

# ONVIF<sup>™</sup>

## Base Device Test Specification

Version 17.06

June 2017

© 2017 ONVIF, Inc. All rights reserved.

Recipients of this document may copy, distribute, publish, or display this document so long as this copyright notice, license and disclaimer are retained with all copies of the document. No license is granted to modify this document.

THIS DOCUMENT IS PROVIDED "AS IS," AND THE CORPORATION AND ITS MEMBERS AND THEIR AFFILIATES, MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; OR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

IN NO EVENT WILL THE CORPORATION OR ITS MEMBERS OR THEIR AFFILIATES BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL DAMAGES, ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THIS DOCUMENT, WHETHER OR NOT (1) THE CORPORATION, MEMBERS OR THEIR AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR (2) SUCH DAMAGES WERE REASONABLY FORESEEABLE, AND ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THIS DOCUMENT. THE FOREGOING DISCLAIMER AND LIMITATION ON LIABILITY DO NOT APPLY TO, INVALIDATE, OR LIMIT REPRESENTATIONS AND WARRANTIES MADE BY THE MEMBERS AND THEIR RESPECTIVE AFFILIATES TO THE CORPORATION AND OTHER MEMBERS IN CERTAIN WRITTEN POLICIES OF THE CORPORATION.

## REVISION HISTORY

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

		<p>New IP Filtering test cases were added:</p> <p>GET IP ADDRESS FILTER</p> <p>SET IP ADDRESS FILTER – IPv4</p> <p>ADD IP ADDRESS FILTER – IPv4</p> <p>REMOVE IP ADDRESS FILTER – IPv4</p> <p>New capabilities test case was added:</p> <p>GET SERVICES – ADVANCED SECURITY SERVICE</p> <p>New event test case was added:</p> <p>REALTIME PULLPOINT SUBSCRIPTION – PULLMESSAGES AS KEEP-ALIVE</p> <p>The following test cases were updated with Id change</p> <p>IPV4 LINK LOCAL ADDRESS</p> <p>GET NETWORK PROTOCOLS CONFIGURATION</p> <p>SET NETWORK PROTOCOLS CONFIGURATION</p> <p>SET NETWORK PROTOCOLS CONFIGURATION - UNSUPPORTED PROTOCOLS</p> <p>SYSTEM COMMAND SETSYSTEMDATEANDTIME</p> <p>SYSTEM COMMAND SETSYSTEMDATEANDTIME USING NTP</p> <p>BASIC NOTIFICATION INTERFACE - INVALID TOPIC EXPRESSION</p> <p>REALTIME PULLPOINT SUBSCRIPTION - INVALID TOPIC EXPRESSION</p> <p>APP_MAX_DELAY was renamed to DISCOVERY_TIMEOUT.</p> <p>Description of DISCOVERY_TIMEOUT was updated.</p> <p>Description in Annex A.7 was updated.</p>
14.06	Jun, 2014	<p>The following test cases were updated with Id change:</p> <p>DEVICE SCOPES CONFIGURATION</p> <p>REALTIME PULLPOINT SUBSCRIPTION – PULLMESSAGES AS KEEP-ALIVE</p> <p>SEEK EVENTS</p> <p>SEEK EVENTS – REVERSE</p>
14.12	Dec, 2014	<p>The following test cases were updated with Id change:</p> <p>SECURITY COMMAND CREATEUSERS</p> <p>REALTIME PULLPOINT SUBSCRIPTION – PULLMESSAGES AS KEEP-ALIVE</p> <p>The following new test cases were added:</p>



		<p>GET SERVICES – ACCESS RULES SERVICE</p> <p>GET SERVICES – CREDENTIALS SERVICE</p> <p>GET SYSTEM URIS</p> <p>START SYSTEM RESTORE</p> <p>START SYSTEM RESTORE – INVALID BACKUP FILE</p> <p>GET REMOTE USER</p> <p>SET REMOTE USER</p> <p>The following new annexes were added:</p> <p>Annex A.20 TooManyUsers fault check</p> <p>Annex A.21 Get service capabilities</p>
15.06	Jun, 2015	<p>The following test cases were added:</p> <p>GET SERVICES – SCHEDULE SERVICE</p> <p>The following test cases were updated:</p> <p>IPV4 LINK LOCAL ADDRESS</p> <p>The following new annexes were updated:</p> <p>A.17 Action URI's for Event Service Messages</p>
16.01	Dec, 2015	<p>The following test cases were added:</p> <p>REALTIME PULLPOINT SUBSCRIPTION - SET SYNCHRONIZATION POINT</p> <p>BASIC NOTIFICATION INTERFACE - SET SYNCHRONIZATION POINT</p> <p>The tests: DEVICE-3-1-14, DEVICE-3-1-8, DEVICE-3-1-15 and IPCONFIG-1-1-5 have been updated.</p>
16.06	Mar 08, 2016	The tests EVENT-3-1-26, EVENT-3-1-27 have been added
16.06	Mar 15, 2016	The tests EVENT-3-1-28 - EVENT-3-1-31 have been added
16.06	May 15, 2016	Auxiliary operation section and test case added
16.07	Jul 08, 2016	Used GetServiceCapabilities to find out which auxiliary commands are available (not GetCapabilities)
16.12	Dec 08, 2016	REALTIME PULLPOINT SUBSCRIPTION – DIGITAL INPUT EVENT was moved from ONVIF Base Test specification to ONVIF Device IO Test specification.
17.01	Jan 09, 2017	Monitoring Events Item was added (moved from ONVIF Profile Q Test Specification)
17.02	Feb 06, 2017	<p>The following test cases have been updated:</p> <p>EVENT-2-1-24 BASIC NOTIFICATION INTERFACE - SET SYNCHRONIZATION POINT</p> <p>EVENT-3-1-25 REALTIME PULLPOINT SUBSCRIPTION – SET SYNCHRONIZATION POINT</p>

17.02	Feb 09, 2017	The following test case was added according to #1230: REALTIME PULLPOINT SUBSCRIPTION – PULLMESSAGES TIMEOUT
17.03	Feb 20, 2017	The following test case were added according to #1274: BASIC NOTIFICATION INTERFACE – CONJUNCTION IN NOTIFY FILTER (OR OPERATION) REALTIME PULLPOINT SUBSCRIPTION – CONJUNCTION IN PULLMESSAGES FILTER (OR OPERATION)
17.06	Mar 02, 2017	Last Clock Synchronization change event (SetSystemDateAndTime) was updated according to #1272
17.06	Mar 06, 2017	The following test cases were removed (moved into ONVIF Imaging test spec) according to #1352: REALTIME PULLPOINT SUBSCRIPTION – IMAGE TOO BLURRY REALTIME PULLPOINT SUBSCRIPTION – IMAGE TOO DARK REALTIME PULLPOINT SUBSCRIPTION – IMAGE TOO BRIGHT REALTIME PULLPOINT SUBSCRIPTION – GLOBAL SCENE CHANGE
17.06	Mar 15, 2017	REALTIME PULLPOINT SUBSCRIPTION – RELAY EVENT was removed according to #1364
17.06	Apr 07, 2017	GET DYNAMIC DNS CONFIGURATION added according to #1341
17.06	Apr 25, 2017	SET DYNAMIC DNS CONFIGURATION added according to #1341
17.06	May 23, 2017	The following test cases were updated according #1412: DEVICE-3-1-4 SYSTEM COMMAND SETSYSTEMDATEANDTIME TEST FOR INVALID TIMEZONE DEVICE-3-1-5 SYSTEM COMMAND SETSYSTEMDATEANDTIME TEST FOR INVALID DATE The following test case was added according #1374: DEVICE-1-1-30 GET SERVICES AND GET CAPABILITIES CONSISTENCY
17.06	Jun 21, 2017	Events Test Cases section and related parts was moved to separate document Event Handling Test Specification. Current document name was changed from Base Test Specification to Base Device Test Specification. The document formating were updated.

**Table of Contents**

<b>1</b>	<b>Introduction</b>	<b>15</b>
1.1	Scope	15
1.1.1	IP Configuration	16
1.1.2	Device Discovery	16
1.1.3	Device Management	17
1.1.4	Security	17
<b>2</b>	<b>Normative references</b>	<b>18</b>
<b>3</b>	<b>Terms and Definitions</b>	<b>20</b>
3.1	Conventions	20
3.2	Definitions	20
3.3	Abbreviations	20
<b>4</b>	<b>Test Overview</b>	<b>22</b>
4.1	Test Setup	22
4.1.1	Network Configuration for DUT	22
4.2	Prerequisites	23
4.3	Test Policy	23
4.3.1	IP Configuration	23
4.3.2	Device Discovery	24
4.3.3	Device Management	24
4.3.4	Security	26
4.3.5	Authentication method selection as a testing framework	27
<b>5</b>	<b>IP Configuration Test Cases</b>	<b>28</b>
5.1	IPv4	28
5.1.1	IPV4 STATIC IP	28
5.1.2	IPV4 DHCP	30
5.1.3	IPV4 LINK LOCAL ADDRESS	34
5.2	IPv6	35
5.2.1	IPV6 STATIC IP	35
5.2.2	IPV6 STATELESS IP CONFIGURATION - ROUTER ADVERTISEMENT	38
5.2.3	IPV6 STATELESS IP CONFIGURATION - NEIGHBOUR DISCOVERY	41

5.2.4	IPV6 STATEFUL IP CONFIGURATION .....	43
<b>6</b>	<b>Device Discovery Test Cases .....</b>	<b>46</b>
6.1	HELLO MESSAGE .....	46
6.2	HELLO MESSAGE VALIDATION .....	47
6.3	SEARCH BASED ON DEVICE SCOPE TYPES .....	48
6.4	SEARCH WITH OMITTED DEVICE AND SCOPE TYPES .....	49
6.5	RESPONSE TO INVALID SEARCH REQUEST .....	50
6.6	SEARCH USING UNICAST PROBE MESSAGE .....	51
6.7	BYE MESSAGE .....	51
6.8	DISCOVERY MODE CONFIGURATION .....	52
6.9	SOAP FAULT MESSAGE .....	54
6.10	DEVICE SCOPES CONFIGURATION .....	55
6.11	Namespace Handling .....	57
6.11.1	DISCOVERY - NAMESPACES (DEFAULT NAMESPACES FOR EACH TAG) .....	57
6.11.2	DISCOVERY - NAMESPACES (DEFAULT NAMESPACES FOR PARENT TAG) .....	58
6.11.3	DISCOVERY - NAMESPACES (NOT STANDARD PREFIXES) .....	60
6.11.4	DISCOVERY - NAMESPACES (DIFFERENT PREFIXES FOR THE SAME NAMESPACE) .....	61
6.11.5	DISCOVERY - NAMESPACES (THE SAME PREFIX FOR DIFFERENT NAMESPACES) .....	63
<b>7</b>	<b>Device Management Test Cases .....</b>	<b>65</b>
7.1	Capabilities .....	65
7.1.1	GET WSDL URL .....	65
7.1.2	ALL CAPABILITIES .....	65
7.1.3	DEVICE CAPABILITIES .....	67
7.1.4	MEDIA CAPABILITIES .....	68
7.1.5	EVENT CAPABILITIES .....	70
7.1.6	PTZ CAPABILITIES .....	71
7.1.7	SOAP FAULT MESSAGE .....	72





7.1.8	IMAGING CAPABILITIES .....	73
7.1.9	ANALYTICS CAPABILITIES .....	74
7.1.10	GET SERVICES – DEVICE SERVICE .....	75
7.1.11	GET SERVICES – MEDIA SERVICE .....	77
7.1.12	GET SERVICES – PTZ SERVICE .....	78
7.1.13	GET SERVICES – EVENT SERVICE .....	79
7.1.14	GET SERVICES – IMAGING SERVICE .....	81
7.1.15	DEVICE SERVICE CAPABILITIES .....	82
7.1.16	GET SERVICES AND GET DEVICE SERVICE CAPABILITIES CONSISTENCY .....	83
7.1.17	GET SERVICES – REPLAY SERVICE .....	85
7.1.18	GET SERVICES – RECORDING SEARCH SERVICE .....	87
7.1.19	GET SERVICES – RECORDING CONTROL SERVICE .....	88
7.1.20	GET SERVICES – RECEIVER SERVICE .....	90
7.1.21	GET SERVICES – ACCESS CONTROL SERVICE .....	91
7.1.22	GET SERVICES – DOOR CONTROL SERVICE .....	93
7.1.23	GET SERVICES – ADVANCED SECURITY SERVICE .....	94
7.1.24	GET SERVICES – ACCESS RULES SERVICE .....	95
7.1.25	GET SERVICES – CREDENTIAL SERVICE .....	97
7.1.26	GET SERVICES – SCHEDULE SERVICE .....	98
7.1.27	GET SERVICES AND GET CAPABILITIES CONSISTENCY .....	100
7.2	Network .....	115
7.2.1	NETWORK COMMAND HOSTNAME CONFIGURATION .....	115
7.2.2	NETWORK COMMAND SETHOSTNAME TEST ERROR CASE .....	116
7.2.3	GET DNS CONFIGURATION .....	117
7.2.4	SET DNS CONFIGURATION - SEARCHDOMAIN .....	118
7.2.5	SET DNS CONFIGURATION - DNSMANUAL IPV4 .....	119
7.2.6	SET DNS CONFIGURATION - DNSMANUAL IPV6 .....	120
7.2.7	SET DNS CONFIGURATION - FROMDHCP .....	121
7.2.8	SET DNS CONFIGURATION - DNSMANUAL INVALID IPV4 .....	122
7.2.9	SET DNS CONFIGURATION - DNSMANUAL INVALID IPV6 .....	123

7.2.10	GET NTP CONFIGURATION .....	125
7.2.11	SET NTP CONFIGURATION - NTPMANUAL IPV4 .....	125
7.2.12	SET NTP CONFIGURATION - NTPMANUAL IPV6 .....	127
7.2.13	SET NTP CONFIGURATION - FROMDHCP .....	128
7.2.14	SET NTP CONFIGURATION - NTPMANUAL INVALID IPV4 .....	129
7.2.15	SET NTP CONFIGURATION - NTPMANUAL INVALID IPV6 .....	130
7.2.16	GET NETWORK INTERFACE CONFIGURATION .....	131
7.2.17	SET NETWORK INTERFACE CONFIGURATION - IPV4 .....	132
7.2.18	SET NETWORK INTERFACE CONFIGURATION - IPV6 .....	135
7.2.19	SET NETWORK INTERFACE CONFIGURATION - INVALID IPV4 .....	137
7.2.20	SET NETWORK INTERFACE CONFIGURATION - INVALID IPV6 .....	138
7.2.21	GET NETWORK PROTOCOLS CONFIGURATION .....	140
7.2.22	SET NETWORK PROTOCOLS CONFIGURATION .....	141
7.2.23	SET NETWORK PROTOCOLS CONFIGURATION - UNSUPPORTED PROTOCOLS .....	142
7.2.24	GET NETWORK DEFAULT GATEWAY CONFIGURATION .....	143
7.2.25	SET NETWORK DEFAULT GATEWAY CONFIGURATION - INVALID IPV4 .....	144
7.2.26	SET NETWORK DEFAULT GATEWAY CONFIGURATION - INVALID IPV6 .....	146
7.2.27	SET NETWORK DEFAULT GATEWAY CONFIGURATION - IPV4 .....	147
7.2.28	SET NETWORK DEFAULT GATEWAY CONFIGURATION - IPV6 .....	148
7.2.29	NETWORK COMMAND SETHOSTNAME TEST .....	150
7.2.30	GET DYNAMIC DNS CONFIGURATION .....	151
7.2.31	SET DYNAMIC DNS CONFIGURATION .....	151
7.3	System .....	154
7.3.1	SYSTEM COMMAND GETSYSTEMDATEANDTIME .....	154
7.3.2	SYSTEM COMMAND SETSYSTEMDATEANDTIME TEST FOR INVALID TIMEZONE .....	155
7.3.3	SYSTEM COMMAND SETSYSTEMDATEANDTIME TEST FOR INVALID DATE .....	157



