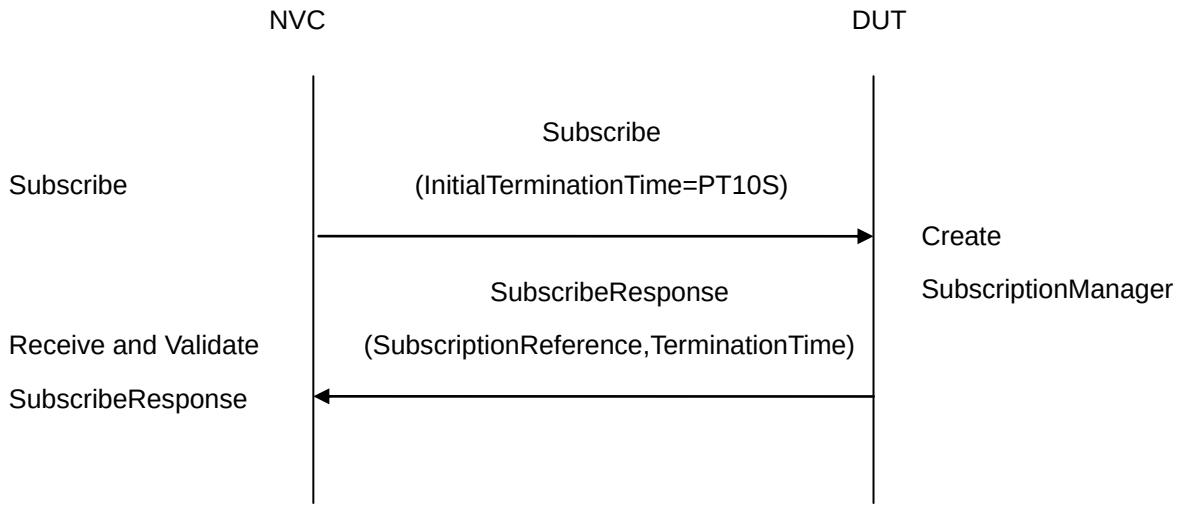
Requirement Level: MUST

Test Purpose: To verify DUT Subscribe command

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.

2. Start the DUT.
3. NVC 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 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 –

DUT passes 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.

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

If DUT can not accept the set value to an InitialTerminationTime, NVC retries to send the Subscribe request with MinimumTime value which is contained in UnacceptableInitialTerminationTimeFault.

7.2.2 BASIC NOTIFICATION INTERFACE - INVALID MESSAGE CONTENT FILTER

Test Label: Event handling SUBSCRIBE INVALID MESSAGE CONTENT FILTER

Test Case ID: EVENT-2-1-2

ONVIF Core Specification Coverage: Basic Notification Interface

Command Under Test: Subscribe

WSDL Reference: event.wsdl

Requirement Level: SHOULD

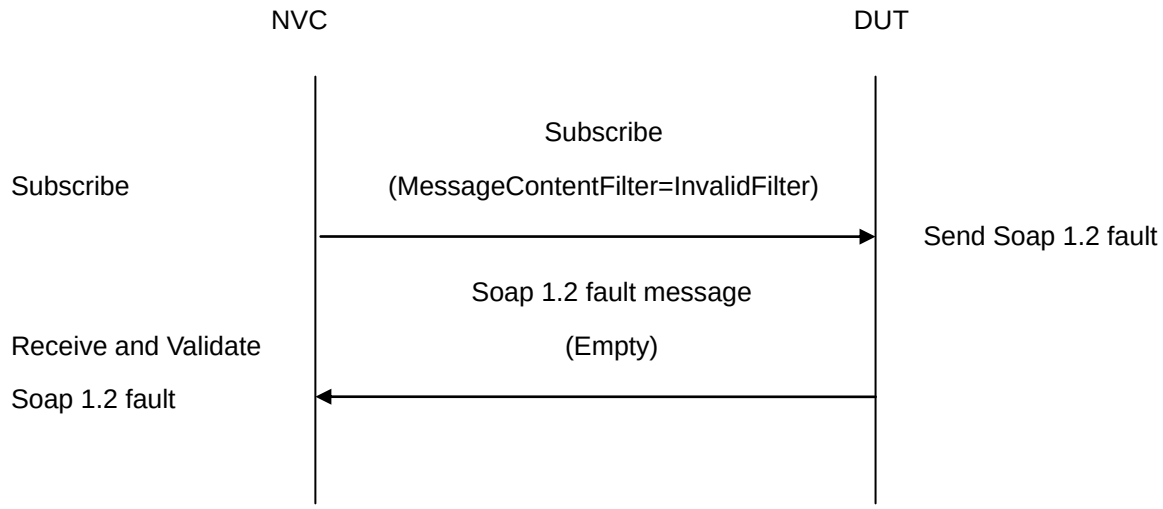
Test Purpose:

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: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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 –

DUT passes all assertions.

FAIL –

The DUT did not send an “InvalidFilterFault” or “InvalidMessageContentExpressionFault” fault message.

The DUT did not send a valid fault message

7.2.3 BASIC NOTIFICATION INTERFACE - INVALID TOPIC EXPRESSION

Test Label: Event handling SUBSCRIBE INVALID FILTER-TOPIC EXPRESSION

Test Case ID: EVENT-2-1-3

ONVIF Core Specification Coverage: Basic Notification Interface

Command Under Test: Subscribe

WSDL Reference: event.wsdl

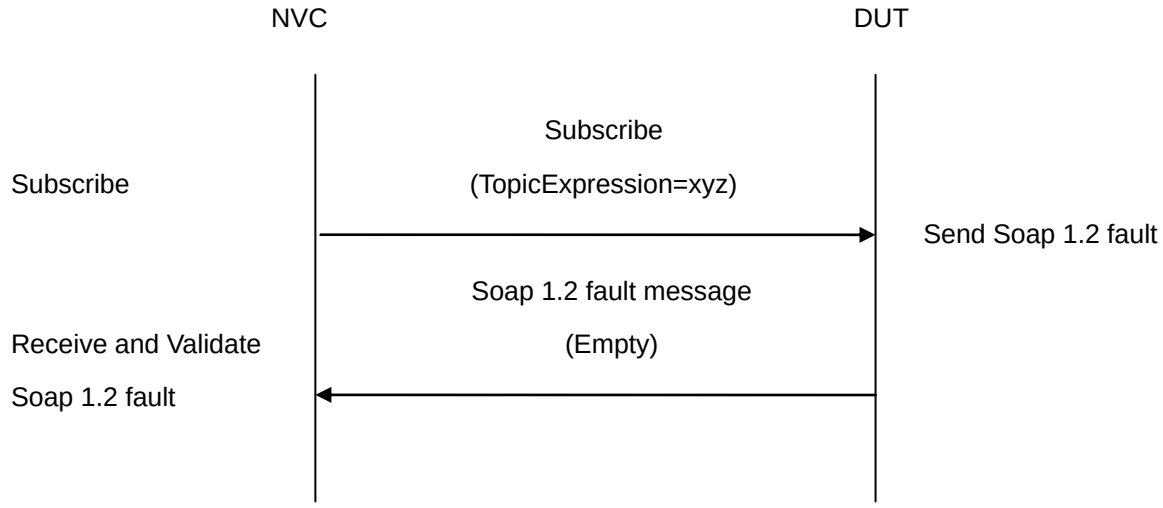
Requirement Level: SHOULD

Test Purpose: To verify that a correct error message "InvalidFilterFault" or “TopicNotSupported” is returned if a Subscribe Request with an invalid Topic Expression is invoked.

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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” fault message.
5. Validate the fault message (valid utc time, valid description)

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send an “InvalidFilterFault” or “TopicNotSupported” fault message.