7.3.4	SYSTEM COMMAND FACTORY DEFAULT HARD .....	159
7.3.5	SYSTEM COMMAND FACTORY DEFAULT SOFT .....	160
7.3.6	SYSTEM COMMAND REBOOT .....	161
7.3.7	SYSTEM COMMAND DEVICE INFORMATION .....	162
7.3.8	SYSTEM COMMAND GETSYSTEMLOG .....	163
7.3.9	SYSTEM COMMAND SETSYSTEMDATEANDTIME .....	164
7.3.10	SYSTEM COMMAND SETSYSTEMDATEANDTIME USING NTP .....	165
7.3.11	GET SYSTEM URIS .....	167
7.3.12	START SYSTEM RESTORE .....	169
7.3.13	START SYSTEM RESTORE – INVALID BACKUP FILE .....	171
7.4	Security .....	172
7.4.1	SECURITY COMMAND GETUSERS .....	172
7.4.2	SECURITY COMMAND CREATEUSERS ERROR CASE .....	173
7.4.3	SECURITY COMMAND DELETEUSERS .....	175
7.4.4	SECURITY COMMAND DELETEUSERS ERROR CASE .....	177
7.4.5	SECURITY COMMAND DELETEUSERS DELETE ALL USERS .....	178
7.4.6	SECURITY COMMAND SETUSER .....	179
7.4.7	SECURITY COMMAND USER MANAGEMENT ERROR CASE .....	181
7.4.8	SECURITY COMMAND CREATEUSERS .....	183
7.4.9	GET REMOTE USER .....	185
7.4.10	SET REMOTE USER .....	185
7.5	I/O .....	187
7.5.1	IO COMMAND GETRELAYOUTPUTS .....	187
7.5.2	RELAY OUTPUTS COUNT IN GETRELAYOUTPUTS AND GETCAPABILITIES .....	188
7.5.3	IO COMMAND SETRELAYOUTPUTSETTINGS .....	189
7.5.4	IO COMMAND SETRELAYOUTPUTSTATE – BISTABLE MODE (OPENED IDLE STATE) .....	192
7.5.5	IO COMMAND SETRELAYOUTPUTSTATE – BISTABLE MODE (CLOSED IDLE STATE) .....	193

- 7.5.6 IO COMMAND SETRELAYOUTPUTSTATE – MONOSTABLE MODE  
(OPENED IDLE STATE) ..... 194
- 7.5.7 IO COMMAND SETRELAYOUTPUTSTATE – MONOSTABLE MODE  
(CLOSED IDLE STATE) ..... 195
- 7.5.8 IO COMMAND SETRELAYOUTPUTSTATE – MONOSTABLE MODE  
(INACTIVE BEFORE DELAYTIME EXPIRED) ..... 197
- 7.5.9 IO COMMAND SETRELAYOUTPUTSETTINGS – INVALID TOKEN ..... 198
- 7.5.10 IO COMMAND SETRELAYOUTPUTSTATE – INVALID TOKEN ..... 199
- 7.6 Namespace Handling ..... 200
  - 7.6.1 DEVICE MANAGEMENT - NAMESPACES (DEFAULT NAMESPACES  
FOR EACH TAG) ..... 200
  - 7.6.2 DEVICE MANAGEMENT - NAMESPACES (DEFAULT NAMESPACES  
FOR PARENT TAG) ..... 201
  - 7.6.3 DEVICE MANAGEMENT - NAMESPACES (NOT STANDARD  
PREFIXES) ..... 203
  - 7.6.4 DEVICE MANAGEMENT - NAMESPACES (DIFFERENT PREFIXES FOR  
THE SAME NAMESPACE) ..... 204
  - 7.6.5 DEVICE MANAGEMENT - NAMESPACES (THE SAME PREFIX FOR  
DIFFERENT NAMESPACES) ..... 205
- 7.7 IP Filtering ..... 206
  - 7.7.1 GET IP ADDRESS FILTER ..... 206
  - 7.7.2 SET IP ADDRESS FILTER – IPv4 ..... 208
  - 7.7.3 ADD IP ADDRESS FILTER – IPv4 ..... 209
  - 7.7.4 REMOVE IP ADDRESS FILTER – IPv4 ..... 210
- 7.8 Auxiliary Operation ..... 212
  - 7.8.1 AUXILIARY COMMANDS ..... 212
- 7.9 Monitoring Events ..... 213
  - 7.9.1 Processor Usage event ..... 213
  - 7.9.2 Last Reset event ..... 216
  - 7.9.3 Last Reboot event ..... 219
  - 7.9.4 Last Reboot event (Status change) ..... 221



7.9.5	Last Clock Synchronization event .....	225
7.9.6	Last Clock Synchronization change event (SetSystemDateAndTime) .....	228
7.9.7	Last Clock Synchronization change event (NTP message) .....	231
7.9.8	Last Backup event .....	235
7.9.9	Fan Failure event .....	238
7.9.10	Power Supply Failure event .....	241
7.9.11	Storage Failure event .....	244
7.9.12	Critical Temperature event .....	247
<b>8</b>	<b>Security Test Cases .....</b>	<b>250</b>
8.1	USER TOKEN PROFILE .....	250
8.2	DIGEST AUTHENTICATION .....	252
<b>A</b>	<b>Helper Procedures and Additional Notes .....</b>	<b>254</b>
A.1	Invalid Device Type and Scope Type .....	254
A.2	Invalid Hostname, DNSname .....	254
A.3	Invalid TimeZone .....	254
A.4	Invalid SOAP 1.2 Fault Message .....	255
A.5	Invalid WSDL URL .....	255
A.6	Valid/Invalid IPv4 Address .....	255
A.7	WS-Discovery Timeout Value .....	255
A.8	Restore Network Settings .....	255
A.9	Subscribe and CreatePullPointSubscription for Receiving All Events .....	257
A.10	Valid Expression Indicating Empty IP Address .....	257
A.11	Example of Requests for Namespaces Test Cases .....	259
A.12	Procedure to Turn On IPv4 DHCP .....	275
A.13	Procedure to Turn On IPv4 DHCP .....	277
A.14	Procedure to Turn On IPv6 DHCP .....	278
A.15	Procedure to Turn Off IPv6 DHCP .....	280
A.16	Name and Token Parameters Maximum Length .....	282
A.17	TooManyUsers Fault Check .....	282
A.18	Get Service Capabilities (Device Management) .....	283
A.19	Get Capabilities (Device Management) .....	283

- A.20 Restoring System Date and Time ..... 284
- A.21 Set NTP Settings ..... 285
- A.22 Restoring NTP Settings ..... 286



# 1 Introduction

The goal of the ONVIF test specification set is to make it possible to realize fully interoperable IP physical security implementation from different vendors. The set of ONVIF test specification describes the test cases need to verify the [ONVIF Network Interface Specs] and [ONVIF Conformance] requirements. In addition, the test cases are to be basic inputs for some Profile specification requirements. It also describes the test framework, test setup, pre-requisites, test policies needed for the execution of the described test cases.

This ONVIF Base Test Specification acts as a supplementary document to the [ONVIF Network Interface Specs], illustrating test cases that need to be executed and passed. And also this specification also acts as an input document to the development of test tool which will be used to test the ONVIF device implementation conformance towards ONVIF standard. As the test tool performs as a Client during testing, this test tool is referred as ONVIF Client hereafter.

## 1.1 Scope

This ONVIF Base Test Specification defines and regulates the conformance testing procedure for the ONVIF conformant devices. Conformance testing is meant to be functional black-box testing. The objective of this specification is to provide the test cases to test individual requirements of ONVIF devices according to ONVIF core services which are defined in [ONVIF Network Interface Specs].

The principal intended purposes are:

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

This specification does not address the following:

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

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

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

### 1.1.1 IP Configuration

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

The scope of this specification is to cover following configurations:

- IPv4 configuration
  - Static IP configuration
  - Link-local address configuration
  - DHCP configuration
- IPv6 configuration
  - Static IP configuration
  - Stateless IP configuration
  - Stateful IP configuration

### 1.1.2 Device Discovery

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

Device Discovery testing tests the following:

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



Refer to Table 1.1 for Device Discovery tests.

**Table 1.1. Device Discovery**

Feature	Messages
Device Discovery	Hello
	Probe
	Probe Match
	Bye

### 1.1.3 Device Management

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

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

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

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

- Namespace handling

### 1.1.4 Security

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

## 2 Normative references

- [ONVIF Conformance] ONVIF Conformance Process Specification:  
<https://www.onvif.org/profiles/conformance/>
- [ONVIF Profile Policy] ONVIF Profile Policy:  
<https://www.onvif.org/profiles/>
- [ONVIF Network Interface Specs] ONVIF Network Interface Specification documents:  
<https://www.onvif.org/profiles/specifications/>
- [ONVIF Core Specs] ONVIF Core Specifications:  
<https://www.onvif.org/profiles/specifications/>
- [ISO/IEC Directives, Part 2] ISO/IEC Directives, Part 2, Annex H:  
<http://www.iso.org/directives>
- [ISO 16484-5] ISO 16484-5:2014-09 Annex P:  
<https://www.iso.org/obp/ui/#iso:std:63753:en>
- [SOAP 1.2, Part 1] W3C SOAP 1.2, Part 1, Messaging Framework:  
<http://www.w3.org/TR/soap12-part1/>
- [XML-Schema, Part 1] W3C XML Schema Part 1: Structures Second Edition:  
<http://www.w3.org/TR/xmlschema-1/>
- [XML-Schema, Part 2] W3C XML Schema Part 2: Datatypes Second Edition:  
<http://www.w3.org/TR/xmlschema-2/>
- [WS-Security] "Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)", OASIS Standard, February 2006.:  
<http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>
- [RFC 3986] "Uniform Resource Identifier (URI): Generic Syntax", T. Berners-Lee et. al., January 2005.:  
<http://www.ietf.org/rfc/rfc3986>

- [RFC 1123] "Requirements for Internet Hosts - Application and Support", IETF, R. Braden (Ed), October 1989.:  
<https://www.ietf.org/rfc/rfc1123>
- [RFC 952] "DOD INTERNET HOST TABLE SPECIFICATION", K. Harrenstien et. al., October 1985.:  
<https://www.ietf.org/rfc/rfc952>
- [RFC 758] "ASSIGNED NUMBERS", J. Postel, August 1979.:  
<https://www.ietf.org/rfc/rfc758>
- [RFC 3927] "Dynamic Configuration of IPv4 Link-Local Addresses", S. Cheshire, B. Aboba and E. Guttman, May 2005.:  
<https://www.ietf.org/rfc/rfc3927>
- [RFC 2780] "IANA Allocation Guidelines For Values in the Internet", S. Bradner and V. Paxson, March 2000.:  
<https://www.ietf.org/rfc/rfc2780>

## 3 Terms and Definitions

### 3.1 Conventions

The key words "shall", "shall not", "should", "should not", "may", "need not", "can", "cannot" in this specification are to be interpreted as described in [ISO/IEC Directives Part 2].

### 3.2 Definitions

This section describes terms and definitions used in this document.

<b>Profile</b>	See ONVIF Profile Policy.
<b>ONVIF Device</b>	Computer appliance or software program that exposes one or multiple ONVIF Web Services.
<b>ONVIF Client</b>	Computer appliance or software program that uses ONVIF Web Services.
<b>SOAP</b>	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.
<b>Device Test Tool</b>	ONVIF Device Test Tool that tests ONVIF Device implementation towards the ONVIF Test Specification set.
<b>Device Management Service</b>	Services to configure common Device settings, such as network, security, system, etc.
<b>Capability</b>	The capability commands allow a client to ask for the services provided by an ONVIF device.
<b>Network</b>	A network is an interconnected group of devices communicating using the Internet protocol.
<b>Proxy Server</b>	A server that services the requests of its clients by forwarding requests to other servers. A Proxy provides indirect network connections to its clients.
<b>Switching Hub</b>	A device for connecting multiple Ethernet devices together, making them act as a single network segment.
<b>Target Service</b>	An endpoint that makes itself available for discovery.

### 3.3 Abbreviations

This section describes abbreviations used in this document.

<b>DP</b>	Discovery Proxy
<b>DNS</b>	Domain Name System

<b>DHCP</b>	Dynamic Host Configuration Protocol
<b>HTTP</b>	Hyper Text Transport Protocol.
<b>HTTPS</b>	Hyper Text Transport Protocol over Secure Socket Layer
<b>WSDL</b>	Web Services Description Language.
<b>XML</b>	eXtensible Markup Language.
<b>WS-I BP 2.0</b>	Web Services Interoperability Basic Profile version 2.0
<b>IP</b>	Internet Protocol
<b>IPv4</b>	Internet Protocol version 4
<b>IPv6</b>	Internet Protocol version 6
<b>NTP</b>	Network Time Protocol
<b>POSIX</b>	Portable Operating System Interface
<b>RTCP</b>	RTP Control Protocol
<b>RTSP</b>	Real Time Streaming Protocol
<b>RTP</b>	Real-time Transport Protocol
<b>SDP</b>	Session Description Protocol
<b>TCP</b>	Transport Control Protocol
<b>UTC</b>	Coordinated Universal Time
<b>UDP</b>	User Datagram Protocol
<b>URI</b>	Uniform Resource Identifier

## 4 Test Overview

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

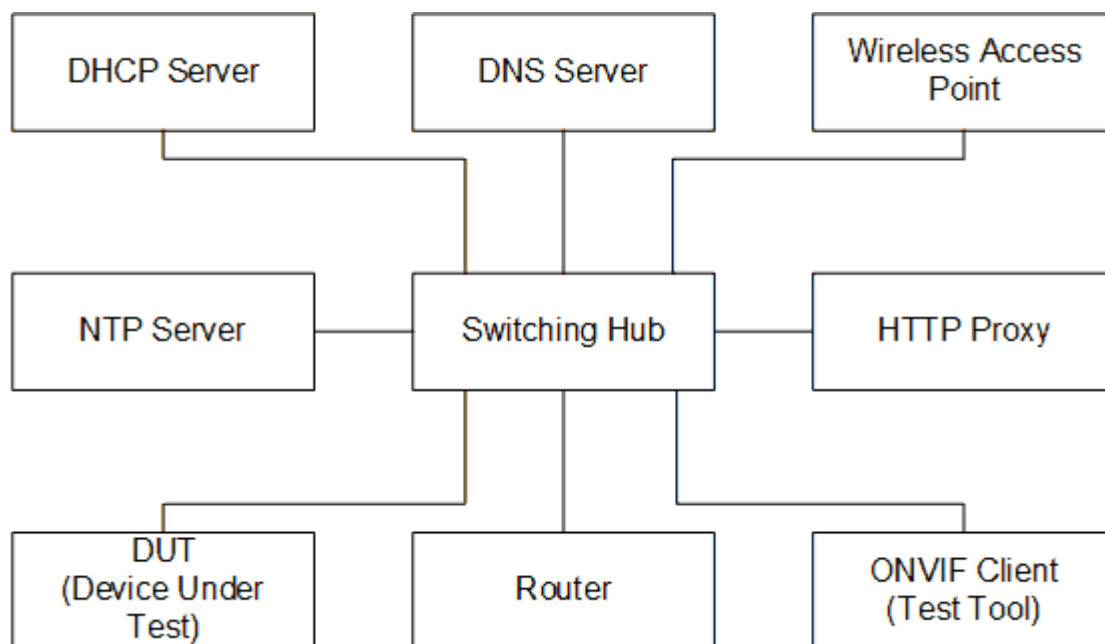
### 4.1 Test Setup

#### 4.1.1 Network Configuration for DUT

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

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

**Figure 4.1. Test Configuration for DUT**



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

**ONVIF Client (Test Tool):** Tests are executed by this system and it controls the behavior of the DUT. It handles both expected and unexpected behavior.

**HTTP Proxy:** provides facilitation in case of RTP and RTSP tunneling over HTTP.

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

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

**DHCP Server:** provides IPv4 Address to the connected devices.

**NTP Server:** provides time synchronization between ONVIF Client and DUT.

**Switching Hub:** provides network connectivity among all the test equipments in the test environment. All devices should be connected to the Switching Hub.

**Router:** provides router advertisements for IPv6 configuration.

## 4.2 Prerequisites

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

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

## 4.3 Test Policy

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

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

### 4.3.1 IP Configuration

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

The device under test shall support SetNetworkInterfaces method.

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

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

The following tests are related to IPv4:

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

The following tests are related to IPv6:

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

The device under test shall have at least one network interface that provides IPv4 connectivity. And it should have at least one network interface that provides IPv6 connectivity.

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

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

Please refer to [Section 5](#) for IP Configuration Test Cases.

## 4.3.2 Device Discovery

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

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

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

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

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

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

Please refer to [Section 6](#) for Device Discovery Test Cases.

## 4.3.3 Device Management



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

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

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

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

Refer to [Annex A.2](#) for valid host name.

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

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

DUT can support the following property Monitoring Events:

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

If DUT supports at least one Monitoring event:

- DUT shall support GetEventProperties command and return all supported events in TopicSet.
- DUT shall support Pull Point Subscription and Topic Expression filter.
- DUT shall generate property events with initial state after subscription was done.

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

Please refer to [Section 7](#) for Device Management Test Cases.

#### 4.3.4 Security

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

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

Please refer to [Section 8](#) for Security Test Cases.

### 4.3.5 Authentication method selection as a testing framework

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

Procedure:

1. ONVIF Client invokes a specific command which is under testing without any user credentials (no WS-UsernameToken, no HTTP digest authentication header).
2. If the DUT returns a correct response, then ONVIF Client determines that DUT does not require any user authentication toward the command according to the configured security policy.
3. If the DUT returns HTTP 401 Unauthorized error along with WWW-Authentication: Digest header, then ONVIF Client determines that DUT supports HTTP digest authentication. ONVIF Client shall provide with the proper level of user credential to continue the test procedure.
4. If the DUT returns SOAP fault (Sender/NotAuthorized) message, then ONVIF Client determines that WS-UsernameToken is supported by DUT. ONVIF Client shall provide with the proper level of user credential to continue the test procedure.

## 5 IP Configuration Test Cases

### 5.1 IPv4

#### 5.1.1 IPV4 STATIC IP

**Test Case ID:** IPCONFIG-1-1-1

**Specification Coverage:** IP Configuration (ONVIF Core Specification)

**Feature Under Test:** Static IPv4 Configuration

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To test IPv4 Static IP Configuration.

**Pre-Requisite:** None.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetNetworkInterfaces** request.
4. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
  - NetworkInterfaces list := *networkInterfacesList1*
5. Set *defaultNetworkSettings* := first interface from *networkInterfacesList1*.
6. ONVIF Client invokes **SetNetworkInterfaces** request with parameters
  - InterfaceToken := *defaultNetworkSettings.@token*
  - NetworkInterface.Enabled skipped
  - NetworkInterface.Link skipped
  - NetworkInterface.MTU skipped
  - NetworkInterface.IPv4.Enabled := true
  - NetworkInterface.IPv4.Manual.Address := another address in the same subnet as an original address

- NetworkInterface.IPv4.Manual.PrefixLength := current IPv4 prefix of the DUT
  - NetworkInterface.IPv4.DHCP := false
  - NetworkInterface.IPv6 skipped
  - NetworkInterface.Extension skipped
7. The DUT responds with **SetNetworkInterfacesResponse** message with parameters
- RebootNeeded =: *rebootNeeded*
8. If *rebootNeeded* = true:
- 8.1. ONVIF Client invokes **SystemReboot** request.
- 8.2. The DUT responds with **SystemRebootResponse** message with parameters
- Message
9. The DUT sends **Hello** message from a newly configured address:
- EndpointReference
  - Types =: *types*
  - Scopes
  - XAddrs =: *xAddrs*
  - MetadataVersion
10. If *types* is skipped or empty, FAIL the test and go to step 20.
11. If *xAddrs* is skipped or empty, FAIL the test and go to step 20.
12. ONVIF Client invokes **GetNetworkInterfaces** request.
13. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
- NetworkInterfaces list =: *networkInterfacesList2*
14. Set *updatedNetworkSettings* := item from *networkInterfacesList2* list with @token = *defaultNetworkSettings.@token*.
15. If *updatedNetworkSettings.IPv4* is skipped or empty, FAIL the test and go to step 20.
16. If *updatedNetworkSettings.IPv4.Enabled* = false, FAIL the test and go to step 20.

17. If *updatedNetworkSettings.IPv4.Config.Manual* is skipped, FAIL the test and go to step 20.
18. If *updatedNetworkSettings.IPv4.Config.Manual* is not equal to address at step 6, FAIL the test and go to step 20.
19. If *updatedNetworkSettings.IPv4.Config.DHCP* = true, FAIL the test and go to step 20.
20. ONVIF Client restores the original settings by following the procedure mentioned in [Annex A.8](#) with the following input and output parameters
  - in *defaultNetworkSettings* - Original default network settings

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetNetworkInterfacesResponse** message.
- DUT did not send **SetNetworkInterfacesResponse** message.
- DUT did not send **SystemRebootResponse** message.
- DUT did not send **Hello** message during *rebootTimeout*.

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

## 5.1.2 IPV4 DHCP

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

**Specification Coverage:** IP Configuration (ONVIF Core Specification)

**Feature Under Test:** DHCP IPv4 Configuration

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To test IPv4 DHCP Configuration.

**Pre-Requisite:** None.

**Test Configuration:** ONVIF Client, DUT, and DHCPv4 server

**Test Procedure:**

1. Start DHCPv4 server.

2. Start an ONVIF Client.
3. Start the DUT.
4. ONVIF Client invokes **GetNetworkInterfaces** request.
5. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
  - NetworkInterfaces list =: *networkInterfacesList1*
6. Set *defaultNetworkSettings* := first interface from *networkInterfacesList1*.
7. If *defaultNetworkSettings*.IPv4.DHCP = true:
  - 7.1. ONVIF Client invokes **SetNetworkInterfaces** request with parameters
    - InterfaceToken := *defaultNetworkSettings*.@token
    - NetworkInterface.Enabled skipped
    - NetworkInterface.Link skipped
    - NetworkInterface.MTU skipped
    - NetworkInterface.IPv4.Enabled := true
    - NetworkInterface.IPv4.Manual.Address := another address in the same subnet as an original address
    - NetworkInterface.IPv4.Manual.PrefixLength := current IPv4 prefix of the DUT
    - NetworkInterface.IPv4.DHCP := false
    - NetworkInterface.IPv6.Enabled = false
    - NetworkInterface.IPv6.AcceptRouterAdvert skipped
    - NetworkInterface.IPv6.Manual skipped
    - NetworkInterface.IPv6.DHCP skipped
    - NetworkInterface.Extension skipped
  - 7.2. The DUT responds with **SetNetworkInterfacesResponse** message with parameters
    - RebootNeeded =: *rebootNeeded*
  - 7.3. If *rebootNeeded* = true:

7.3.1. ONVIF Client invokes **SystemReboot** request.

7.3.2. The DUT responds with **SystemRebootResponse** message with parameters

- Message

7.4. The DUT sends **Hello** message from a newly configured address:

- EndpointReference
- Types =: *types*
- Scopes
- XAddrs =: *xAddrs*
- MetadataVersion

8. ONVIF Client invokes **SetNetworkInterfaces** request with parameters

- InterfaceToken := *defaultNetworkSettings.@token*
- NetworkInterface.Enabled skipped
- NetworkInterface.Link skipped
- NetworkInterface.MTU skipped
- NetworkInterface.IPv4.Enabled := true
- NetworkInterface.IPv4.Manual skipped
- NetworkInterface.IPv4.DHCP := true
- NetworkInterface.IPv6.Enabled = false
- NetworkInterface.IPv6.AcceptRouterAdvert skipped
- NetworkInterface.IPv6.Manual skipped
- NetworkInterface.IPv6.DHCP skipped
- NetworkInterface.Extension skipped

9. The DUT responds with **SetNetworkInterfacesResponse** message with parameters

- RebootNeeded =: *rebootNeeded*



10. If *rebootNeeded* = true:
  - 10.1. ONVIF Client invokes **SystemReboot** request.
  - 10.2. The DUT responds with **SystemRebootResponse** message with parameters
    - Message
11. The DUT sends **Hello** message from a newly configured address:
  - EndpointReference
  - Types =: *types*
  - Scopes
  - XAddrs =: *xAddrs*
  - MetadataVersion
12. If *types* is skipped or empty, FAIL the test and go to step 21.
13. If *xAddrs* is skipped or empty, FAIL the test and go to step 21.
14. ONVIF Client invokes **GetNetworkInterfaces** request.
15. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
  - NetworkInterfaces list =: *networkInterfacesList2*
16. Set *updatedNetworkSettings* := item from *networkInterfacesList2* list with @token = *defaultNetworkSettings.@token*.
17. If *updatedNetworkSettings.IPv4* is skipped or empty, FAIL the test and go to step 21.
18. If *updatedNetworkSettings.IPv4.Enabled* = false, FAIL the test and go to step 21.
19. If *updatedNetworkSettings.IPv4.Config.FromDHCP* is skipped, FAIL the test and go to step 21.
20. If *updatedNetworkSettings.IPv4.Config.DHCP* = false, FAIL the test and go to step 21.
21. ONVIF Client restores the original settings by following the procedure mentioned in [Annex A.8](#) with the following input and output parameters
  - in *defaultNetworkSettings* - Original default network settings

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetNetworkInterfacesResponse** message.
- DUT did not send **SetNetworkInterfacesResponse** message.
- DUT did not send **SystemRebootResponse** message.
- DUT did not send **Hello** message during *rebootTimeout*.

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

### 5.1.3 IPV4 LINK LOCAL ADDRESS

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

**Specification Coverage:** IP Configuration (ONVIF Core Specification), Set zero configuration (ONVIF Core Specification)

**Feature Under Test:** SetZeroConfiguration, GetZeroConfiguration, Link-Local IPv4 Configuration

**WSDL Reference:** devicemgmt.wsdl

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

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetNetworkInterfaces** request.
4. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
  - NetworkInterfaces list =: *networkInterfacesList1*
5. Set *defaultNetworkSettings* := first interface from *networkInterfacesList1*.
6. ONVIF Client invokes **GetZeroConfiguration** request.
7. The DUT responds with **GetZeroConfigurationResponse** message with parameters

- ZeroConfiguration =: *initialZeroConfiguration*
8. ONVIF Client invokes **SetZeroConfiguration** request with parameters
    - InterfaceToken := *defaultNetworkSettings.@token*
    - Enabled := true
  9. The DUT responds with **SetZeroConfigurationResponse** message.
  10. ONVIF Client waits until *timeout1* timeout expires.
  11. ONVIF Client invokes **GetZeroConfiguration** request.
  12. The DUT responds with **GetZeroConfigurationResponse** message with parameters
    - ZeroConfiguration =: *updatedZeroConfiguration*
  13. If *updatedZeroConfiguration.Enabled* = failed, FAIL the test and go to step 16.
  14. If *updatedZeroConfiguration.InterfaceToken* != *defaultNetworkSettings.@token*, FAIL the test and go to step 16.
  15. If *updatedZeroConfiguration.Addresses* is skipped, FAIL the test and go to step 16.
  16. ONVIF Client restores the original zero configuration settings.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetNetworkInterfacesResponse** message.
- DUT did not send **GetZeroConfigurationResponse** message.
- DUT did not send **SetZeroConfigurationResponse** message.

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

## 5.2 IPv6

### 5.2.1 IPV6 STATIC IP

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

**Specification Coverage:** IP Configuration (ONVIF Core Specification)

**Feature Under Test:** Static IPv6 Configuration

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To test IPv6 Static IP Configuration.

**Pre-Requisite:** IPv6 is supported by DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetNetworkInterfaces** request.
4. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
  - NetworkInterfaces list := *networkInterfacesList1*
5. Set *defaultNetworkSettings* := interface from *networkInterfacesList1* list with specified IPv6.
6. ONVIF Client invokes **SetNetworkInterfaces** request with parameters
  - InterfaceToken := *defaultNetworkSettings.@token*
  - NetworkInterface.Enabled := true
  - NetworkInterface.Link skipped
  - NetworkInterface.MTU skipped
  - NetworkInterface.IPv4 skipped
  - NetworkInterface.IPv6.Enabled := true
  - NetworkInterface.IPv6.AcceptRouterAdvert := *defaultNetworkSettings.IPv6.Config.AcceptRouterAdvert*
  - NetworkInterface.IPv6.Manual.Address := "2001:1:1:1:1:1:1:1"
  - NetworkInterface.IPv6.Manual.PrefixLength := 64
  - NetworkInterface.IPv6.DHCP := Off

- NetworkInterface.Extension skipped
7. The DUT responds with **SetNetworkInterfacesResponse** message with parameters
    - RebootNeeded =: *rebootNeeded*
  8. If *rebootNeeded* = true:
    - 8.1. ONVIF Client invokes **SystemReboot** request.
    - 8.2. The DUT responds with **SystemRebootResponse** message with parameters
      - Message
  9. The DUT sends **Hello** message from a newly configured address:
    - EndpointReference
    - Types =: *types*
    - Scopes
    - XAddrs =: *xAddrs*
    - MetadataVersion
  10. If *types* is skipped or empty, FAIL the test and go to step 20.
  11. If *xAddrs* is skipped or empty, FAIL the test and go to step 20.
  12. ONVIF Client invokes **GetNetworkInterfaces** request.
  13. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
    - NetworkInterfaces list =: *networkInterfacesList2*
  14. Set *updatedNetworkSettings* := item from *networkInterfacesList2* list with @token = *defaultNetworkSettings.@token*.
  15. If *updatedNetworkSettings.IPv6* is skipped or empty, FAIL the test and go to step 20.
  16. If *updatedNetworkSettings.IPv6.Enabled* = false, FAIL the test and go to step 20.
  17. If *updatedNetworkSettings.IPv6.Config.Manual* is skipped, FAIL the test and go to step 20.
  18. If *updatedNetworkSettings.IPv6.Config.Manual* is not equal to address at step 6, FAIL the test and go to step 20.
  19. If *updatedNetworkSettings.IPv6.Config.DHCP* != Off, FAIL the test and go to step 20.

20. ONVIF Client restores the original settings by following the procedure mentioned in [Annex A.8](#) with the following input and output parameters

- in *defaultNetworkSettings* - Original default network settings

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetNetworkInterfacesResponse** message.
- DUT did not send **SetNetworkInterfacesResponse** message.
- DUT did not send **SystemRebootResponse** message.
- DUT did not send **Hello** message during *rebootTimeout*.

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

## 5.2.2 IPV6 STATELESS IP CONFIGURATION - ROUTER ADVERTISEMENT

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

**Specification Coverage:** IP Configuration (ONVIF Core Specification)

**Feature Under Test:** Static IPv6 Stateless Configuration which Accepts Router Advertisement

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** IPv6 is supported by DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetNetworkInterfaces** request.

4. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
  - NetworkInterfaces list =: *networkInterfacesList1*
5. Set *defaultNetworkSettings* := interface from *networkInterfacesList1* list with specified IPv6.
6. ONVIF Client invokes **SetNetworkInterfaces** request with parameters
  - InterfaceToken := *defaultNetworkSettings.@token*
  - NetworkInterface.Enabled := true
  - NetworkInterface.Link skipped
  - NetworkInterface.MTU skipped
  - NetworkInterface.IPv4 skipped
  - NetworkInterface.IPv6.Enabled := true
  - NetworkInterface.IPv6.AcceptRouterAdvert := true
  - NetworkInterface.IPv6.Manual skipped
  - NetworkInterface.IPv6.DHCP := Off
  - NetworkInterface.Extension skipped
7. The DUT responds with **SetNetworkInterfacesResponse** message with parameters
  - RebootNeeded =: *rebootNeeded*
8. If *rebootNeeded* = true:
  - 8.1. ONVIF Client invokes **SystemReboot** request.
  - 8.2. The DUT responds with **SystemRebootResponse** message with parameters
    - Message
9. The DUT sends **Hello** message from a newly configured address:
  - EndpointReference
  - Types =: *types*
  - Scopes
  - XAddrs =: *xAddrs*

- MetadataVersion
10. If *types* is skipped or empty, FAIL the test and go to step 20.
  11. If *xAddrs* is skipped or empty, FAIL the test and go to step 20.
  12. ONVIF Client invokes **GetNetworkInterfaces** request.
  13. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
    - NetworkInterfaces list =: *networkInterfacesList2*
  14. Set *updatedNetworkSettings* := item from *networkInterfacesList2* list with @token = *defaultNetworkSettings.@token*.
  15. If *updatedNetworkSettings.IPv6* is skipped or empty, FAIL the test and go to step 20.
  16. If *updatedNetworkSettings.IPv6.Enabled* = false, FAIL the test and go to step 20.
  17. If *updatedNetworkSettings.IPv6.Config.AcceptRouterAdvert* = false, FAIL the test and go to step 20.
  18. If *updatedNetworkSettings.IPv6.Config.FromRA* is skipped, FAIL the test and go to step 20.
  19. If *updatedNetworkSettings.IPv6.Config.DHCP* != Off, FAIL the test and go to step 20.
  20. ONVIF Client restores the original settings by following the procedure mentioned in [Annex A.8](#) with the following input and output parameters
    - in *defaultNetworkSettings* - Original default network settings

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetNetworkInterfacesResponse** message.
- DUT did not send **SetNetworkInterfacesResponse** message.
- DUT did not send **SystemRebootResponse** message.
- DUT did not send **Hello** message during *rebootTimeout*.