The DUT did not send a valid fault message

7.2.4 BASIC NOTIFICATION INTERFACE - RENEW

Test Label: Event handling Renew

Test Case ID: EVENT-2-1-4

ONVIF Core Specification Coverage: Basic Notification Interface

Command Under Test: Subscribe, Renew

WSDL Reference: event.wsdl

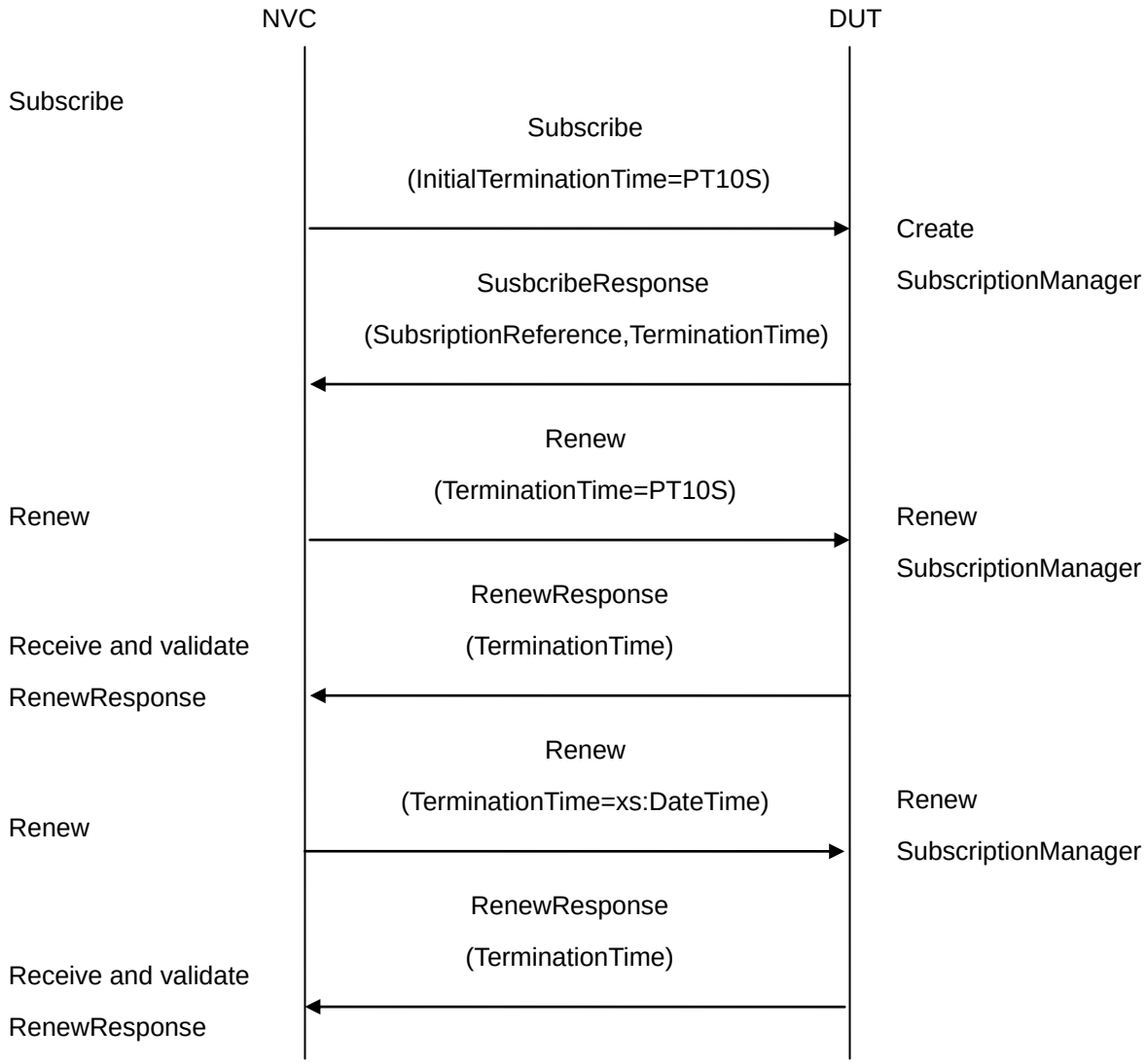
Requirement Level: MUST

Test Purpose: To verify Renew command

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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. NVC 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. NVC 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>=CurrentTime+ TerminationTime)

Test Result:

PASS –

DUT passes 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 RenewResponses

The DUT did not send valid values for CurrentTime and TerminationTime

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

If DUT can not accept the set value to an InitialTerminationTime, NVC retries to send the Subscribe request with MinimumTime value which is contained in UnacceptableInitialTerminationTimeFault.

If DUT can not accept the set value to a TerminationTime, NVC retries to send the Renew request MinimumTime value which is contained in UnacceptableTerminationTimeFault.

7.2.5 BASIC NOTIFICATION INTERFACE - UNSUBSCRIBE

Test Label: Event handling UNSUBSCRIBE

Test Case ID: EVENT-2-1-5

ONVIF Core Specification Coverage: Basic Notification Interface

Command Under Test: Subscribe, Unsubscribe

WSDL Reference: event.wsdl

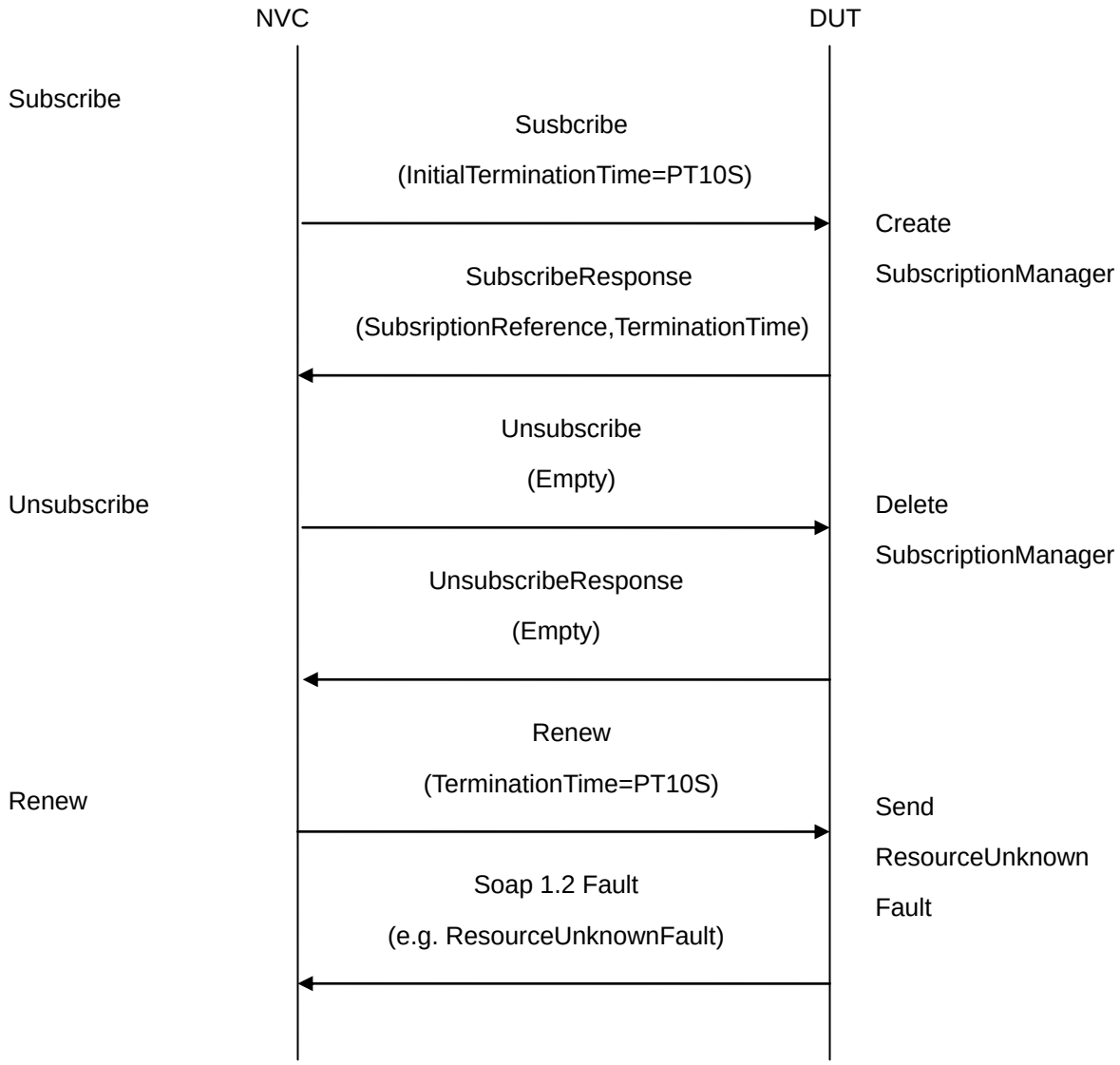
Requirement Level: MUST

Test Purpose: To verify Unsubscribe command

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke Subscribe message (InitialTerminationTime=PT10S) to instantiate an SubscriptionManager
4. Verify that the DUT sends a SubscribeResponse with valid values for SubscriptionManager, CurrentTime and TerminationTime.
5. NVC will invoke Unsubscribe command

6. Verify that the DUT sends a UnsubscribeResponse
7. NVC 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 –

DUT passes 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

Note: If DUT can not accept the set value to an InitialTerminationTime, NVC retries to send the Subscribe request with MinimumTime value which is contained in UnacceptableInitialTerminationTimeFault.