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



## 5.2.3 IPV6 STATELESS IP CONFIGURATION - NEIGHBOUR DISCOVERY

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

**Specification Coverage:** IP Configuration (ONVIF Core Specification)

**Feature Under Test:** Static IPv6 Stateless Configuration which uses Neighbor Discovery

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requirement:** IPv6 is supported by DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetNetworkInterfaces** request.
4. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
  - NetworkInterfaces list := *networkInterfacesList1*
5. Set *defaultNetworkSettings* := interface from *networkInterfacesList1* list with specified IPv6.
6. ONVIF Client invokes **SetNetworkInterfaces** request with parameters
  - InterfaceToken := *defaultNetworkSettings.@token*
  - NetworkInterface.Enabled := true
  - NetworkInterface.Link skipped
  - NetworkInterface.MTU skipped
  - NetworkInterface.IPv4 skipped
  - NetworkInterface.IPv6.Enabled := true
  - NetworkInterface.IPv6.AcceptRouterAdvert skipped
  - NetworkInterface.IPv6.Manual skipped

- NetworkInterface.IPv6.DHCP := Off
  - NetworkInterface.Extension skipped
7. The DUT responds with **SetNetworkInterfacesResponse** message with parameters
    - RebootNeeded =: *rebootNeeded*
  8. If *rebootNeeded* = true:
    - 8.1. ONVIF Client invokes **SystemReboot** request.
    - 8.2. The DUT responds with **SystemRebootResponse** message with parameters
      - Message
  9. The DUT sends **Hello** message from a newly configured address:
    - EndpointReference
    - Types =: *types*
    - Scopes
    - XAddrs =: *xAddrs*
    - MetadataVersion
  10. If *types* is skipped or empty, FAIL the test and go to step 19.
  11. If *xAddrs* is skipped or empty, FAIL the test and go to step 19.
  12. ONVIF Client invokes **GetNetworkInterfaces** request.
  13. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
    - NetworkInterfaces list =: *networkInterfacesList2*
  14. Set *updatedNetworkSettings* := item from *networkInterfacesList2* list with @token = *defaultNetworkSettings.@token*.
  15. If *updatedNetworkSettings.IPv6* is skipped or empty, FAIL the test and go to step 19.
  16. If *updatedNetworkSettings.IPv6.Enabled* = false, FAIL the test and go to step 19.
  17. If *updatedNetworkSettings.IPv6.Config.LinkLocal* is skipped, FAIL the test and go to step 19.
  18. If *updatedNetworkSettings.IPv6.Config.DHCP* != Off, FAIL the test and go to step 19.

19. ONVIF Client restores the original settings by following the procedure mentioned in [Annex A.8](#) with the following input and output parameters

- in *defaultNetworkSettings* - Original default network settings

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetNetworkInterfacesResponse** message.
- DUT did not send **SetNetworkInterfacesResponse** message.
- DUT did not send **SystemRebootResponse** message.
- DUT did not send **Hello** message during *rebootTimeout*.

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

## 5.2.4 IPV6 STATEFUL IP CONFIGURATION

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

**Specification Coverage:** IP Configuration (ONVIF Core Specification)

**Feature Under Test:** IPv6 Stateful Configuration (DHCP)

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requirement:** IPv6 is supported by DUT.

**Test Configuration:** ONVIF Client, DUT, and DHCPv6 Server

**Test Procedure:**

1. Start DHCPv6 server.
2. Start an ONVIF Client.
3. Start the DUT.
4. ONVIF Client invokes **GetNetworkInterfaces** request.

5. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
  - NetworkInterfaces list =: *networkInterfacesList1*
6. Set *defaultNetworkSettings* := interface from *networkInterfacesList1* list with specified IPv6.
7. ONVIF Client invokes **SetNetworkInterfaces** request with parameters
  - InterfaceToken := *defaultNetworkSettings.@token*
  - NetworkInterface.Enabled := true
  - NetworkInterface.Link skipped
  - NetworkInterface.MTU skipped
  - NetworkInterface.IPv4 skipped
  - NetworkInterface.IPv6.Enabled := true
  - NetworkInterface.IPv6.AcceptRouterAdvert skipped
  - NetworkInterface.IPv6.Manual skipped
  - NetworkInterface.IPv6.DHCP := Stateful
  - NetworkInterface.Extension skipped
8. The DUT responds with **SetNetworkInterfacesResponse** message with parameters
  - RebootNeeded =: *rebootNeeded*
9. If *rebootNeeded* = true:
  - 9.1. ONVIF Client invokes **SystemReboot** request.
  - 9.2. The DUT responds with **SystemRebootResponse** message with parameters
    - Message
10. The DUT sends **Hello** message from a newly configured address:
  - EndpointReference
  - Types =: *types*
  - Scopes
  - XAddrs =: *xAddrs*

- MetadataVersion
11. If *types* is skipped or empty, FAIL the test and go to step 20.
  12. If *xAddrs* is skipped or empty, FAIL the test and go to step 20.
  13. ONVIF Client invokes **GetNetworkInterfaces** request.
  14. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
    - NetworkInterfaces list =: *networkInterfacesList2*
  15. Set *updatedNetworkSettings* := item from *networkInterfacesList2* list with @token = *defaultNetworkSettings.@token*.
  16. If *updatedNetworkSettings.IPv6* is skipped or empty, FAIL the test and go to step 20.
  17. If *updatedNetworkSettings.IPv6.Enabled* = false, FAIL the test and go to step 20.
  18. If *updatedNetworkSettings.IPv6.Config.FromDHCP* is skipped, FAIL the test and go to step 20.
  19. If *updatedNetworkSettings.IPv6.Config.DHCP* != Stateful, FAIL the test and go to step 20.
  20. ONVIF Client restores the original settings by following the procedure mentioned in [Annex A.8](#) with the following input and output parameters
    - in *defaultNetworkSettings* - Original default network settings

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetNetworkInterfacesResponse** message.
- DUT did not send **SetNetworkInterfacesResponse** message.
- DUT did not send **SystemRebootResponse** message.
- DUT did not send **Hello** message during *rebootTimeout*.

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

## 6 Device Discovery Test Cases

### 6.1 HELLO MESSAGE

**Test Case ID:** DISCOVERY-1-1-1

**Specification Coverage:** Hello, Reboot

**Feature Under Test:** Hello

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

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **SystemReboot** request to reboot the DUT.
4. DUT sends **SystemRebootResponse** message.
5. ONVIF Client waits for the user-defined boot time to receive **Hello** message from DUT.
6. Verify that the DUT transmits the **Hello** message with multicast address 239.255.255.250, and port number 3702.
7. In case of device discovery with IPv6 interface, verify that the DUT transmits the **Hello** message with multicast address FF02::C (link-local scope), and port number 3702.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **SystemRebootResponse** message.
- The DUT did not send the multicast **Hello** message.

## 6.2 HELLO MESSAGE VALIDATION

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

**Specification Coverage:** Endpoint reference, Hello, Types, Scopes, Reboot

**Feature Under Test:** Hello

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

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

**Test Result:**

**PASS –**

- 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.

**Note:** See [Annex A.1](#) for Device and Scope Types definition.

## 6.3 SEARCH BASED ON DEVICE SCOPE TYPES

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

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

**Feature Under Test:** Probe, ProbeMatch

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

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetScopes** request to retrieve existing scopes list.
4. DUT replies with the list of scopes types in **GetScopesResponse** message.
5. ONVIF Client will transmit the multicast **Probe** request with different scope types (type, location, hardware and name).
6. ONVIF Client will verify the **ProbeMatch** 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 entries.



- The DUT did not send mandatory XML elements (device type, scope list, service address and scope matching rule) in the **ProbeMatch** message.
- The DUT did not send **ProbeMatch** message within the time out period of DISCOVERY\_TIMEOUT.
- The DUT did not send **ProbeMatch** message with a namespace of the Types value.
- The DUT did not send **ProbeMatch** message with fixed entry point to <d:XAddr> element ("http://<onvif\_host>/onvif/device\_service").
- In case of IPv6, the DUT did not send **ProbeMatch** message with <d:XAddr> including IPv6 address.
- The DUT did not send **ProbeMatch** message with a correct XML element RelatesTo that has the same value as MessageID of **Probe** request (not to omit "urn" namespace).

**Note:** See [Annex A.1](#) for Device and Scope Types definition.

**Note:** See [Annex A.7](#) for the value of DISCOVERY\_TIMEOUT.

## 6.4 SEARCH WITH OMITTED DEVICE AND SCOPE TYPES

**Test Case ID:** DISCOVERY-1-1-4

**Specification Coverage:** Services overview, Types, Scopes, Probe and Probe Match

**Feature Under Test:** Probe, ProbeMatch

**WSDL Reference:** ws-discovery.wsdl

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **ProbeMatch** message within the time out period of DISCOVERY\_TIMEOUT.
- The DUT did not send mandatory XML elements (device type, scope list, service address and scope matching rule) in the **ProbeMatch** message.
- The DUT did not send **ProbeMatch** message with a namespace of the Types value.
- The DUT did not send **ProbeMatch** message with fixed entry point to <d:XAddr> element ("http://<onvif\_host>/onvif/device\_service").
- In case of IPv6, the DUT did not send **ProbeMatch** message with <d:XAddr> of including IPv6 address.
- The DUT did not send **ProbeMatch** message with a correct XML element RelatesTo that it has same value as MessageID of **Probe** request (not to omit "urn" namespace).

**Note:** See [Annex A.1](#) for Device and Scope Types definition.

**Note:** See [Annex A.7](#) for the value of DISCOVERY\_TIMEOUT.

## 6.5 RESPONSE TO INVALID SEARCH REQUEST

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

**Specification Coverage:** Probe and Probe Match

**Feature Under Test:** Probe

**WSDL Reference:** ws-discovery.wsdl

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did send **ProbeMatch** message.

**Note:** See [Annex A.1](#) for Device and Scope Types definition.

## 6.6 SEARCH USING UNICAST PROBE MESSAGE

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

**Test Purpose:** To verify DUT behavior for Unicast Probe request.

**Note:** All Tests [DISCOVERY-1-1-3](#), [DISCOVERY-1-1-4](#), [DISCOVERY-1-1-5](#) to be repeated with Unicast Probe request.

## 6.7 BYE MESSAGE

**Test Case ID:** DISCOVERY-1-1-8

**Specification Coverage:** Bye, Reboot

**Feature Under Test:** Bye

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

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **SystemRebootResponse** message.
- The DUT did not send **Bye** message.

## 6.8 DISCOVERY MODE CONFIGURATION

**Test Case ID:** DISCOVERY-1-1-9

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

**Feature Under Test:** GetDiscoveryMode, SetDiscoveryMode

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

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

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

- The DUT **GetDiscoveryModeResponse** did not have a Discovery mode parameter of NonDiscoverable at step 8.
- The DUT sent **ProbeMatch** message within the time out period of DISCOVERY\_TIMEOUT.
- The DUT did not send **SystemRebootResponse** message.
- The DUT sent **Bye** or **Hello** message at step 13.

**Note:** See [Annex A.7](#) for the value of DISCOVERY\_TIMEOUT.

## 6.9 SOAP FAULT MESSAGE

**Test Case ID:** DISCOVERY-1-1-10

**Specification Coverage:** SOAP Fault Messages

**Feature Under Test:** Probe

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

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will transmit Unicast **Probe** request 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.

**Note:** Refer [RFC 3986] for scope matching definitions.

## 6.10 DEVICE SCOPES CONFIGURATION

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

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

**Feature Under Test:** AddScopes, SetScopes, RemoveScopes

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

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetScopes** request to retrieve existing scope types.
4. The DUT replies with the list of scopes types in the **GetScopesResponse** message.
5. ONVIF Client will invoke **SetScopes** request to replace existing list with a list of new scope.
6. The DUT replies with **SetScopesResponse** message indicating success.
7. ONVIF Client waits for 2 seconds.
8. ONVIF Client will invoke **AddScopes** request to add new scope types to the existing scope list.
9. The DUT replies with **AddScopesResponse** message indicating success.
10. The DUT sends Multicast **Hello** message to indicate the change in the metadata (i.e. addition of new scope types to the existing list).
11. ONVIF Client will invoke Multicast **Probe** request to search the DUT with newly added scope types.

12. Verify that the DUT issued a **ProbeMatch** message.
13. ONVIF Client will invoke **RemoveScopes** request to delete the newly configured scope types.
14. The DUT replies with **RemoveScopesResponse** message indicating success.
15. The DUT sends Multicast **Hello** message to indicate the change in the metadata (i.e. deletion of scope types from the existing list).
16. ONVIF Client will invoke Multicast **Probe** request to search the DUT with deleted scope types.
17. Verify that the DUT did not send **ProbeMatch** 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 **SetScopesResponse** message.
- The DUT did not send **AddScopesResponse** message.
- The DUT did not send multicast **Hello** message after the change in its metadata (addition/deletion of scope types) with a complete list of current scopes.
- The DUT did not send mandatory XML elements (device, new scope type, service address and scope matching rule) in the **ProbeMatch** message.
- The DUT did not send **RemoveScopesResponse** message.
- The DUT did not send **ProbeMatch** message within the time out period of DISCOVERY\_TIMEOUT.

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

**Note:** Whenever there is a change in the metadata of the Target Service, "MetadataVersion" is incremented by  $\geq 1$ .



**Note:** See [Annex A.4](#) for Invalid SOAP 1.2 fault message definition.

**Note:** Refer [RFC 3986] for scope matching definitions.

**Note:** See [Annex A.7](#) for the value of DISCOVERY\_TIMEOUT.

## 6.11 Namespace Handling

### 6.11.1 DISCOVERY - NAMESPACES (DEFAULT NAMESPACES FOR EACH TAG)

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

**Specification Coverage:** None.

**Feature Under Test:** XML namespaces definition

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

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetScopes** request to retrieve existing scopes list.
4. DUT replies with the list of scopes types in **GetScopesResponse** message.
5. ONVIF Client will transmit the multicast **Probe** request with different scope types (type, location, hardware and name).
6. ONVIF Client will verify the **ProbeMatch** 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 entries.
- The DUT did not send mandatory XML elements (device type, scope list, service address and scope matching rule) in the **ProbeMatch** message.
- The DUT did not send **ProbeMatch** message within the time out period of DISCOVERY\_TIMEOUT.
- The DUT did not send **ProbeMatch** message with a namespace of the Types value.
- The DUT did not send **ProbeMatch** message with fixed entry point to <d:XAddr> element ("http://<onvif\_host>/onvif/device\_service").
- In case of IPv6, the DUT did not send **ProbeMatch** message with <d:XAddr> including IPv6 address.
- The DUT did not send **ProbeMatch** message with a correct XML element RelatesTo that has the same value as MessageID of **Probe** request (not to omit "urn" namespace).

**Note:** See [Annex A.1](#) for Device and Scope Types definition.

**Note:** See [Annex A.7](#) for the value of DISCOVERY\_TIMEOUT.

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

## 6.11.2 DISCOVERY - NAMESPACES (DEFAULT NAMESPACES FOR PARENT TAG)

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

**Specification Coverage:** None.

**Feature Under Test:** XML namespaces definition

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

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetScopes** request to retrieve existing scopes list.
4. DUT replies with the list of scopes types in **GetScopesResponse** message.
5. ONVIF Client will transmit the multicast **Probe** request with different scope types (type, location, hardware and name).
6. ONVIF Client will verify the **ProbeMatch** 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 entries.
- The DUT did not send mandatory XML elements (device type, scope list, service address and scope matching rule) in the **ProbeMatch** message.
- The DUT did not send **ProbeMatch** message within the time out period of DISCOVERY\_TIMEOUT.
- The DUT did not send **ProbeMatch** message with a namespace of the Types value.
- The DUT did not send **ProbeMatch** message with fixed entry point to <d:XAddr> element ("http://<onvif\_host>/onvif/device\_service").
- In case of IPv6, the DUT did not send **ProbeMatch** message with <d:XAddr> including IPv6 address.
- The DUT did not send **ProbeMatch** message with a correct XML element RelatesTo that has the same value as MessageID of **Probe** request (not to omit "urn" namespace).

**Note:** See [Annex A.1](#) for Device and Scope Types definition.

**Note:** See [Annex A.7](#) for the value of DISCOVERY\_TIMEOUT.

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

### 6.11.3 DISCOVERY - NAMESPACES (NOT STANDARD PREFIXES)

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

**Specification Coverage:** None.

**Feature Under Test:** XML namespaces definition

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

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

**Pre-Requirement:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetScopes** request to retrieve existing scopes list.
4. DUT replies with the list of scopes types in **GetScopesResponse** message.
5. ONVIF Client will transmit the multicast **Probe** request with different scope types (type, location, hardware and name).
6. ONVIF Client will verify the **ProbeMatch** 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 entries.
- The DUT did not send mandatory XML elements (device type, scope list, service address and scope matching rule) in the **ProbeMatch** message.
- The DUT did not send **ProbeMatch** message within the time out period of DISCOVERY\_TIMEOUT.
- The DUT did not send **ProbeMatch** message with a namespace of the Types value.
- The DUT did not send **ProbeMatch** message with fixed entry point to <d:XAddr> element ("http://<onvif\_host>/onvif/device\_service").
- In case of IPv6, the DUT did not send **ProbeMatch** message with <d:XAddr> including IPv6 address.
- The DUT did not send **ProbeMatch** message with a correct XML element RelatesTo that has the same value as MessageID of **Probe** request (not to omit "urn" namespace).

**Note:** See [Annex A.1](#) for Device and Scope Types definition.

**Note:** See [Annex A.7](#) for the value of DISCOVERY\_TIMEOUT.

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

## 6.11.4 DISCOVERY - NAMESPACES (DIFFERENT PREFIXES FOR THE SAME NAMESPACE)

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

**Specification Coverage:** None.

**Feature Under Test:** XML namespaces definition

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

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetScopes** request to retrieve existing scopes list.
4. DUT replies with the list of scopes types in **GetScopesResponse** message.
5. ONVIF Client will transmit the multicast **Probe** request with different scope types (type, location, hardware and name).
6. ONVIF Client will verify the **ProbeMatch** 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 entries.
- The DUT did not send mandatory XML elements (device type, scope list, service address and scope matching rule) in the **ProbeMatch** message.
- The DUT did not send **ProbeMatch** message within the time out period of DISCOVERY\_TIMEOUT.
- The DUT did not send **ProbeMatch** message with a namespace of the Types value.
- The DUT did not send **ProbeMatch** message with fixed entry point to <d:XAddr> element ("http://<onvif\_host>/onvif/device\_service").
- In case of IPv6, the DUT did not send **ProbeMatch** message with <d:XAddr> including IPv6 address.
- The DUT did not send **ProbeMatch** message with a correct XML element RelatesTo that has the same value as MessageID of **Probe** request (not to omit "urn" namespace).

**Note:** See [Annex A.1](#) for Device and Scope Types definition.

**Note:** See [Annex A.7](#) for the value of DISCOVERY\_TIMEOUT.

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

## 6.11.5 DISCOVERY - NAMESPACES (THE SAME PREFIX FOR DIFFERENT NAMESPACES)

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

**Specification Coverage:** None.

**Feature Under Test:** XML namespaces definition

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

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetScopes** request to retrieve existing scopes list.
4. DUT replies with the list of scopes types in **GetScopesResponse** message.
5. ONVIF Client will transmit the multicast **Probe** request with different scope types (type, location, hardware and name).
6. ONVIF Client will verify the **ProbeMatch** 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 entries.
- The DUT did not send mandatory XML elements (device type, scope list, service address and scope matching rule) in the **ProbeMatch** message.

- The DUT did not send **ProbeMatch** message within the time out period of DISCOVERY\_TIMEOUT.
- The DUT did not send **ProbeMatch** message with a namespace of the Types value.
- The DUT did not send **ProbeMatch** message with fixed entry point to <d:XAddr> element ("http://<onvif\_host>/onvif/device\_service").
- In case of IPv6, the DUT did not send **ProbeMatch** message with <d:XAddr> including IPv6 address.
- The DUT did not send **ProbeMatch** message with a correct XML element RelatesTo that has the same value as MessageID of **Probe** request (not to omit "urn" namespace).

**Note:** See [Annex A.1](#) for Device and Scope Types definition.

**Note:** See [Annex A.7](#) for the value of DISCOVERY\_TIMEOUT.

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



## 7 Device Management Test Cases

### 7.1 Capabilities

#### 7.1.1 GET WSDL URL

**Test Case ID:** DEVICE-1-1-1

**Specification Coverage:** Get WSDL URL

**Feature Under Test:** GetWsdUrl

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetWsdUrl** request.
4. The DUT responds with **GetWsdUrlResponse** message with parameters
  - WsdUrl := *wsdlUrl*
5. If *wsdlUrl* is not valid URI, FAIL the test.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

#### 7.1.2 ALL CAPABILITIES

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

**Specification Coverage:** Capability exchange

**Feature Under Test:** GetCapabilities

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify all Capabilities of the DUT.

**Pre-Requisite:** GetCapabilities is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetCapabilities** message with parameters:
  - Category[0] := All
4. The DUT responds with a **GetCapabilitiesResponse** message with parameters:
  - Capabilities =: *capabilities1*
5. If *capabilities1.Device* is not specified, FAIL the test and skip other steps.
6. If *capabilities1.Events* is not specified, FAIL the test and skip other steps.
7. If Media is supported by the DUT and *capabilities1.Media* is not specified, FAIL the test and skip other steps.
8. If PTZ is supported by the DUT and *capabilities1.PTZ* is not specified, FAIL the test and skip other steps.
9. If Imaging is supported by the DUT and *capabilities1.Imaging* is not specified, FAIL the test and skip other steps.
10. If Analytics is supported by the DUT and *capabilities1.Analytics* is not specified, FAIL the test and skip other steps.
11. If DeviceIO is supported by the DUT and *capabilities1.Extension.DeviceIO* is not specified, FAIL the test and skip other steps.
12. ONVIF Client invokes **GetCapabilities** message with parameters:
  - Category list skipped
13. The DUT responds with a **GetCapabilitiesResponse** message with parameters:

- Capabilities =: *capabilities1*
14. If *capabilities1.Device* is not specified, FAIL the test and skip other steps.
  15. If *capabilities1.Events* is not specified, FAIL the test and skip other steps.
  16. If Media is supported by the DUT and *capabilities1.Media* is not specified, FAIL the test and skip other steps.
  17. If PTZ is supported by the DUT and *capabilities1.PTZ* is not specified, FAIL the test and skip other steps.
  18. If Imaging is supported by the DUT and *capabilities1.Imaging* is not specified, FAIL the test and skip other steps.
  19. If Analytics is supported by the DUT and *capabilities1.Analytics* is not specified, FAIL the test and skip other steps.
  20. If DeviceIO is supported by the DUT and *capabilities1.Extension.DeviceIO* is not specified, FAIL the test and skip other steps.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.3 DEVICE CAPABILITIES

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

**Specification Coverage:** Capability exchange.

**Feature Under Test:** GetCapabilities

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify Device Capabilities of the DUT.

**Pre-Requisite:** GetCapabilities is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT.

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetCapabilities** message with parameters:
  - Category[0] := Device
4. The DUT responds with a **GetCapabilitiesResponse** message with parameters:
  - Capabilities =: *capabilities*
5. If *capabilities.Device* is not specified, FAIL the test and skip other steps.
6. If *capabilities.Device.XAddr* is not valid URI, FAIL the test and skip other steps.
7. If *capabilities.Analytics* is specified, FAIL the test and skip other steps.
8. If *capabilities.Events* is specified, FAIL the test and skip other steps.
9. If *capabilities.Imaging* is specified, FAIL the test and skip other steps.
10. If *capabilities.Media* is specified, FAIL the test and skip other steps.
11. If *capabilities.PTZ* is specified, FAIL the test and skip other steps.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.4 MEDIA CAPABILITIES

**Test Case ID:** DEVICE-1-1-4**Specification Coverage:** Capability exchange**Feature Under Test:** GetCapabilities**WSDL Reference:** devicemgmt.wsdl**Test Purpose:** To verify Media Capabilities of the DUT.**Pre-Requisite:** GetCapabilities is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT.**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetCapabilities** message with parameters:
  - Category[0] := Media
4. If Media Service is supported by the DUT:
  - 4.1. The DUT responds with a **GetCapabilitiesResponse** message with parameters:
    - Capabilities =: *capabilities*
  - 4.2. If *capabilities*.Media is not specified, FAIL the test and skip other steps.
  - 4.3. If *capabilities*.Media.XAddr is not valid URI, FAIL the test and skip other steps.
  - 4.4. If *capabilities*.Analytics is specified, FAIL the test and skip other steps.
  - 4.5. If *capabilities*.Device is specified, FAIL the test and skip other steps.
  - 4.6. If *capabilities*.Imaging is specified, FAIL the test and skip other steps.
  - 4.7. If *capabilities*.Events is specified, FAIL the test and skip other steps.
  - 4.8. If *capabilities*.PTZ is specified, FAIL the test and skip other steps.
5. If Media Service is not supported by the DUT:
  - 5.1. The DUT returns **env:Receiver/ter:ActionNotSupported/ter:NoSuchService** SOAP 1.2 fault.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send the **env:Receiver/ter:ActionNotSupported/ter:NoSuchService** SOAP 1.2 fault message, if Media Service is not supported by the DUT.
- The DUT did not send **GetCapabilitiesResponse** message, if Media Service is supported by the DUT.

## 7.1.5 EVENT CAPABILITIES

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

**Specification Coverage:** Capability exchange.

**Feature Under Test:** GetCapabilities.

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify event capabilities of the DUT.

**Pre-Requisite:** GetCapabilities is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT.

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetCapabilities** message with parameters:
  - Category[0] := Events
4. The DUT responds with a **GetCapabilitiesResponse** message with parameters:
  - Capabilities =: *capabilities*
5. If *capabilities.Events* is not specified, FAIL the test and skip other steps.
6. If *capabilities.Events.XAddr* is not valid URI, FAIL the test and skip other steps.
7. If *capabilities.Analytics* is specified, FAIL the test and skip other steps.
8. If *capabilities.Device* is specified, FAIL the test and skip other steps.
9. If *capabilities.Imaging* is specified, FAIL the test and skip other steps.
10. If *capabilities.Media* is specified, FAIL the test and skip other steps.
11. If *capabilities.PTZ* is specified, FAIL the test and skip other steps.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.6 PTZ CAPABILITIES

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

**Specification Coverage:** Capability exchange.

**Feature Under Test:** GetCapabilities.

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify PTZ capabilities of the DUT.

**Pre-Requisite:** GetCapabilities is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT.

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetCapabilities** message with parameters:
  - Category[0] := PTZ
4. If PTZ Service is supported by the DUT:
  - 4.1. The DUT responds with a **GetCapabilitiesResponse** message with parameters:
    - Capabilities =: *capabilities*
  - 4.2. If *capabilities*.PTZ is not specified, FAIL the test and skip other steps.
  - 4.3. If *capabilities*.PTZ.XAddr is not valid URI, FAIL the test and skip other steps.
  - 4.4. If *capabilities*.Analytics is specified, FAIL the test and skip other steps.
  - 4.5. If *capabilities*.Device is specified, FAIL the test and skip other steps.
  - 4.6. If *capabilities*.Imaging is specified, FAIL the test and skip other steps.
  - 4.7. If *capabilities*.Events is specified, FAIL the test and skip other steps.
  - 4.8. If *capabilities*.Media is specified, FAIL the test and skip other steps.
5. If PTZ Service is not supported by the DUT:

- 5.1. The DUT returns **env:Receiver/ter:ActionNotSupported/ter:NoSuchService** SOAP 1.2 fault.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send the **env:Receiver/ter:ActionNotSupported/ter:NoSuchService** SOAP 1.2 fault message, if PTZ Service is not supported by the DUT.
- The DUT did not send **GetCapabilitiesResponse** message, if PTZ Service is supported by the DUT.

## 7.1.7 SOAP FAULT MESSAGE

**Test Case ID:** DEVICE-1-1-9

**Feature Under Test:** GetCapabilities

**Specification Coverage:** Capability exchange.

**WSDL Reference:** devicemgmt.wsdl

**Pre-Requisite:** GetCapabilities is supported by the DUT.

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

**Test Configuration:** ONVIF Client and DUT.

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetCapabilities** message with parameters:
  - Category[0] := invalid capability category
4. The DUT returns SOAP 1.2 fault.

**Test Result:****PASS –**



- DUT passes all assertions.

**FAIL –**

- The DUT did not send the SOAP 1.2 fault message.

**Note:** See [Annex A.4](#) for Invalid SOAP 1.2 fault message definition.

## 7.1.8 IMAGING CAPABILITIES

**Test Case ID:** DEVICE-1-1-10

**Specification Coverage:** Capability exchange

**Feature Under Test:** GetCapabilities

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify Imaging Capabilities of the DUT.

**Pre-Requirement:** GetCapabilities is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT.

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetCapabilities** message with parameters:
  - Category[0] := Imaging
4. If Imaging Service is supported by the DUT:
  - 4.1. The DUT responds with a **GetCapabilitiesResponse** message with parameters:
    - Capabilities =: *capabilities*
  - 4.2. If *capabilities.Imaging* is not specified, FAIL the test and skip other steps.
  - 4.3. If *capabilities.Imaging.XAddr* is not valid URI, FAIL the test and skip other steps.
  - 4.4. If *capabilities.Analytics* is specified, FAIL the test and skip other steps.
  - 4.5. If *capabilities.Device* is specified, FAIL the test and skip other steps.
  - 4.6. If *capabilities.PTZ* is specified, FAIL the test and skip other steps.

- 4.7. If *capabilities.Events* is specified, FAIL the test and skip other steps.
- 4.8. If *capabilities.Media* is specified, FAIL the test and skip other steps.
5. If Imaging Service is not supported by the DUT:
  - 5.1. The DUT returns **env:Receiver/ter:ActionNotSupported/ter:NoSuchService** SOAP 1.2 fault.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send the **env:Receiver/ter:ActionNotSupported/ter:NoSuchService** SOAP 1.2 fault message, if Imaging Service is not supported by the DUT.
- The DUT did not send **GetCapabilitiesResponse** message, if Imaging Service is supported by the DUT.

## 7.1.9 ANALYTICS CAPABILITIES

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

**Specification Coverage:** Capability exchange

**Feature Under Test:** GetCapabilities

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify Analytics Capabilities of the DUT.

**Pre-Requisite:** GetCapabilities is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT.

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetCapabilities** message with parameters:
  - Category[0] := Analytics

4. If Analytics Service is supported by the DUT:
  - 4.1. The DUT responds with a **GetCapabilitiesResponse** message with parameters:
    - Capabilities =: *capabilities*
  - 4.2. If *capabilities.Analytics* is not specified, FAIL the test and skip other steps.
  - 4.3. If *capabilities.Analytics.XAddr* is not valid URI, FAIL the test and skip other steps.
  - 4.4. If *capabilities.Imaging* is specified, FAIL the test and skip other steps.
  - 4.5. If *capabilities.Device* is specified, FAIL the test and skip other steps.
  - 4.6. If *capabilities.PTZ* is specified, FAIL the test and skip other steps.
  - 4.7. If *capabilities.Events* is specified, FAIL the test and skip other steps.
  - 4.8. If *capabilities.Media* is specified, FAIL the test and skip other steps.
5. If Analytics Service is not supported by the DUT:
  - 5.1. The DUT returns **env:Receiver/ter:ActionNotSupported/ter:NoSuchService** SOAP 1.2 fault.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send the **env:Receiver/ter:ActionNotSupported/ter:NoSuchService** SOAP 1.2 fault message, if Analytics Service is not supported by the DUT.
- The DUT did not send **GetCapabilitiesResponse** message, if Analytics Service is supported by the DUT.

## 7.1.10 GET SERVICES – DEVICE SERVICE

**Test Case ID:** DEVICE-1-1-13**Specification Coverage:** Capability exchange**Feature Under Test:** GetServices**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify that Device Management Service is received using GetServices request.

**Pre-Requisite:** GetServices is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := false
4. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithoutCapabilities*
5. If *listOfServicesWithoutCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/device/wsd", FAIL the test and skip other steps.
6. Set *deviceServ* := item from *listOfServicesWithoutCapabilities* list with Namespace = "http://www.onvif.org/ver10/device/wsd".
7. If *deviceServ.Capabilities* is specified, FAIL the test and skip other steps.
8. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := true
9. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithCapabilities*
10. If *listOfServicesWithCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/device/wsd", FAIL the test and skip other steps.
11. Set *deviceServ* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver10/device/wsd".
12. If *deviceServ.Capabilities* is not specified, FAIL the test and skip other steps.
13. If *deviceServ.Capabilities* does not contain valid Capabilities element for Device Management service from "http://www.onvif.org/ver10/device/wsd" namespace, FAIL the test and skip other steps.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.11 GET SERVICES – MEDIA SERVICE

**Test Case ID:** DEVICE-1-1-14

**Specification Coverage:** Capability exchange

**Feature Under Test:** GetServices

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify that Media Service is received using GetServices request.