7.2.6 BASIC NOTIFICATION INTERFACE - RESOURCE UNKNOWN

Test Label: Event handling RESOURCE UNKNOWN

Test Case ID: EVENT-2-1-6

ONVIF Core Specification Coverage: Basic Notification Interface

Command Under Test: Subscribe, Unsubscribe

WSDL Reference: event.wsdl

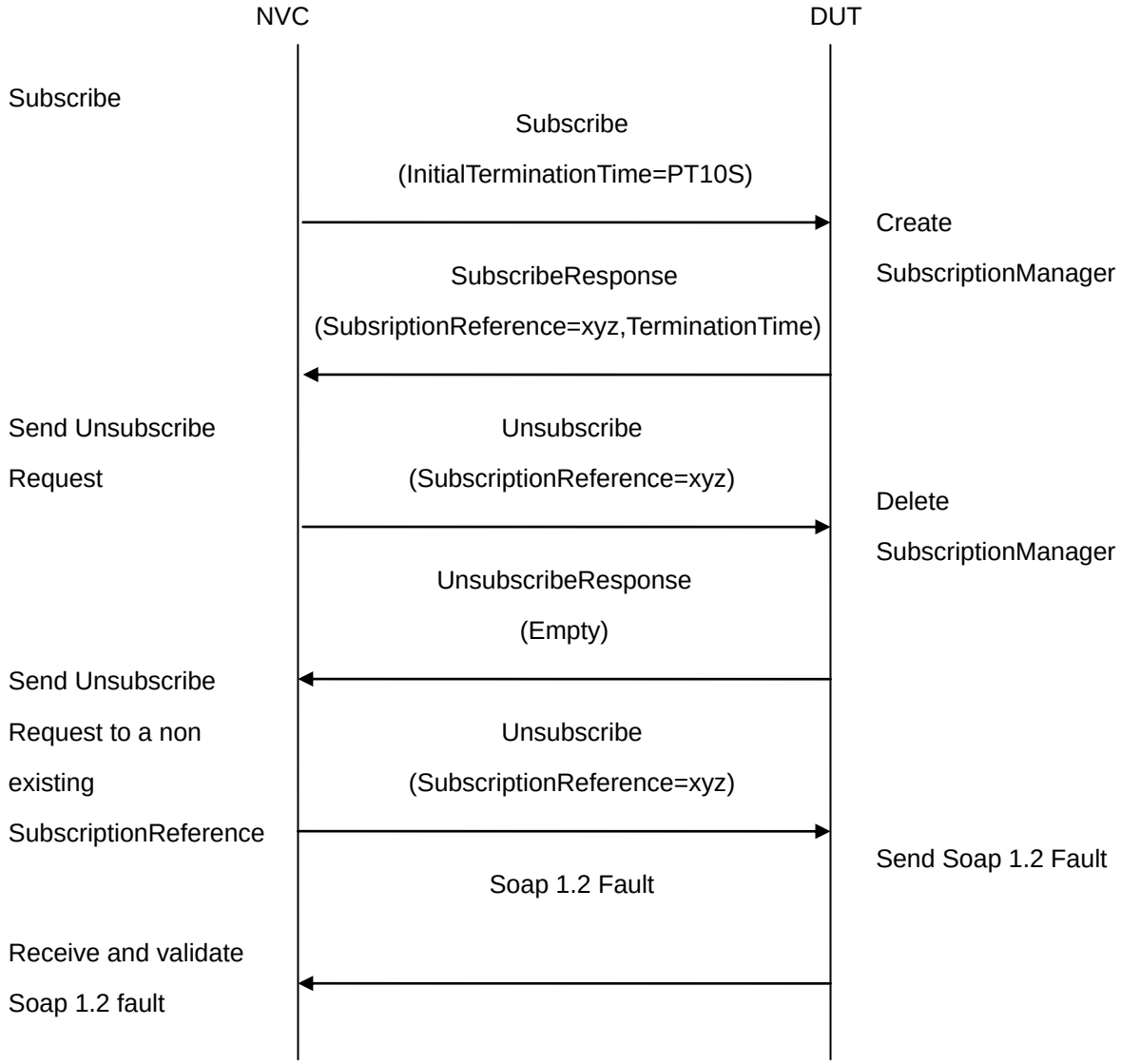
Requirement Level: SHOULD

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: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke Subscribe message with a 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. NVC will invoke Unsubscribe command to delete the SubscriptionManager
6. Verify that the DUT sends a UnsubscribeResponse
7. Send a 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 –

DUT passes 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

Note: If DUT can not accept the set value to an InitialTerminationTime, NVC retries to send the Subscribe request with MinimumTime value which is contained in UnacceptableInitialTerminationTimeFault.

7.2.7 BASIC NOTIFICATION INTERFACE - NOTIFY

Test Label: Event handling NOTIFY

Test Case ID: EVENT-2-1-7

ONVIF Core Specification Coverage: Basic Notification Interface, SetSynchronizationPoint

Command Under Test: Subscribe, Unsubscribe, SetSynchronizationPoint, Notify

WSDL Reference: event.wsdl

Requirement Level: MUST

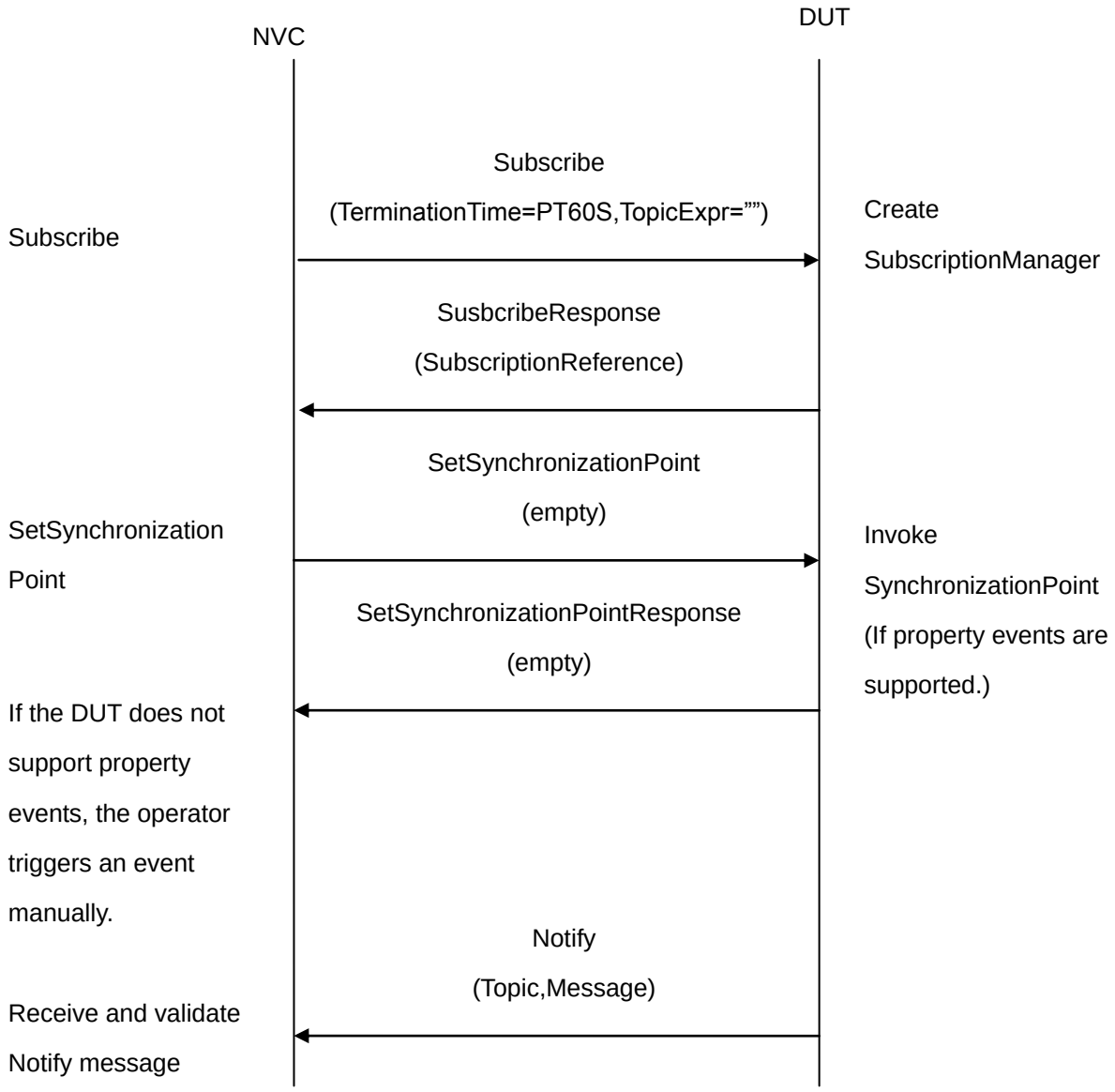
Test Purpose: To verify Notify message

Pre-Requisite: The DUT MUST provide at least one event.

The test operator has to ensure that the event is triggered and sent out. NVC will invoke a SetSynchronizationPoint request. If the DUT does not support property events of if it is not possible to invoke a SetSynchronizationPoint, the test operator has to trigger the event manually.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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 an 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 –

DUT passes 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

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 on how to compose Subscribe when the client is interested in receiving all event supported by the DUT.

7.2.8 BASIC NOTIFICATION INTERFACE - NOTIFY FILTER

Test Label: Event handling NOTIFY FILTER

Test Case ID: EVENT-2-1-8

ONVIF Core Specification Coverage: Basic Notification Interface, SetSynchronizationPoint

Command Under Test: GetEventProperties, Subscribe, Unsubscribe, SetSynchronizationPoint, Notify

WSDL Reference: event.wsdl

Requirement Level: MUST

Test Purpose:

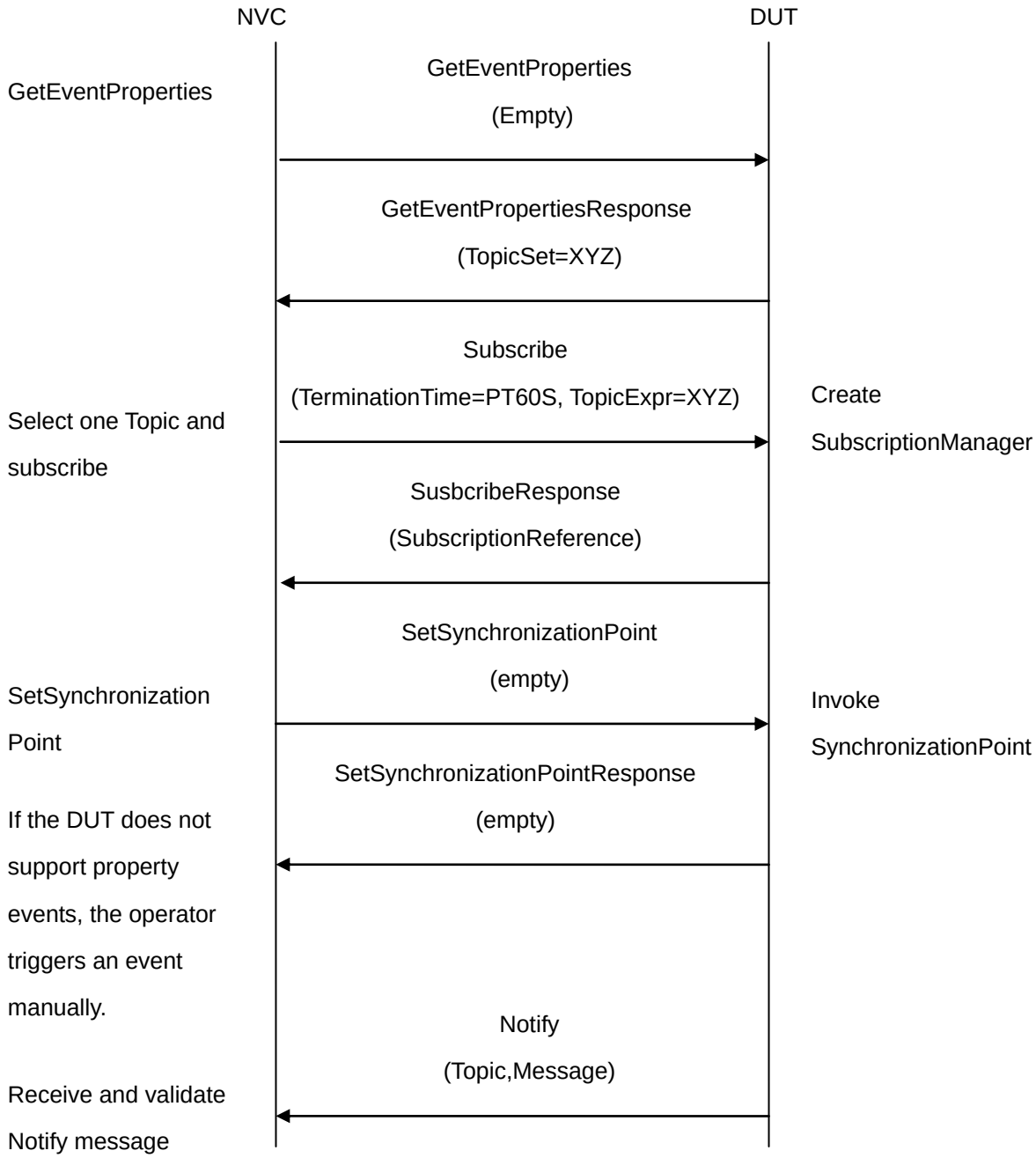
To verify that the device sends Notification messages; to verify if the DUT handles event filtering in a correct way.

Pre-Requisite: The DUT MUST provide at least one event.

The test operator has to ensure that the event is triggered and sent out. NVC will invoke a SetSynchronizationPoint request. If the DUT doesn't support property event or if it is not possible to invoke a SetSynchronizationPoint, the test operator has to trigger the event manually.

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke GetEventProperties command
4. Verify that the DUT sends a GetEventPropertiesResponse, select one Topic.
5. NVC will invoke Subscribe with this Topic as Filter and a 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 DUT sends SetSynchronizationPointResponse
9. If the DUT does not support property events, the operator has to trigger an event manually.
10. Verify that 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 –

DUT passes 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

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

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-3-1-1

ONVIF Core Specification Coverage: CreatePullPointSubscription

Command Under Test: CreatePullPointSubscription

WSDL Reference: event.wsdl

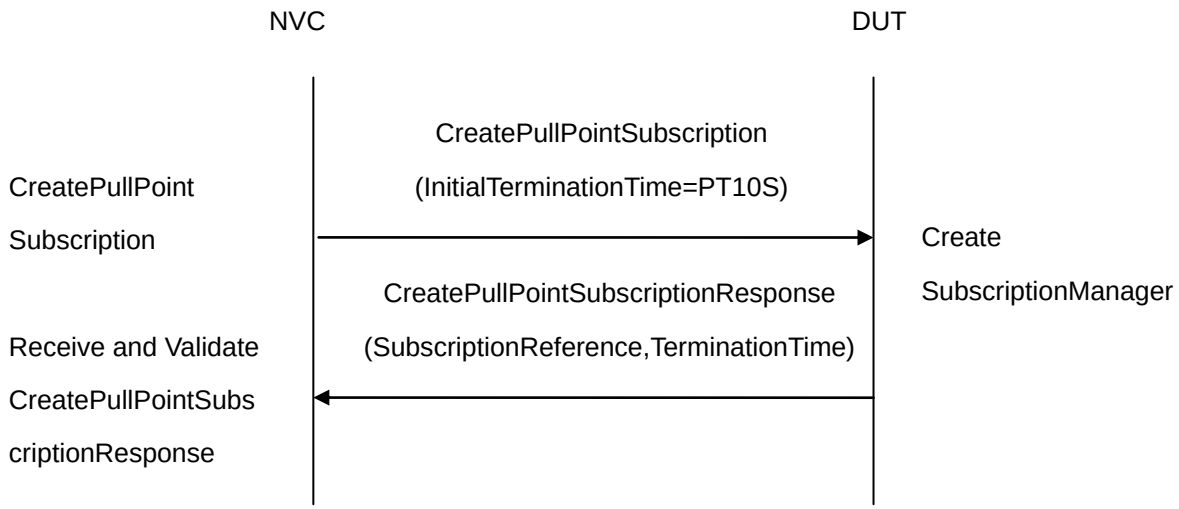
Requirement Level: MUST

Test Purpose: To verify DUT CreatePullPointSubscription command

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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 DUT sends CreatePullPointSubscriptionResponse message
5. Validate that valid values for SubscriptionReference CurrentTime and TerminationTime are returned (TerminationTime>=CurrentTime+InitialTerminationTime)

Test Result:

PASS –

DUT passes 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.

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

If DUT can not accept the set value to an InitialTerminationTime, NVC retries to send the CreatePullpointSubscription request with MinimumTime value which is contained in UnacceptableInitialTerminationTime fault.

7.3.2 REALTIME PULLPOINT SUBSCRIPTION - INVALID MESSAGE CONTENT FILTER

Test Label: event handling CREATE PULL POINT SUBSCRIPTION – INVALID MESSAGE CONTENT FILTER

Test Case ID: EVENT-3-1-2

ONVIF Core Specification Coverage: CreatePullPointSubscription

Command Under Test: CreatePullPointSubscription

WSDL Reference: event.wsdl

Requirement Level: SHOULD

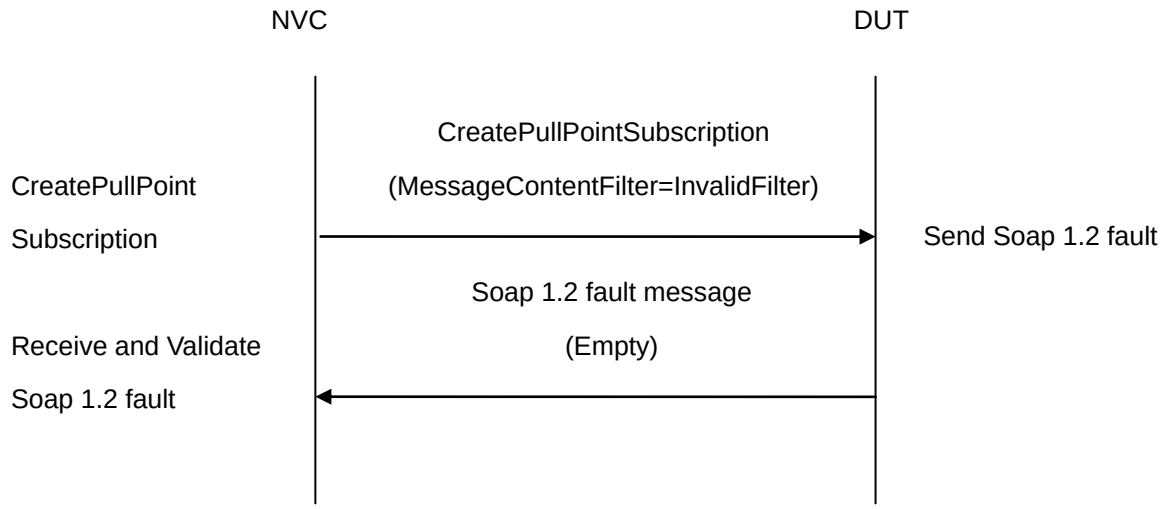
Test Purpose:

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: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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 –

DUT passes all assertions.

FAIL –

The DUT did not send an “InvalidFilterFault” or “InvalidMessageContentExpressionFault” fault message.

The DUT did not send a valid fault message

7.3.3 REALTIME PULLPOINT SUBSCRIPTION - INVALID TOPIC EXPRESSION

Test Label: Event handling REALTIME PULLPOINT INVALID FILTER-TOPIC EXPRESSION

Test Case ID: EVENT-3-1-3

ONVIF Core Specification Coverage: CreatePullPointSubscription

Command Under Test: CreatePullPointSubscription

WSDL Reference: event.wsdl

Requirement Level: SHOULD

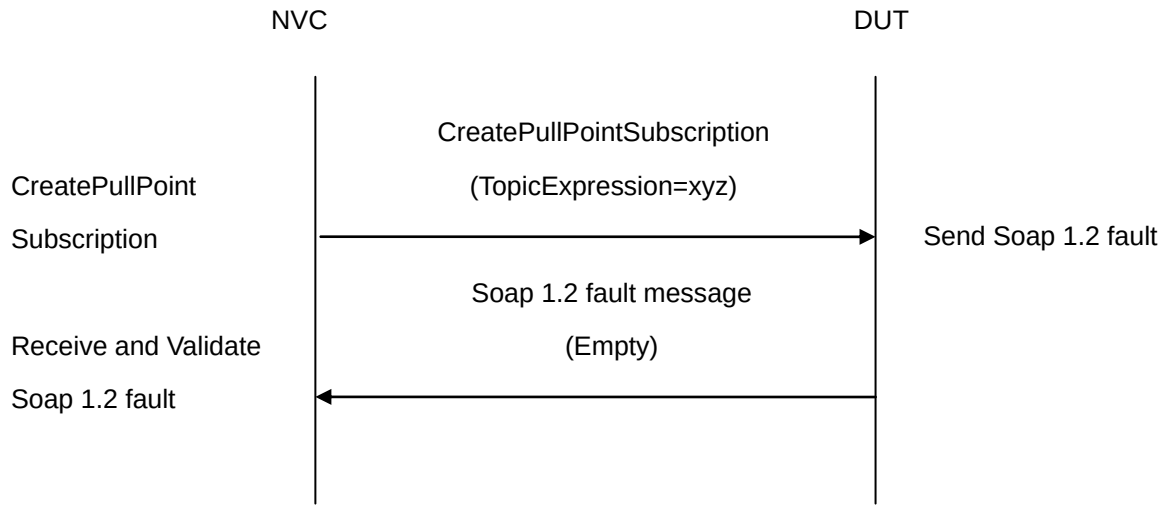
Test Purpose:

To verify that a correct error message "InvalidFilterFault" or “TopicNotSupported” is returned if a CreatePullPointSubscription command with an invalid TopicExpression is invoked.