**Pre-Requirement:** GetServices is supported by the DUT. Media Service is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := false
4. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithoutCapabilities*
5. If *listOfServicesWithoutCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/media/wsdl", FAIL the test and skip other steps.
6. Set *mediaServ* := item from *listOfServicesWithoutCapabilities* list with Namespace = "http://www.onvif.org/ver10/media/wsdl".
7. If *mediaServ.Capabilities* is specified, FAIL the test and skip other steps.
8. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := true

9. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithCapabilities*
10. If *listOfServicesWithCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/media/wsd", FAIL the test and skip other steps.
11. Set *mediaServ* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver10/media/wsd".
12. If *mediaServ.Capabilities* is not specified, FAIL the test and skip other steps.
13. If *mediaServ.Capabilities* does not contain valid Capabilities element for Media service from "http://www.onvif.org/ver10/media/wsd" namespace, FAIL the test and skip other steps.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.12 GET SERVICES – PTZ SERVICE

**Test Case ID:** DEVICE-1-1-15**Specification Coverage:** Capability exchange**Feature Under Test:** GetServices**WSDL Reference:** devicemgmt.wsd**Test Purpose:** To verify that PTZ Service is received using GetServices request.**Pre-Requisite:** GetServices is supported by the DUT. PTZ Service is supported by the DUT.**Test Configuration:** ONVIF Client and DUT**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := false

4. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithoutCapabilities*
5. If *listOfServicesWithoutCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver20/ptz/wsdl", FAIL the test and skip other steps.
6. Set *ptzServ* := item from *listOfServicesWithoutCapabilities* list with Namespace = "http://www.onvif.org/ver20/ptz/wsdl".
7. If *ptzServ.Capabilities* is specified, FAIL the test and skip other steps.
8. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := true
9. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithCapabilities*
10. If *listOfServicesWithCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver20/ptz/wsdl", FAIL the test and skip other steps.
11. Set *ptzServ* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver20/ptz/wsdl".
12. If *ptzServ.Capabilities* is not specified, FAIL the test and skip other steps.
13. If *ptzServ.Capabilities* does not contain valid Capabilities element for PTZ service from "http://www.onvif.org/ver20/ptz/wsdl" namespace, FAIL the test and skip other steps.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.13 GET SERVICES – EVENT SERVICE

**Test Case ID:** DEVICE-1-1-16**Specification Coverage:** Capability exchange**Feature Under Test:** GetServices

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify that Event Service is received using GetServices request.

**Pre-Requisite:** GetServices is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := false
4. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithoutCapabilities*
5. If *listOfServicesWithoutCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/events/wsdl", FAIL the test and skip other steps.
6. Set *eventsServ* := item from *listOfServicesWithoutCapabilities* list with Namespace = "http://www.onvif.org/ver10/events/wsdl".
7. If *eventsServ.Capabilities* is specified, FAIL the test and skip other steps.
8. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := true
9. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithCapabilities*
10. If *listOfServicesWithCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/events/wsdl", FAIL the test and skip other steps.
11. Set *eventsServ* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver10/events/wsdl".
12. If *eventsServ.Capabilities* is not specified, FAIL the test and skip other steps.
13. If *eventsServ.Capabilities* does not contain valid Capabilities element for Event service from "http://www.onvif.org/ver10/events/wsdl" namespace, FAIL the test and skip other steps.

**Test Result:**



**PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.14 GET SERVICES – IMAGING SERVICE

**Test Case ID:** DEVICE-1-1-17

**Specification Coverage:** Capability exchange

**Feature Under Test:** GetServices

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify that Imaging Service is received using GetServices request.

**Pre-Requirement:** GetServices is supported by the DUT. Imaging Service is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := false
4. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithoutCapabilities*
5. If *listOfServicesWithoutCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver20/imaging/wsdl", FAIL the test and skip other steps.
6. Set *imagingServ* := item from *listOfServicesWithoutCapabilities* list with Namespace = "http://www.onvif.org/ver20/imaging/wsdl".
7. If *imagingServ.Capabilities* is specified, FAIL the test and skip other steps.
8. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := true

9. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithCapabilities*
10. If *listOfServicesWithCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver20/imaging/wsdl", FAIL the test and skip other steps.
11. Set *imagingServ* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver20/imaging/wsdl".
12. If *imagingServ.Capabilities* is not specified, FAIL the test and skip other steps.
13. If *imagingServ.Capabilities* does not contain valid Capabilities element for Imaging service from "http://www.onvif.org/ver20/imaging/wsdl" namespace, FAIL the test and skip other steps.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.15 DEVICE SERVICE CAPABILITIES

**Test Case ID:** DEVICE-1-1-18

**Specification Coverage:** Capability exchange

**Feature Under Test:** GetServiceCapabilities (for Device Management service)

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** GetServices is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServiceCapabilities** message.

4. The DUT responds with a **GetServiceCapabilitiesResponse** message with parameters:
  - Capabilities

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.16 GET SERVICES AND GET DEVICE SERVICE CAPABILITIES CONSISTENCY

**Test Case ID:** DEVICE-1-1-19

**Specification Coverage:** Capability exchange

**Feature Under Test:** GetServices, GetServiceCapabilities (for Device Management Service)

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requirement:** GetServices is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := true
4. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithCapabilities*
5. Set *deviceServ* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver10/device/wsdl".
6. ONVIF Client invokes **GetServiceCapabilities** message.

7. The DUT responds with a **GetServiceCapabilitiesResponse** message with parameters:
  - Capabilities =: *deviceCapabilities*
8. If *deviceCapabilities* are not equal to *deviceServ.Capabilities.Capabilities*, FAIL the test and skip other steps.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

**Note:** The following fields are compared at step 8:

- Network
  - IPFilter
  - ZeroConfiguration
  - IPVersion6
  - DynDNS
  - Dot11Configuration
  - HostnameFromDHCP
  - NTP
- System
  - DiscoveryBye
  - DiscoveryResolve
  - FirmwareUpgrade
  - HttpFirmwareUpgrade
  - HttpSupportInformation
  - HttpSystemBackup

- HttpSystemLogging
- RemoteDiscovery
- SystemBackup
- SystemLogging
- Security
  - TLS1.0
  - TLS1.1
  - TLS1.2
  - OnboardKeyGeneration
  - AccessPolicyConfig
  - Dot1X
  - RemoteUserHandling
  - X.509Token
  - SAMLToken
  - KerberosToken
  - UsernameToken
  - HttpDigest
  - RELToken
  - DefaultAccessPolicy
  - SupportedEAPMethod
- Misc
  - AuxiliaryCommands

## 7.1.17 GET SERVICES – REPLAY SERVICE

**Test Case ID:** DEVICE-1-1-20

**Specification Coverage:** Capability exchange

**Feature Under Test:** GetServices

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify getting Replay Service with using GetServices request.

**Pre-Requisite:** GetServices is supported by the DUT. Replay Service is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := false
4. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithoutCapabilities*
5. If *listOfServicesWithoutCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/replay/wsdl", FAIL the test and skip other steps.
6. Set *replayServ* := item from *listOfServicesWithoutCapabilities* list with Namespace = "http://www.onvif.org/ver10/replay/wsdl".
7. If *replayServ.Capabilities* is specified, FAIL the test and skip other steps.
8. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := true
9. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithCapabilities*
10. If *listOfServicesWithCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/replay/wsdl", FAIL the test and skip other steps.
11. Set *replayServ* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver10/replay/wsdl".
12. If *replayServ.Capabilities* is not specified, FAIL the test and skip other steps.

13. If *replayServ.Capabilities* does not contain valid Capabilities element for Replay service from "http://www.onvif.org/ver10/replay/wsd" namespace, FAIL the test and skip other steps.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.18 GET SERVICES – RECORDING SEARCH SERVICE

**Test Case ID:** DEVICE-1-1-21

**Specification Coverage:** Capability exchange

**Feature Under Test:** GetServices

**WSDL Reference:** devicemgmt.wsd

**Test Purpose:** To verify that Recording Search Service is received using GetServices request.

**Pre-Requisite:** GetServices is supported by the DUT. Recording Search Service is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := false
4. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithoutCapabilities*
5. If *listOfServicesWithoutCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/search/wsd", FAIL the test and skip other steps.

6. Set *searchServ* := item from *listOfServicesWithoutCapabilities* list with Namespace = "http://www.onvif.org/ver10/search/wsdli".
7. If *searchServ.Capabilities* is specified, FAIL the test and skip other steps.
8. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := true
9. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithCapabilities*
10. If *listOfServicesWithCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/search/wsdli", FAIL the test and skip other steps.
11. Set *searchServ* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver10/search/wsdli".
12. If *searchServ.Capabilities* is not specified, FAIL the test and skip other steps.
13. If *searchServ.Capabilities* does not contain valid Capabilities element for Recording Search service from "http://www.onvif.org/ver10/search/wsdli" namespace, FAIL the test and skip other steps.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.19 GET SERVICES – RECORDING CONTROL SERVICE

**Test Case ID:** DEVICE-1-1-22

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

**Feature Under Test:** GetServices

**WSDL Reference:** devicemgmt.wsdli

**Test Purpose:** To verify that Recording Control Service is received using GetServices request.



**Pre-Requirement:** GetServices is supported by the DUT. Recording Control Service is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := false
4. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithoutCapabilities*
5. If *listOfServicesWithoutCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/recording/wsdl", FAIL the test and skip other steps.
6. Set *recordingServ* := item from *listOfServicesWithoutCapabilities* list with Namespace = "http://www.onvif.org/ver10/recording/wsdl".
7. If *recordingServ.Capabilities* is specified, FAIL the test and skip other steps.
8. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := true
9. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithCapabilities*
10. If *listOfServicesWithCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/recording/wsdl", FAIL the test and skip other steps.
11. Set *recordingServ* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver10/recording/wsdl".
12. If *recordingServ.Capabilities* is not specified, FAIL the test and skip other steps.
13. If *recordingServ.Capabilities* does not contain valid Capabilities element for Recording Control service from "http://www.onvif.org/ver10/recording/wsdl" namespace, FAIL the test and skip other steps.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.20 GET SERVICES – RECEIVER SERVICE

**Test Case ID:** DEVICE-1-1-23

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

**Feature Under Test:** GetServices

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify that Receiver Service is received using GetServices request.

**Pre-Requirement:** GetServices is supported by the DUT. Receiver Service is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := false
4. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithoutCapabilities*
5. If *listOfServicesWithoutCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/receiver/wsdl", FAIL the test and skip other steps.
6. Set *receiverServ* := item from *listOfServicesWithoutCapabilities* list with Namespace = "http://www.onvif.org/ver10/receiver/wsdl".
7. If *receiverServ.Capabilities* is specified, FAIL the test and skip other steps.
8. ONVIF Client invokes **GetServices** message with parameters:

- IncludeCapability := true
9. The DUT responds with a **GetServicesResponse** message with parameters:
    - Service list =: *listOfServicesWithCapabilities*
  10. If *listOfServicesWithCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/receiver/wsd", FAIL the test and skip other steps.
  11. Set *receiverServ* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver10/receiver/wsd".
  12. If *receiverServ.Capabilities* is not specified, FAIL the test and skip other steps.
  13. If *receiverServ.Capabilities* does not contain valid Capabilities element for Receiver service from "http://www.onvif.org/ver10/receiver/wsd" namespace, FAIL the test and skip other steps.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.21 GET SERVICES – ACCESS CONTROL SERVICE

**Test Case ID:** DEVICE-1-1-24

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

**Feature Under Test:** GetServices

**WSDL Reference:** devicemgmt.wsd

**Test Purpose:** To verify that Access Control Service is received using GetServices request.

**Pre-Requisite:** Access Control Service is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := false
4. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithoutCapabilities*
5. If *listOfServicesWithoutCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/accesscontrol/wsd", FAIL the test and skip other steps.
6. Set *accessControlServ* := item from *listOfServicesWithoutCapabilities* list with Namespace = "http://www.onvif.org/ver10/accesscontrol/wsd".
7. If *accessControlServ.Capabilities* is specified, FAIL the test and skip other steps.
8. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := true
9. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithCapabilities*
10. If *listOfServicesWithCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/accesscontrol/wsd", FAIL the test and skip other steps.
11. Set *accessControlServ* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver10/accesscontrol/wsd".
12. If *accessControlServ.Capabilities* is not specified, FAIL the test and skip other steps.
13. If *accessControlServ.Capabilities* does not contain valid Capabilities element for Access Control service from "http://www.onvif.org/ver10/accesscontrol/wsd" namespace, FAIL the test and skip other steps.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.22 GET SERVICES – DOOR CONTROL SERVICE

**Test Case ID:** DEVICE-1-1-25

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

**Feature Under Test:** GetServices

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify getting Door Control Service using GetServices request.

**Pre-Requirement:** Door Control Service is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := false
4. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithoutCapabilities*
5. If *listOfServicesWithoutCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/doorcontrol/wsdl", FAIL the test and skip other steps.
6. Set *doorControlServ* := item from *listOfServicesWithoutCapabilities* list with Namespace = "http://www.onvif.org/ver10/doorcontrol/wsdl".
7. If *doorControlServ.Capabilities* is specified, FAIL the test and skip other steps.
8. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := true
9. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithCapabilities*

10. If *listOfServicesWithCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/doorcontrol/wsd", FAIL the test and skip other steps.
11. Set *doorControlServ* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver10/doorcontrol/wsd".
12. If *doorControlServ.Capabilities* is not specified, FAIL the test and skip other steps.
13. If *doorControlServ.Capabilities* does not contain valid Capabilities element for Door Control service from "http://www.onvif.org/ver10/doorcontrol/wsd" namespace, FAIL the test and skip other steps.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.23 GET SERVICES – ADVANCED SECURITY SERVICE

**Test Case ID:** DEVICE-1-1-26

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

**Feature Under Test:** GetServices

**WSDL Reference:** devicemgmt.wsd

**Test Purpose:** To verify getting Advanced Security Service using GetServices request.

**Pre-Requisite:** Advanced Security Service is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := false

4. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithoutCapabilities*
5. If *listOfServicesWithoutCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/advancedsecurity/wsd", FAIL the test and skip other steps.
6. Set *advancedSecurityServ* := item from *listOfServicesWithoutCapabilities* list with Namespace = "http://www.onvif.org/ver10/advancedsecurity/wsd".
7. If *advancedSecurityServ.Capabilities* is specified, FAIL the test and skip other steps.
8. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := true
9. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithCapabilities*
10. If *listOfServicesWithCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/advancedsecurity/wsd", FAIL the test and skip other steps.
11. Set *advancedSecurityServ* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver10/advancedsecurity/wsd".
12. If *advancedSecurityServ.Capabilities* is not specified, FAIL the test and skip other steps.
13. If *advancedSecurityServ.Capabilities* does not contain valid Capabilities element for Advanced Security service from "http://www.onvif.org/ver10/advancedsecurity/wsd" namespace, FAIL the test and skip other steps.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.24 GET SERVICES – ACCESS RULES SERVICE

**Test Case ID:** DEVICE-1-1-27**Specification Coverage:** Capability exchange (ONVIF Core Specification)

**Feature Under Test:** GetServices

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify getting Access Rules Service using GetServices request.

**Pre-Requirement:** Access Rules Service is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := false
4. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithoutCapabilities*
5. If *listOfServicesWithoutCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/accessrules/wsdl", FAIL the test and skip other steps.
6. Set *accessRulesServ* := item from *listOfServicesWithoutCapabilities* list with Namespace = "http://www.onvif.org/ver10/accessrules/wsdl".
7. If *accessRulesServ.Capabilities* is specified, FAIL the test and skip other steps.
8. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := true
9. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithCapabilities*
10. If *listOfServicesWithCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/accessrules/wsdl", FAIL the test and skip other steps.
11. Set *accessRulesServ* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver10/accessrules/wsdl".
12. If *accessRulesServ.Capabilities* is not specified, FAIL the test and skip other steps.



13. If *accessRulesServ.Capabilities* does not contain valid Capabilities element for Access Rules service from "http://www.onvif.org/ver10/accessrules/wsdll" namespace, FAIL the test and skip other steps.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.25 GET SERVICES – CREDENTIAL SERVICE

**Test Case ID:** DEVICE-1-1-28

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

**Feature Under Test:** GetServices

**WSDL Reference:** devicemgmt.wsdll

**Test Purpose:** To verify getting Credential Service using GetServices request.

**Pre-Requisite:** Credential Service is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := false
4. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithoutCapabilities*
5. If *listOfServicesWithoutCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/credential/wsdll", FAIL the test and skip other steps.