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke CreatePullPointSubscription message with an invalid Topic Expression “xyz”.
4. Verify that the DUT generates an “InvalidFilterFault” or “TopicNotSupported” fault message.
5. Validate valid the fault message (valid utc time, valid description)

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send an “InvalidFilterFault” or “TopicNotSupported” fault message.

The DUT did not send a valid fault message

7.3.4 REALTIME PULLPOINT SUBSCRIPTION - RENEW

Test Label: Event handling RealtimePullPoint Renew

Test Case ID: EVENT-3-1-4

ONVIF Core Specification Coverage: Basic Notification Interface, CreatePullPointSubscription

Command Under Test: CreatePullPointSubscription, Renew

WSDL Reference: event.wsdl

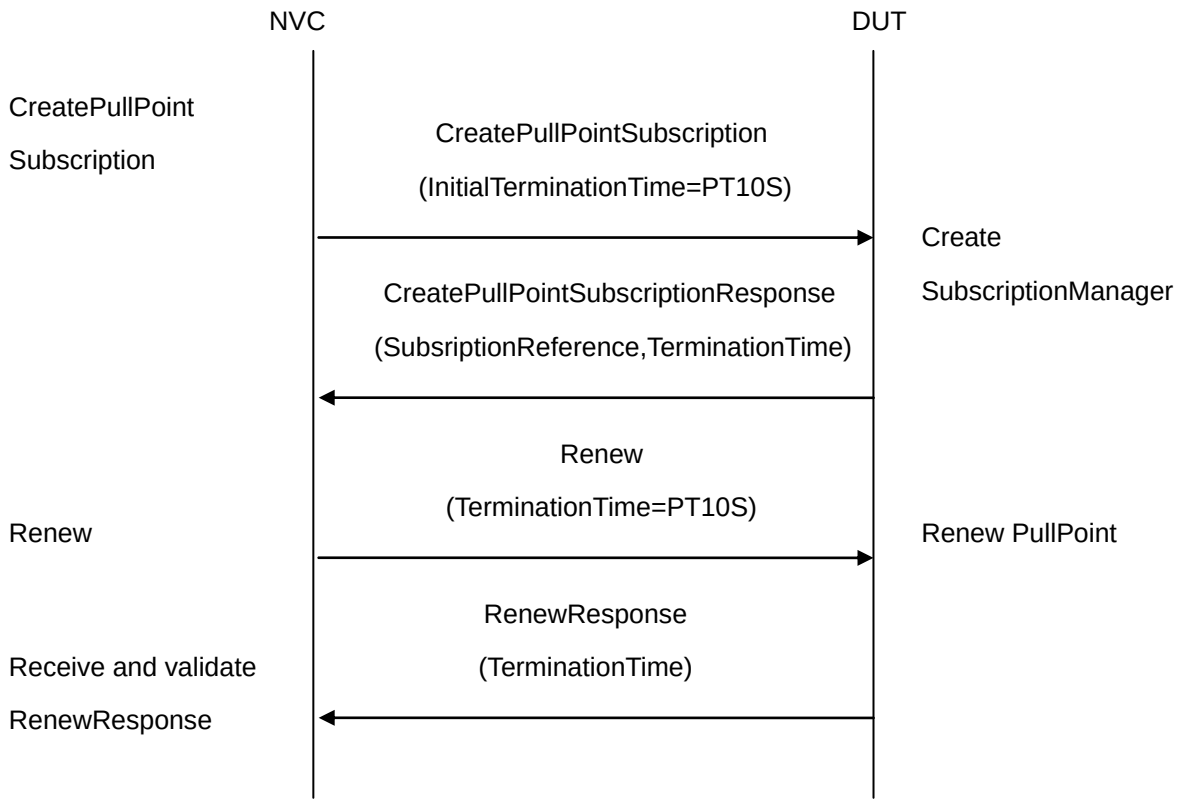
Requirement Level: MUST

Test Purpose: To verify Renew command

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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. NVC 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 –

DUT passes 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

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

If DUT can not accept the set value to an InitialTerminationTime, NVC retries to send the CreatePullpointSubscription request with MinimumTime value which is contained in UnacceptableInitialTerminationTime fault.

7.3.5 REALTIME PULLPOINT SUBSCRIPTION - UNSUBSCRIBE

Test Label: Event handling RealTime PullPoint UNSUBSCRIBE

Test Case ID: EVENT-3-1-5

ONVIF Core Specification Coverage: Basic Notification Interface, CreatePullPointSubscription

Command Under Test: CreatePullPointSubscription, Unsubscribe, Renew

WSDL Reference: event.wsdl

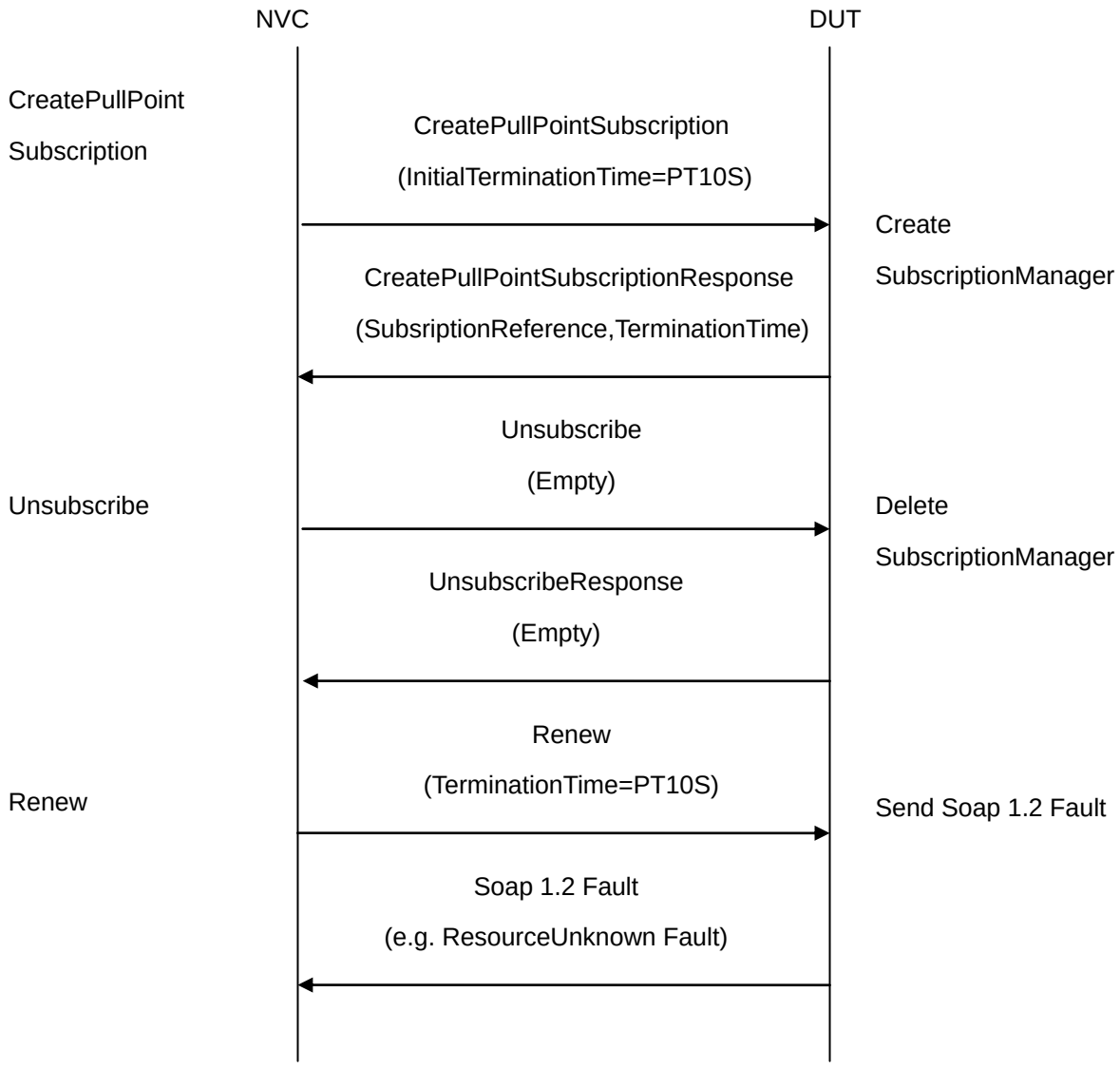
Requirement Level: MUST

Test Purpose: To verify Unsubscribe command

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC will invoke CreatePullPointSubscription message (InitialTerminationTime=PT10S) to instantiate an SubscriptionManager
4. Verify that the DUT sends a CreatePullPointSubscriptionResponse. Validate CurrentTime and TerminationTime

5. NVC will invoke Unsubscribe command to terminate the SubscriptionManager
6. Verify that the DUT sends a UnsubscribeResponse
7. NVC 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 –

DUT passes 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.

Note: If DUT can not accept the set value to an InitialTerminationTime, NVC retries to send the CreatePullpointSubscription request with MinimumTime value which is contained in UnacceptableInitialTerminationTime fault.

7.3.6 REALTIME PULLPOINT SUBSCRIPTION - TIMEOUT

Test Label: Event handling Realtime Pull Point Timeout

Test Case ID: EVENT-3-1-6

ONVIF Core Specification Coverage: CreatePullPointSubscription, Basic Notification Interface

Command Under Test: CreatePullPointSubscription

WSDL Reference: event.wsdl

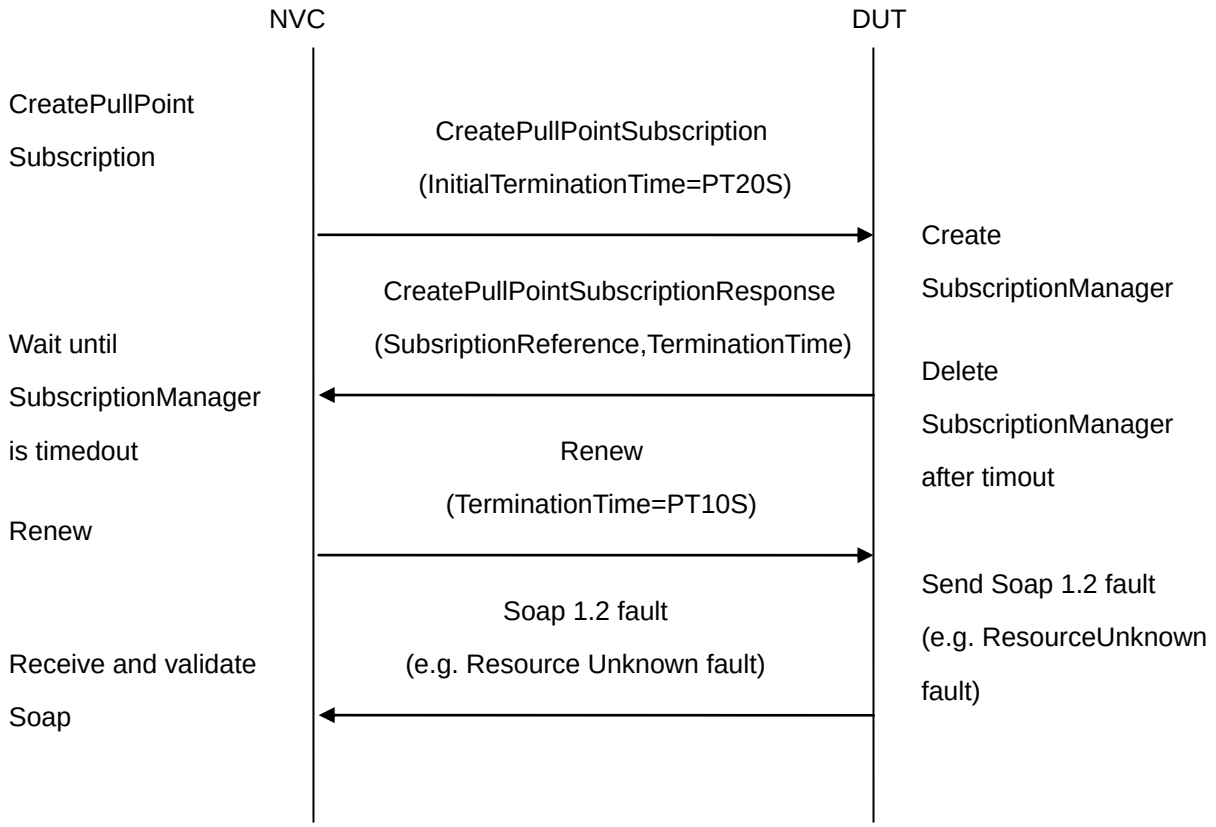
Requirement Level: MUST

Test Purpose: To verify if a SubscriptionManager times out correctly

Pre-Requisite: None

Test Configuration: NVC and DUT

Test Sequence:



Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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 20 s (SubscriptionManager timeout)
7. NVC 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 –

DUT passes 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

Note: If DUT can not accept the set value to an InitialTerminationTime, NVC retries to send the CreatePullpointSubscription request with MinimumTime value which is contained in UnacceptableInitialTerminationTime fault.

7.3.7 REALTIME PULLPOINT SUBSCRIPTION - PULLMESSAGES

Test Label: event handling REALTIME PULL POINT INTERFACE PullMessages

Test Case ID: EVENT-3-1-7

ONVIF Core Specification Coverage: CreatePullPointSubscription, SetSynchronizationPoint, PullMessages

Command Under Test: CreatePullPointSubscription, SetSynchronizationPoint, PullMessages

WSDL Reference: event.wsdl

Requirement Level: MUST

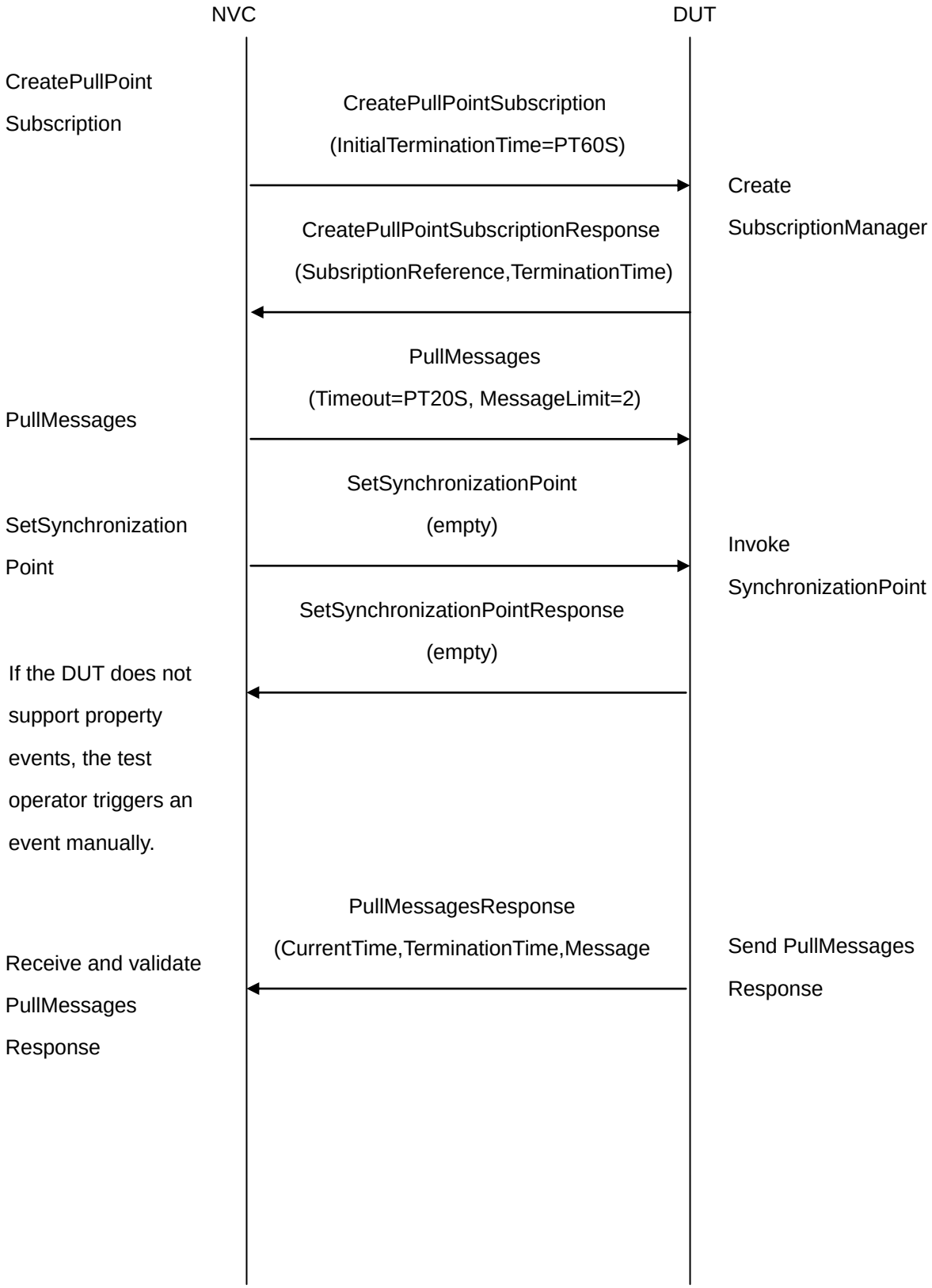
Test Purpose: To verify PullMessages command

Pre-Requisite: The DUT MUST provide at least one event.

The test operator has to ensure that the event is triggered and sent out. NVC 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: NVC and DUT

Test Sequence:


NVC
DUT
CreatePullPoint Subscription
CreatePullPointSubscription (InitialTerminationTime=PT60S)
Create SubscriptionManager
CreatePullPointSubscriptionResponse (SubsriptionReference,TerminationTime)
PullMessages (Timeout=PT20S, MessageLimit=2)
PullMessages
SetSynchronizationPoint (empty)
SetSynchronization Point
Invoke SynchronizationPoint
SetSynchronizationPointResponse (empty)
If the DUT does not support property events, the test operator triggers an event manually.
PullMessagesResponse (CurrentTime,TerminationTime,Message
Send PullMessages Response
Receive and validate PullMessages Response

Test Procedure:

1. Start an NVC.
2. Start the DUT.
3. NVC 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. NVC will invoke PullMessages command with a PullMessagesTimeout of 20s and a MessageLimit of 2
6. NVC 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 –

DUT passes 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.