6. Set *credential/Serv* := item from *listOfServicesWithoutCapabilities* list with Namespace = "http://www.onvif.org/ver10/credential/wsd".
7. If *credential/Serv.Capabilities* is specified, FAIL the test and skip other steps.
8. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := true
9. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithCapabilities*
10. If *listOfServicesWithCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/credential/wsd", FAIL the test and skip other steps.
11. Set *credential/Serv* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver10/credential/wsd".
12. If *credential/Serv.Capabilities* is not specified, FAIL the test and skip other steps.
13. If *credential/Serv.Capabilities* does not contain valid Capabilities element for Credential service from "http://www.onvif.org/ver10/credential/wsd" namespace, FAIL the test and skip other steps.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.26 GET SERVICES – SCHEDULE SERVICE

**Test Case ID:** DEVICE-1-1-29

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

**Feature Under Test:** GetServices

**WSDL Reference:** devicemgmt.wsd

**Test Purpose:** To verify getting Schedule Service using GetServices request.

**Pre-Requirement:** Schedule Service is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := false
4. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithoutCapabilities*
5. If *listOfServicesWithoutCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/schedule/wsd", FAIL the test and skip other steps.
6. Set *scheduleServ* := item from *listOfServicesWithoutCapabilities* list with Namespace = "http://www.onvif.org/ver10/schedule/wsd".
7. If *scheduleServ.Capabilities* is specified, FAIL the test and skip other steps.
8. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := true
9. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServicesWithCapabilities*
10. If *listOfServicesWithCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/schedule/wsd", FAIL the test and skip other steps.
11. Set *scheduleServ* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver10/schedule/wsd".
12. If *scheduleServ.Capabilities* is not specified, FAIL the test and skip other steps.
13. If *scheduleServ.Capabilities* does not contain valid Schedule element for Credential service from "http://www.onvif.org/ver10/schedule/wsd" namespace, FAIL the test and skip other steps.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.1.27 GET SERVICES AND GET CAPABILITIES CONSISTENCY

**Test Case ID:** DEVICE-1-1-30

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

**Feature Under Test:** GetServices, GetCapabilities

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** GetServices is supported by the DUT. GetCapabilities is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
  - IncludeCapability := true
4. The DUT responds with a **GetServicesResponse** message with parameters:
  - Service list =: *listOfServices*
5. ONVIF Client invokes **GetCapabilities** message with parameters:
  - Category[0] := All
6. The DUT responds with a **GetCapabilitiesResponse** message with parameters:
  - Capabilities =: *capabilities*

7. Set *deviceServ* := item from *listOfServices* list with Namespace = "http://www.onvif.org/ver10/device/wsdl".
8. Set *deviceServCapabilities* := *deviceServ.Capabilities.Capabilities*.
9. If *capabilities.Device.Network.IPFilter* is skipped or *capabilities.Device.Network.IPFilter* = false:
  - 9.1. If *deviceServCapabilities.Network.@IPFilter* = true, FAIL the test and skip other steps.
10. If *capabilities.Device.Network.IPFilter* = true:
  - 10.1. If *deviceServCapabilities.Network.@IPFilter* is skipped or *deviceServCapabilities.Network.@IPFilter* = false, FAIL the test and skip other steps.
11. If *capabilities.Device.Network.ZeroConfiguration* is skipped or *capabilities.Device.Network.ZeroConfiguration* = false:
  - 11.1. If *deviceServCapabilities.Network.@ZeroConfiguration* = true, FAIL the test and skip other steps.
12. If *capabilities.Device.Network.ZeroConfiguration* = true:
  - 12.1. If *deviceServCapabilities.Network.@ZeroConfiguration* is skipped or *deviceServCapabilities.Network.@ZeroConfiguration* = false, FAIL the test and skip other steps.
13. If *capabilities.Device.Network.IPVersion6* is skipped or *capabilities.Device.Network.IPVersion6* = false:
  - 13.1. If *deviceServCapabilities.Network.@IPVersion6* = true, FAIL the test and skip other steps.
14. If *capabilities.Device.Network.IPVersion6* = true:
  - 14.1. If *deviceServCapabilities.Network.@IPVersion6* is skipped or *deviceServCapabilities.Network.@IPVersion6* = false, FAIL the test and skip other steps.
15. If *capabilities.Device.Network.DynDNS* is skipped or *capabilities.Device.Network.DynDNS* = false:
  - 15.1. If *deviceServCapabilities.Network.@DynDNS* = true, FAIL the test and skip other steps.
16. If *capabilities.Device.Network.DynDNS* = true:

- 16.1. If `deviceServCapabilities.Network.@DynDNS` is skipped or `deviceServCapabilities.Network.@DynDNS = false`, FAIL the test and skip other steps.
17. If `capabilities.Device.Network.Extension.Dot11Configuration` is skipped or `capabilities.Device.Network.Extension.Dot11Configuration = false`:
- 17.1. If `deviceServCapabilities.Network.@Dot11Configuration = true`, FAIL the test and skip other steps.
18. If `capabilities.Device.Network.Extension.Dot11Configuration = true`:
- 18.1. If `deviceServCapabilities.Network.@Dot11Configuration` is skipped or `deviceServCapabilities.Network.@Dot11Configuration = false`, FAIL the test and skip other steps.
19. If `capabilities.Device.System.DiscoveryResolve` is skipped or `capabilities.Device.System.DiscoveryResolve = false`:
- 19.1. If `deviceServCapabilities.System.@DiscoveryResolve = true`, FAIL the test and skip other steps.
20. If `capabilities.Device.System.DiscoveryResolve = true`:
- 20.1. If `deviceServCapabilities.System.@DiscoveryResolve` is skipped or `deviceServCapabilities.System.@DiscoveryResolve = false`, FAIL the test and skip other steps.
21. If `capabilities.Device.System.DiscoveryBye` is skipped or `capabilities.Device.System.DiscoveryBye = false`:
- 21.1. If `deviceServCapabilities.System.@DiscoveryBye = true`, FAIL the test and skip other steps.
22. If `capabilities.Device.System.DiscoveryBye = true`:
- 22.1. If `deviceServCapabilities.System.@DiscoveryBye` is skipped or `deviceServCapabilities.System.@DiscoveryBye = false`, FAIL the test and skip other steps.
23. If `capabilities.Device.System.RemoteDiscovery` is skipped or `capabilities.Device.System.RemoteDiscovery = false`:
- 23.1. If `deviceServCapabilities.System.@RemoteDiscovery = true`, FAIL the test and skip other steps.
24. If `capabilities.Device.System.RemoteDiscovery = true`:

- 24.1. If `deviceServCapabilities.System.@RemoteDiscovery` is skipped or `deviceServCapabilities.System.@RemoteDiscovery = false`, FAIL the test and skip other steps.
25. If `capabilities.Device.System.SystemBackup` is skipped or `capabilities.Device.System.SystemBackup = false`:
- 25.1. If `deviceServCapabilities.System.@SystemBackup = true`, FAIL the test and skip other steps.
26. If `capabilities.Device.System.SystemBackup = true`:
- 26.1. If `deviceServCapabilities.System.@SystemBackup` is skipped or `deviceServCapabilities.System.@SystemBackup = false`, FAIL the test and skip other steps.
27. If `capabilities.Device.System.SystemLogging` is skipped or `capabilities.Device.System.SystemLogging = false`:
- 27.1. If `deviceServCapabilities.System.@SystemLogging = true`, FAIL the test and skip other steps.
28. If `capabilities.Device.System.SystemLogging = true`:
- 28.1. If `deviceServCapabilities.System.@SystemLogging` is skipped or `deviceServCapabilities.System.@SystemLogging = false`, FAIL the test and skip other steps.
29. If `capabilities.Device.System.FirmwareUpgrade` is skipped or `capabilities.Device.System.FirmwareUpgrade = false`:
- 29.1. If `deviceServCapabilities.System.@FirmwareUpgrade = true`, FAIL the test and skip other steps.
30. If `capabilities.Device.System.FirmwareUpgrade = true`:
- 30.1. If `deviceServCapabilities.System.@FirmwareUpgrade` is skipped or `deviceServCapabilities.System.@FirmwareUpgrade = false`, FAIL the test and skip other steps.
31. If `capabilities.Device.System.Extension.HttpFirmwareUpgrade` is skipped or `capabilities.Device.System.Extension.HttpFirmwareUpgrade = false`:
- 31.1. If `deviceServCapabilities.System.@HttpFirmwareUpgrade = true`, FAIL the test and skip other steps.



32. If *capabilities.Device.System.Extension.HttpFirmwareUpgrade* = true:

32.1. If *deviceServCapabilities.System.@HttpFirmwareUpgrade* is skipped or *deviceServCapabilities.System.@HttpFirmwareUpgrade* = false, FAIL the test and skip other steps.

33. If *capabilities.Device.System.Extension.HttpSystemBackup* is skipped or *capabilities.Device.System.Extension.HttpSystemBackup* = false:

33.1. If *deviceServCapabilities.System.@HttpSystemBackup* = true, FAIL the test and skip other steps.

34. If *capabilities.Device.System.Extension.HttpSystemBackup* = true:

34.1. If *deviceServCapabilities.System.@HttpSystemBackup* is skipped or *deviceServCapabilities.System.@HttpSystemBackup* = false, FAIL the test and skip other steps.

35. If *capabilities.Device.System.Extension.HttpSystemLogging* is skipped or *capabilities.Device.System.Extension.HttpSystemLogging* = false:

35.1. If *deviceServCapabilities.System.@HttpSystemLogging* = true, FAIL the test and skip other steps.

36. If *capabilities.Device.System.Extension.HttpSystemLogging* = true:

36.1. If *deviceServCapabilities.System.@HttpSystemLogging* is skipped or *deviceServCapabilities.System.@HttpSystemLogging* = false, FAIL the test and skip other steps.

37. If *capabilities.Device.System.Extension.HttpSupportInformation* is skipped or *capabilities.Device.System.Extension.HttpSupportInformation* = false:

37.1. If *deviceServCapabilities.System.@HttpSupportInformation* = true, FAIL the test and skip other steps.

38. If *capabilities.Device.System.Extension.HttpSupportInformation* = true:

38.1. If *deviceServCapabilities.System.@HttpSupportInformation* is skipped or *deviceServCapabilities.System.@HttpSupportInformation* = false, FAIL the test and skip other steps.

39. If *capabilities.Device.Security.TLS1.1* is skipped or *capabilities.Device.Security.TLS1.1* = false:

39.1. If *deviceServCapabilities.Security.@TLS1.1* = true, FAIL the test and skip other steps.



40. If *capabilities.Device.Security.TLS1.1* = true:

40.1. If *deviceServCapabilities.Security.@TLS1.1* is skipped or *deviceServCapabilities.Security.@TLS1.1* = false, FAIL the test and skip other steps.

41. If *capabilities.Device.Security.TLS1.2* is skipped or *capabilities.Device.Security.TLS1.2* = false:

41.1. If *deviceServCapabilities.Security.@TLS1.2* = true, FAIL the test and skip other steps.

42. If *capabilities.Device.Security.TLS1.2* = true:

42.1. If *deviceServCapabilities.Security.@TLS1.2* is skipped or *deviceServCapabilities.Security.@TLS1.2* = false, FAIL the test and skip other steps.

43. If *capabilities.Device.Security.OnboardKeyGeneration* is skipped or *capabilities.Device.Security.OnboardKeyGeneration* = false:

43.1. If *deviceServCapabilities.Security.@OnboardKeyGeneration* = true, FAIL the test and skip other steps.

44. If *capabilities.Device.Security.OnboardKeyGeneration* = true:

44.1. If *deviceServCapabilities.Security.@OnboardKeyGeneration* is skipped or *deviceServCapabilities.Security.@OnboardKeyGeneration* = false, FAIL the test and skip other steps.

45. If *capabilities.Device.Security.AccessPolicyConfig* is skipped or *capabilities.Device.Security.AccessPolicyConfig* = false:

45.1. If *deviceServCapabilities.Security.@AccessPolicyConfig* = true, FAIL the test and skip other steps.

46. If *capabilities.Device.Security.AccessPolicyConfig* = true:

46.1. If *deviceServCapabilities.Security.@AccessPolicyConfig* is skipped or *deviceServCapabilities.Security.@AccessPolicyConfig* = false, FAIL the test and skip other steps.

47. If *capabilities.Device.Security.X.509Token* is skipped or *capabilities.Device.Security.X.509Token* = false:

47.1. If *deviceServCapabilities.Security.@X.509Token* = true, FAIL the test and skip other steps.

48. If *capabilities.Device.Security.X.509Token* = true:

- 48.1. If `deviceServCapabilities.Security.@X.509Token` is skipped or `deviceServCapabilities.Security.@X.509Token = false`, FAIL the test and skip other steps.
49. If `capabilities.Device.Security.SAMLTOKEN` is skipped or `capabilities.Device.Security.SAMLTOKEN = false`:
- 49.1. If `deviceServCapabilities.Security.@SAMLTOKEN = true`, FAIL the test and skip other steps.
50. If `capabilities.Device.Security.SAMLTOKEN = true`:
- 50.1. If `deviceServCapabilities.Security.@SAMLTOKEN` is skipped or `deviceServCapabilities.Security.@SAMLTOKEN = false`, FAIL the test and skip other steps.
51. If `capabilities.Device.Security.KerberosToken` is skipped or `capabilities.Device.Security.KerberosToken = false`:
- 51.1. If `deviceServCapabilities.Security.@KerberosToken = true`, FAIL the test and skip other steps.
52. If `capabilities.Device.Security.KerberosToken = true`:
- 52.1. If `deviceServCapabilities.Security.@KerberosToken` is skipped or `deviceServCapabilities.Security.@KerberosToken = false`, FAIL the test and skip other steps.
53. If `capabilities.Device.Security.RELToken` is skipped or `capabilities.Device.Security.RELToken = false`:
- 53.1. If `deviceServCapabilities.Security.@RELToken = true`, FAIL the test and skip other steps.
54. If `capabilities.Device.Security.RELToken = true`:
- 54.1. If `deviceServCapabilities.Security.@RELToken` is skipped or `deviceServCapabilities.Security.@RELToken = false`, FAIL the test and skip other steps.
55. If `capabilities.Device.Security.Extension.TLS1.0` is skipped or `capabilities.Device.Security.Extension.TLS1.0 = false`:
- 55.1. If `deviceServCapabilities.Security.@TLS1.0 = true`, FAIL the test and skip other steps.
56. If `capabilities.Device.Security.Extension.TLS1.0 = true`:

- 56.1. If `deviceServCapabilities.Security.@TLS1.0` is skipped or `deviceServCapabilities.Security.@TLS1.0 = false`, FAIL the test and skip other steps.
57. If `capabilities.Device.Security.Extension.Extension.Dot1X` is skipped or `capabilities.Device.Security.Extension.Extension.Dot1X = false`:
- 57.1. If `deviceServCapabilities.Security.@Dot1X = true`, FAIL the test and skip other steps.
58. If `capabilities.Device.Security.Extension.Extension.Dot1X = true`:
- 58.1. If `deviceServCapabilities.Security.@Dot1X` is skipped or `deviceServCapabilities.Security.@Dot1X = false`, FAIL the test and skip other steps.
59. If `capabilities.Device.Security.Extension.Extension.SupportedEAPMethod` is skipped:
- 59.1. If `deviceServCapabilities.Security.@SupportedEAPMethods` is not empty, FAIL the test and skip other steps.
60. If `capabilities.Device.Security.Extension.Extension.SupportedEAPMethod` is not empty:
- 60.1. If `deviceServCapabilities.Security.@SupportedEAPMethods` is skipped or `deviceServCapabilities.Security.@SupportedEAPMethods` is empty, FAIL the test and skip other steps.
- 60.2. For each `supportedEAPMethod1` from `capabilities.Device.Security.Extension.Extension.SupportedEAPMethod` list:
- 60.2.1. If `deviceServCapabilities.Security.@SupportedEAPMethods` does not contain `supportedEAPMethod1`, FAIL the test and skip other steps.
61. If `capabilities.Device.Security.Extension.Extension.RemoteUserHandling` is skipped or `capabilities.Device.Security.Extension.Extension.RemoteUserHandling = false`:
- 61.1. If `deviceServCapabilities.Security.@RemoteUserHandling = true`, FAIL the test and skip other steps.
62. If `capabilities.Device.Security.Extension.Extension.RemoteUserHandling = true`:
- 62.1. If `deviceServCapabilities.Security.@RemoteUserHandling` is skipped or `deviceServCapabilities.Security.@RemoteUserHandling = false`, FAIL the test and skip other steps.
63. Set `eventsServ` := item from `listOfServices` list with Namespace = "http://www.onvif.org/ver10/events/wsdl".
64. Set `eventsServCapabilities` := `eventsServ.Capabilities.Capabilities`.