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, NVC 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 can not accept the set value to Timeout or MessageLimit, NVC retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

See Annex A.9 on how to compose Subscribe when the client is interested in receiving all event supported by the DUT.

7.3.8 REALTIME PULLPOINT SUBSCRIPTION - PULLMESSAGES FILTER

Test Label: event handling REALTIME PULL POINT INTERFACE PullMessages Filter

Test Case ID: EVENT-3-1-8

ONVIF Core Specification Coverage: CreatePullPointSubscription, SetSynchronizationPoint, PullMessages, MessageFilter

Command Under Test: GetEventProperties, CreatePullPointSubscription, SetSynchronizationPoint, PullMessages

WSDL Reference: event.wsdl

Requirement Level: MUST

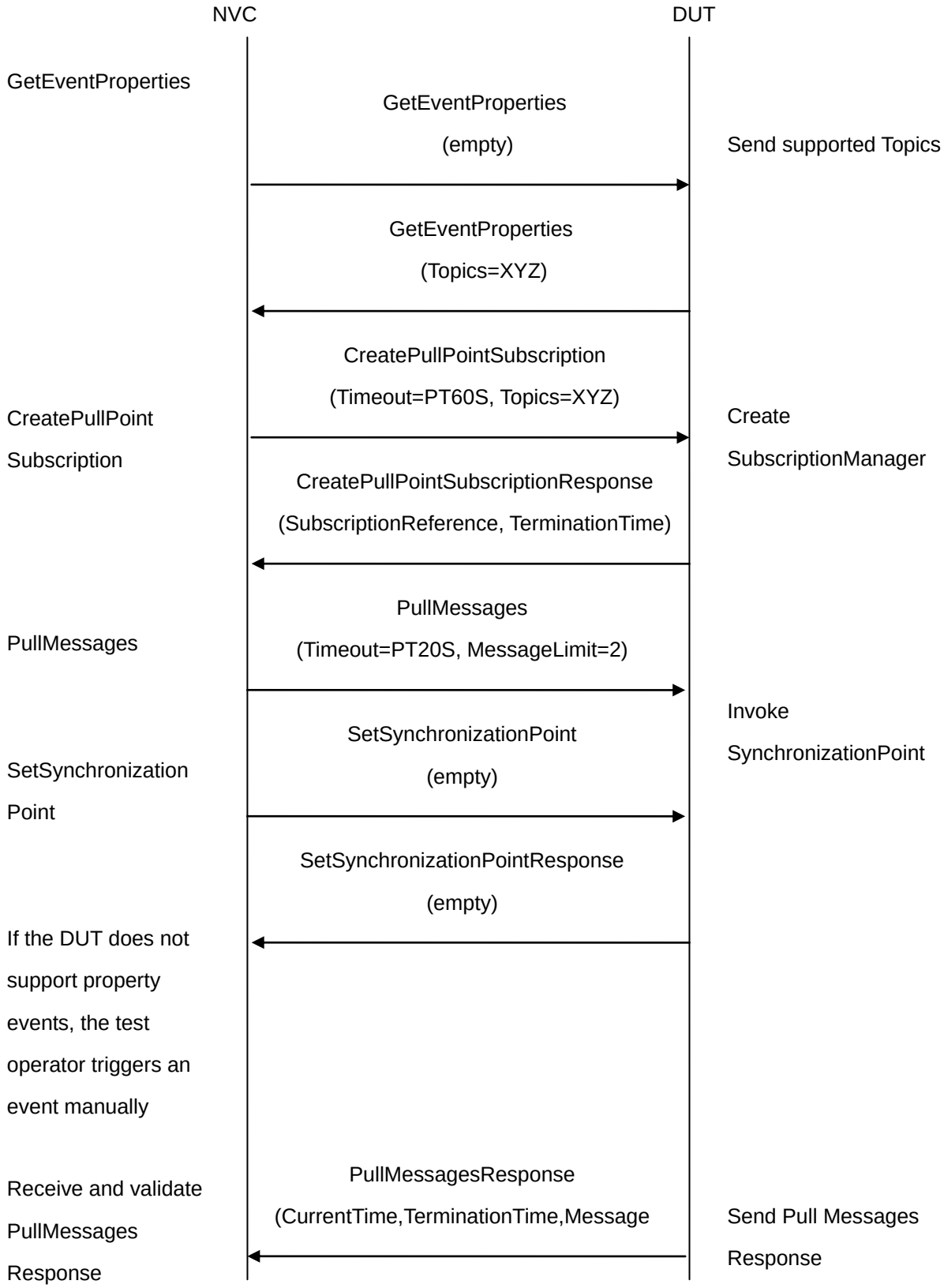
Test Purpose: To verify PullMessages command

Pre-Requisite: The DUT MUST provide at least one event.

The test operator has to ensure that the event is triggered and sent out. NVC 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: NVC and DUT

Test Sequence:


NVC
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 NVC.
2. Start the DUT.
3. NVC invokes GetEventProperties command to retrieve the supported Topics.
4. Verify that the DUT sends a GetEventPropertiesResponse; select one Topic
5. NVC 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. NVC will invoke PullMessages command with a PullMessagesTimeout of 20s and a MessageLimit of 2
9. NVC will invoke SetSynchronizationPoint command to trigger an 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 –

DUT passes 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

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, NVC 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 can not accept the set value to Timeout or MessageLimit, NVC retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

8 Security Test Cases

8.1 USER TOKEN PROFILE

Test Label: Security – Usern token profile

Test Case ID: SECURITY-1-1-1

ONVIF Core Specification Coverage: Message level security

Command Under Test: None

WSDL Reference: None

Requirement Level: MUST

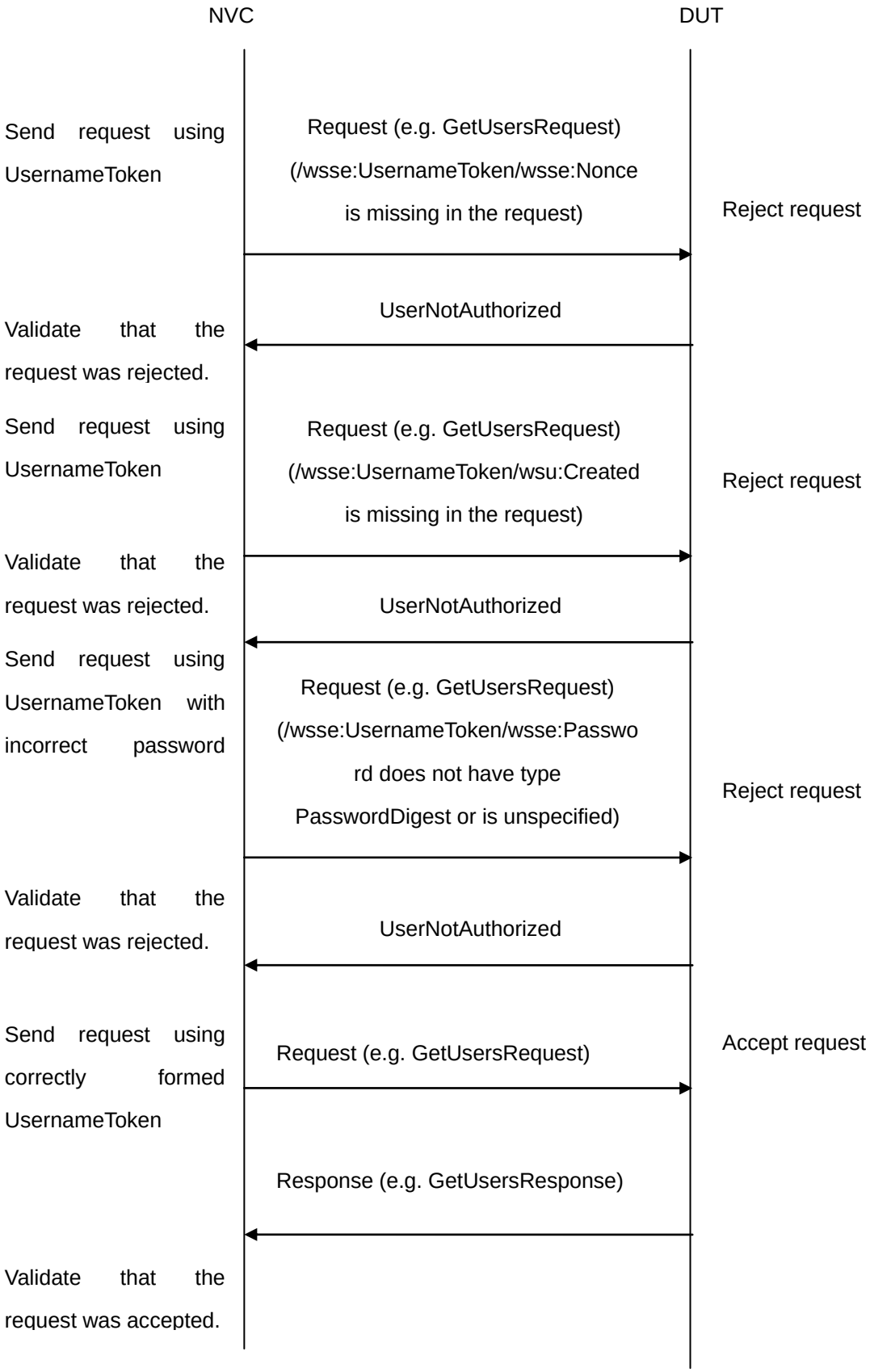
Test Purpose: To verify that the DUT supports the User Token Profile for Message level security.

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 must be created before the test is executed. Similarly, if there exists such a user but the password has not been set, then the password must 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 must choose another operation that does require authentication.

Test Configuration: DUT and NVC

Test Sequence:


NVC
DUT
Send request using UsernameToken
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
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
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. NVC sends a request that requires authentication (e.g. GetUsers) to the DUT with incorrectly implemented UsernameToken. Each of the following must be tested:
    a. Missing nonce.
    b. Missing timestamp.
    c. Incorrect password type.
2. Verify that the DUT rejects all incorrect requests.
3. NVC 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 –

DUT passes 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 Core] 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>
```

9 Annex

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: dn:NetworkVideoTransmitter

- <d:Types> list must include “NetworkVideoTransmitter” , otherwise it is considered as invalid device type.

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 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 invalid timezone.

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 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 must be valid according to the defined network mask and gateway (the gateway must be reachable and must 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

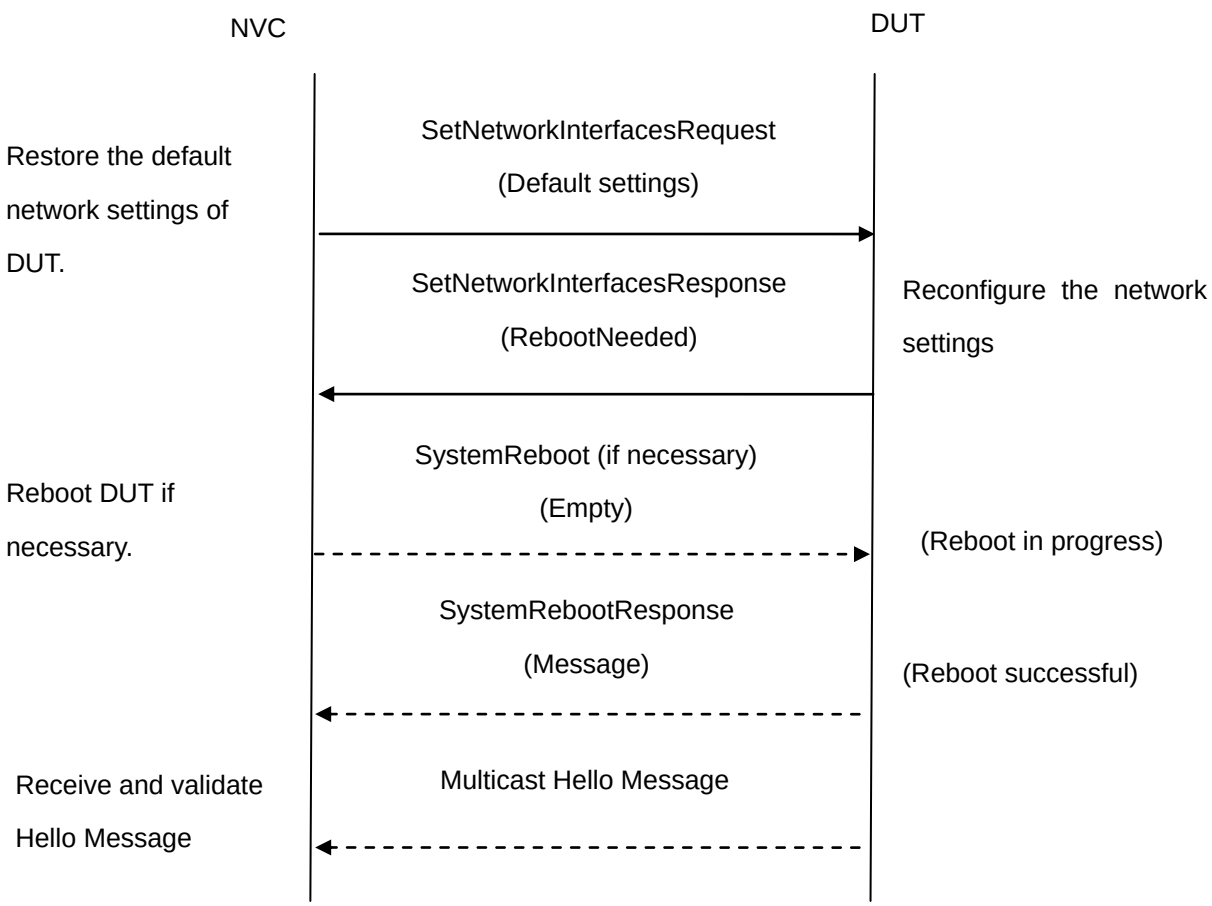
NVC uses the following timeout value in the respective Test result steps.

APP_MAX_DELAY 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, NVC 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 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 a client might be interested in receiving all or some of the Events produced by the DUT.

If a client is interested in receiving some events it includes a filter tag in the CreatePullPointSubscription or Subscribe requests describing the events which the client is interested in (see examples in the Core Spec).

If a 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 a 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, NVC has to send request with specific namespaces definition. Examples of how this request must look like are followed.

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#Pass
wordDigest">5zjIbmvWxVevGlpqg6Qnt9h8Fmo=</Password>
        <Nonce
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">ikcoi
K+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">servi
ce</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#Pass
wordDigest">znYLkZuoGqC5RrFD4KDs529JvHI=</Password>
        <Nonce
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">7L0d
q2/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">servi
ce</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#Pass
wordDigest">tg6EnHtyMWW8eUfntHO6XpPjOsg=</Password>

        <Nonce
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">iBIGp
SuHtNPbdSWGzG48ng==</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:mustUnderstand="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/NotificationPro
ducer/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/anonymous<
/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#Pass
wordDigest">5zjIbmvWxVevGlpqg6Qnt9h8Fmo=</Password>

       <Nonce
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">ikcoi
K+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:mustUnderstand="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_server</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-a932-2dd0fe
45cc87</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/2005/0
4/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</Types>

      <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#Pass
wordDigest">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#Pass
wordDigest">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:mustUnderstand="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/NotificationPro
ducer/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#Pass
wordDigest">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:mustUnderstand="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_server</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-a932-2dd0fe
45cc87</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/2005/0
4/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</Types>

      <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 not 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#Pass
wordDigest">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#Pass
wordDigest">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#Pass
wordDigest">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:mustUnderstand="1">http://docs.oasis-open.org/wsn/bw-2/NotificationProducer/SubscribeRe
quest</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#Pass
wordDigest">tg6EnHtyMWW8eUfntHO6XpPjOsg=</prefix4:Password>
        <prefix4:Nonce>iBIGpSuHtNPbdSWGzG48ng==</prefix4:Nonce>
```

```
      <prefix3:Created>2011-04-25T10:54:34Z</prefix3:Created>
    </prefix4:UsernameToken>
  </prefix4:Security>
  <prefix2:To prefix1:mustUnderstand="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_server</prefix2:Ad
dress>
   </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:Action>
 </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">us
er</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#Pass
wordDigest">5zjIbmvWxVevGlpqg6Qnt9h8Fmo=</p6:Password>
        <p7:Nonce
xmlns:p7="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">ik
coiK+AmJvA5UpfxTzG8Q==</p7:Nonce>
        <p8:Created
xmlns:p8="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">20
11-04-25T09:27:48Z</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#Pass
wordDigest">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">20
11-04-25T09:27:49Z</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">se
rvice</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#Pass
wordDigest">tg6EnHtyMWW8eUfntHO6XpPjOsg=</p7:Password>

      <p8:Nonce
xmlns:p8="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">iB
IGpSuHtNPbdSWGzG48ng==</p8:Nonce>

      <p9:Created
xmlns:p9="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">20
11-04-25T10:54:34Z</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:mustUnderstand="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/Notification
Producer/SubscribeRequest</e3:Action>

  <e4:MessageID
xmlns:e4="http://www.w3.org/2005/08/addressing">urn:uuid:9f2a12de-3a76-461b-a421-e472517bcc
7e</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/addressing/anonymo
us</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">us
er</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#Pass
wordDigest">5zjIbmvWxVevGlpqg6Qnt9h8Fmo=</e10:Password>

      <e11:Nonce
xmlns:e11="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">i
kcoiK+AmJvA5UpfxTzG8Q==</e11:Nonce>

      <e12:Created
xmlns:e12="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">2
011-04-25T09:27:48Z</e12:Created>

    </e8:UsernameToken>

  </e7:Security>

  <e13:To t:mustUnderstand="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/services</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:A
ddress>

    </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-a932-2dd
0fe45cc87</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/200
5/04/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:Types>

      <p9:Scopes
xmlns:p9="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</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#Pass
wordDigest">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">20
11-04-25T09:27:48Z</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#Pass
wordDigest">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">20
11-04-25T09:27:49Z</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#Pass
wordDigest">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">20
11-04-25T10:54:34Z</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:mustUnderstand="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/Notification
Producer/SubscribeRequest</p1:Action>

    <p1:MessageID
xmlns:p1="http://www.w3.org/2005/08/addressing">urn:uuid:9f2a12de-3a76-461b-a421-e472517bcc
7e</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#Pass
wordDigest">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">20
11-04-25T09:27:48Z</p1:Created>

    </p1:UsernameToken>

  </p1:Security>

  <p1:To t:mustUnderstand="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/services</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_server</p1:Add
ress>

    </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-a932-2dd
0fe45cc87</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/200
5/04/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:Types>

      <p1:Scopes
xmlns:p1="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</p1:Scopes>

    </p1:Probe>

  </p1:Body>

</p1:Envelope>
```