65. If *capabilities.Events.WSSubscriptionPolicySupport* = false:

65.1. If *eventsServCapabilities.@WSSubscriptionPolicySupport* = true, FAIL the test and skip other steps.

66. If *capabilities.Events.WSSubscriptionPolicySupport* = true:

66.1. If *eventsServCapabilities.@WSSubscriptionPolicySupport* is skipped or *eventsServCapabilities.@WSSubscriptionPolicySupport* = false, FAIL the test and skip other steps.

67. If *capabilities.Events.WSPullPointSupport* = false:

67.1. If *eventsServCapabilities.@WSPullPointSupport* = true, FAIL the test and skip other steps.

68. If *capabilities.Events.WSPullPointSupport* = true:

68.1. If *eventsServCapabilities.@WSPullPointSupport* is skipped or *eventsServCapabilities.@WSPullPointSupport* = false, FAIL the test and skip other steps.

69. If *capabilities.Events.WSPausableSubscriptionManagerInterfaceSupport* = false:

69.1. If *eventsServCapabilities.@WSPausableSubscriptionManagerInterfaceSupport* = true, FAIL the test and skip other steps.

70. If *capabilities.Events.WSPausableSubscriptionManagerInterfaceSupport* = true:

70.1. If *eventsServCapabilities.@WSPausableSubscriptionManagerInterfaceSupport* is skipped or *eventsServCapabilities.@WSPausableSubscriptionManagerInterfaceSupport* = false, FAIL the test and skip other steps.

71. If *capabilities* contains Analytics element:

71.1. If *listOfServices* does not contain item with Namespace = "http://www.onvif.org/ver20/analytics/wsdI", FAIL the test and skip other steps.

71.2. Set *analyticsServ* := item from *listOfServices* list with Namespace = "http://www.onvif.org/ver20/analytics/wsdI".

71.3. Set *analyticsServCapabilities* := *analyticsServ.Capabilities.Capabilities*.

71.4. If *capabilities.Analytics.RuleSupport* = false:

71.4.1. If *analyticsServCapabilities.@RuleSupport* = true, FAIL the test and skip other steps.

71.5. If *capabilities.Analytics.RuleSupport* = true:

71.5.1. If *analyticsServCapabilities.@RuleSupport* is skipped or *analyticsServCapabilities.@RuleSupport* = false, FAIL the test and skip other steps.

71.6. If *capabilities.Analytics.AnalyticsModuleSupport* = false:

71.6.1. If *analyticsServCapabilities.@AnalyticsModuleSupport* = true, FAIL the test and skip other steps.

71.7. If *capabilities.Analytics.AnalyticsModuleSupport* = true:

71.7.1. If *analyticsServCapabilities.@AnalyticsModuleSupport* is skipped or *analyticsServCapabilities.@AnalyticsModuleSupport* = false, FAIL the test and skip other steps.

72. If *capabilities* contains Imaging element:

72.1. If *listOfServices* does not contain item with Namespace = "http://www.onvif.org/ver20/imaging/wsdl", FAIL the test and skip other steps.

73. If *capabilities* contains Media element:

73.1. If *listOfServices* does not contain item with Namespace = "http://www.onvif.org/ver10/media/wsdl", FAIL the test and skip other steps.

73.2. Set *mediaServ* := item from *listOfServices* list with Namespace = "http://www.onvif.org/ver10/media/wsdl".

73.3. Set *mediaServCapabilities* := *mediaServ.Capabilities.Capabilities*.

73.4. If *capabilities.Media.StreamingCapabilities.RTPMulticast* is skipped or *capabilities.Media.StreamingCapabilities.RTPMulticast* = false:

73.4.1. If *mediaServCapabilities.StreamingCapabilities.@RTPMulticast* = true, FAIL the test and skip other steps.

73.5. If *capabilities.Media.StreamingCapabilities.RTPMulticast* = true:

73.5.1. If *mediaServCapabilities.StreamingCapabilities.@RTPMulticast* is skipped or *deviceServCapabilities.StreamingCapabilities.@RTPMulticast* = false, FAIL the test and skip other steps.

73.6. If *capabilities.Media.StreamingCapabilities.RTP\_TCP* is skipped or *capabilities.Media.StreamingCapabilities.RTP\_TCP* = false:

73.6.1. If *mediaServCapabilities.StreamingCapabilities.@RTP\_TCP* = true, FAIL the test and skip other steps.

73.7. If *capabilities.Media.StreamingCapabilities.RTP\_TCP* = true:

73.7.1. If *mediaServCapabilities.StreamingCapabilities.@RTP\_TCP* is skipped or *deviceServCapabilities.StreamingCapabilities.@RTP\_TCP* = false, FAIL the test and skip other steps.

73.8. If *capabilities.Media.StreamingCapabilities.RTP\_RTSP\_TCP* is skipped or *capabilities.Media.StreamingCapabilities.RTP\_RTSP\_TCP* = false:

73.8.1. If *mediaServCapabilities.StreamingCapabilities.@RTP\_RTSP\_TCP* = true, FAIL the test and skip other steps.

73.9. If *capabilities.Media.StreamingCapabilities.RTP\_RTSP\_TCP* = true:

73.9.1. If *mediaServCapabilities.StreamingCapabilities.@RTP\_RTSP\_TCP* is skipped or *deviceServCapabilities.StreamingCapabilities.@RTP\_RTSP\_TCP* = false, FAIL the test and skip other steps.

73.10. If *capabilities.Media.Extension.ProfileCapabilities.MaximumNumberOfProfiles* is not skipped:

73.10.1. If *mediaServCapabilities.ProfileCapabilities.@MaximumNumberOfProfiles* is skipped or *mediaServCapabilities.ProfileCapabilities.@MaximumNumberOfProfiles* != *capabilities.Media.Extension.ProfileCapabilities.MaximumNumberOfProfiles*, FAIL the test and skip other steps.

74. If *capabilities* contains PTZ element:

74.1. If *listOfServices* does not contain item with Namespace = "http://www.onvif.org/ver20/ptz/wsd", FAIL the test and skip other steps.

75. If *capabilities* contains Extension.DeviceIO element:

75.1. If *listOfServices* does not contain item with Namespace = "http://www.onvif.org/ver10/deviceIO/wsd", FAIL the test and skip other steps.

75.2. Set *deviceIOServ* := item from *listOfServices* list with Namespace = "http://www.onvif.org/ver10/deviceIO/wsd".



- 75.3. Set *deviceIOServCapabilities* := *deviceIOServ.Capabilities.Capabilities*.
- 75.4. If *capabilities.Extension.DeviceIO.VideoSources* = 0:
- 75.4.1. If *deviceIOServCapabilities.@VideoSources* > 0, FAIL the test and skip other steps.
- 75.5. If *capabilities.Extension.DeviceIO.VideoSources* > 0:
- 75.5.1. If *deviceIOServCapabilities.@VideoSources* is skipped or *deviceIOServCapabilities.@VideoSources* != *capabilities.Extension.DeviceIO.VideoSources*, FAIL the test and skip other steps.
- 75.6. If *capabilities.Extension.DeviceIO.VideoOutputs* = 0:
- 75.6.1. If *deviceIOServCapabilities.@VideoOutputs* > 0, FAIL the test and skip other steps.
- 75.7. If *capabilities.Extension.DeviceIO.VideoOutputs* > 0:
- 75.7.1. If *deviceIOServCapabilities.@VideoOutputs* is skipped or *deviceIOServCapabilities.@VideoOutputs* != *capabilities.Extension.DeviceIO.VideoOutputs*, FAIL the test and skip other steps.
- 75.8. If *capabilities.Extension.DeviceIO.AudioSources* = 0:
- 75.8.1. If *deviceIOServCapabilities.@AudioSources* > 0, FAIL the test and skip other steps.
- 75.9. If *capabilities.Extension.DeviceIO.AudioSources* > 0:
- 75.9.1. If *deviceIOServCapabilities.@AudioSources* is skipped or *deviceIOServCapabilities.@AudioSources* != *capabilities.Extension.DeviceIO.AudioSources*, FAIL the test and skip other steps.
- 75.10. If *capabilities.Extension.DeviceIO.AudioOutputs* = 0:
- 75.10.1. If *deviceIOServCapabilities.@AudioOutputs* > 0, FAIL the test and skip other steps.
- 75.11. If *capabilities.Extension.DeviceIO.AudioOutputs* > 0:

75.11.1.If *deviceIOServCapabilities.@AudioOutputs* is skipped or *deviceIOServCapabilities.@AudioOutputs* != *capabilities.Extension.DeviceIO.AudioOutputs*, FAIL the test and skip other steps.

75.12. If *capabilities.Extension.DeviceIO.RelayOutputs* = 0:

75.12.1.If *deviceIOServCapabilities.@RelayOutputs* > 0, FAIL the test and skip other steps.

75.13. If *capabilities.Extension.DeviceIO.RelayOutputs* > 0:

75.13.1.If *deviceIOServCapabilities.@RelayOutputs* is skipped or *deviceIOServCapabilities.@RelayOutputs* != *capabilities.Extension.DeviceIO.RelayOutputs*, FAIL the test and skip other steps.

76. If *capabilities* contains *Extension.Recording* element:

76.1. If *listOfServices* does not contain item with Namespace = "http://www.onvif.org/ver10/recording/wsd", FAIL the test and skip other steps.

76.2. Set *recordingServ* := item from *listOfServices* list with Namespace = "http://www.onvif.org/ver10/recording/wsd".

76.3. Set *recordingServCapabilities* := *recordingServ.Capabilities.Capabilities*.

76.4. If *capabilities.Extension.Recording.DynamicRecordings* = false:

76.4.1. If *recordingServCapabilities.@DynamicRecordings* = true, FAIL the test and skip other steps.

76.5. If *capabilities.Extension.Recording.DynamicRecordings* = true:

76.5.1. If *recordingServCapabilities.@DynamicRecordings* is skipped or *recordingServCapabilities.@DynamicRecordings* = false, FAIL the test and skip other steps.

76.6. If *capabilities.Extension.Recording.DynamicTracks* = false:

76.6.1. If *recordingServCapabilities.@DynamicTracks* = true, FAIL the test and skip other steps.

76.7. If *capabilities.Extension.Recording.DynamicTracks* = true:



76.7.1. If *recordingServCapabilities.@DynamicTracks* is skipped or *recordingServCapabilities.@DynamicTracks* = false, FAIL the test and skip other steps.

77. If *capabilities* contains Extension.Search element:

77.1. If *listOfServices* does not contain item with Namespace = "http://www.onvif.org/ver10/search/wsdl", FAIL the test and skip other steps.

77.2. Set *searchServ* := item from *listOfServices* list with Namespace = "http://www.onvif.org/ver10/search/wsdl".

77.3. Set *searchServCapabilities* := *searchServ.Capabilities.Capabilities*.

77.4. If *capabilities.Extension.Search.MetadataSearch* = false:

77.4.1. If *searchServCapabilities.@MetadataSearch* = true, FAIL the test and skip other steps.

77.5. If *capabilities.Extension.Search.MetadataSearch* = true:

77.5.1. If *searchServCapabilities.@MetadataSearch* is skipped or *searchServCapabilities.@MetadataSearch* = false, FAIL the test and skip other steps.

78. If *capabilities* contains Extension.Replay element:

78.1. If *listOfServices* does not contain item with Namespace = "http://www.onvif.org/ver10/replay/wsdl", FAIL the test and skip other steps.

79. If *capabilities* contains Extension.Receiver element:

79.1. If *listOfServices* does not contain item with Namespace = "http://www.onvif.org/ver10/receiver/wsdl", FAIL the test and skip other steps.

79.2. Set *receiverServ* := item from *listOfServices* list with Namespace = "http://www.onvif.org/ver10/receiver/wsdl".

79.3. Set *receiverServCapabilities* := *receiverServ.Capabilities.Capabilities*.

79.4. If *capabilities.Extension.Receiver.RTP\_Multicast* = false:

79.4.1. If *receiverServCapabilities.@RTP\_Multicast* = true, FAIL the test and skip other steps.

79.5. If *capabilities.Extension.Receiver.RTP\_Multicast* = true:

- 79.5.1. If *receiverServCapabilities.@RTP\_Multicast* is skipped or *receiverServCapabilities.@RTP\_Multicast* = false, FAIL the test and skip other steps.
- 79.6. If *capabilities.Extension.Receiver.RTP\_TCP* = false:
- 79.6.1. If *receiverServCapabilities.@RTP\_TCP* = true, FAIL the test and skip other steps.
- 79.7. If *capabilities.Extension.Receiver.RTP\_TCP* = true:
- 79.7.1. If *receiverServCapabilities.@RTP\_TCP* is skipped or *receiverServCapabilities.@RTP\_TCP* = false, FAIL the test and skip other steps.
- 79.8. If *capabilities.Extension.Receiver.RTP\_RTSP\_TCP* = false:
- 79.8.1. If *receiverServCapabilities.@RTP\_RTSP\_TCP* = true, FAIL the test and skip other steps.
- 79.9. If *capabilities.Extension.Receiver.RTP\_RTSP\_TCP* = true:
- 79.9.1. If *receiverServCapabilities.@RTP\_RTSP\_TCP* is skipped or *receiverServCapabilities.@RTP\_RTSP\_TCP* = false, FAIL the test and skip other steps.
- 79.10. If *capabilities.Extension.Receiver.SupportedReceivers* = 0:
- 79.10.1. If *receiverServCapabilities.@SupportedReceivers* > 0, FAIL the test and skip other steps.
- 79.11. If *capabilities.Extension.Receiver.SupportedReceivers* > 0:
- 79.11.1. If *receiverServCapabilities.@SupportedReceivers* != *capabilities.Extension.Receiver.SupportedReceivers*, FAIL the test and skip other steps.
- 79.12. If *capabilities.Extension.Receiver.MaximumRTSPURILength* = 0:
- 79.12.1. If *receiverServCapabilities.@MaximumRTSPURILength* > 0, FAIL the test and skip other steps.
- 79.13. If *capabilities.Extension.Receiver.MaximumRTSPURILength* > 0:

79.13.1.If *receiverServCapabilities.@MaximumRTSPURILength* is skipped or *receiverServCapabilities.@MaximumRTSPURILength* != *capabilities.Extension.Receiver.MaximumRTSPURILength*, FAIL the test and skip other steps.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetServicesResponse** message.
- DUT did not send **GetCapabilitiesResponse** message.

## 7.2 Network

### 7.2.1 NETWORK COMMAND HOSTNAME CONFIGURATION

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

**Specification Coverage:** Get hostname (ONVIF Core Specification)

**Feature Under Test:** GetHostname

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.2.2 NETWORK COMMAND SETHOSTNAME TEST ERROR CASE

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

**Specification Coverage:** Get system date and time (ONVIF Core Specification)

**Feature Under Test:** Set hostname

**WSDL Reference:** devicemgmt.wsdl

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

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetHostname** request to get the Hostname of the DUT.
4. The DUT returns configured currently Hostname settings.
5. ONVIF Client will invoke **SetHostname** request (Name = "Onvif\_test1") to configure the hostname.
6. The DUT returns **env:Sender/ter:InvalidArgVal/ter:InvalidHostname** SOAP 1.2 fault.
7. Verify hostname from DUT through **GetHostname** request.
8. The 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 a correct fault code in the SOAP 1.2 fault message (**env:Sender/ter:InvalidArgVal/ter:InvalidHostname**).
- The DUT did not send **GetHostnameResponse** message.
- The DUT returned "Onvif\_test1" as its Hostname.

**Note:** Hostname "Onvif\_test1" is just an example.

**Note:** See [Annex A.2](#) for Invalid Hostname.

**Note:** See [Annex A.4](#) for SOAP 1.2 fault message definitions.

## 7.2.3 GET DNS CONFIGURATION

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

**Specification Coverage:** Get DNS settings (ONVIF Core Specification)

**Feature Under Test:** GetDNS

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

**Test Result:****PASS –**

- 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.

## 7.2.4 SET DNS CONFIGURATION - SEARCHDOMAIN

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

**Specification Coverage:** Set DNS settings (ONVIF Core Specification)

**Feature Under Test:** SetDNS

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetDNS** request to retrieve the original settings of the DUT.
4. ONVIF Client will invoke **SetDNS** request (FromDHCP = false, SearchDomain = "domain.name").
5. Verify that the DUT sends **SetDNSResponse**.
6. Verify the DNS configurations at the DUT through **GetDNS** request.

7. The DUT sends its DNS configurations in the **GetDNSResponse** message (DNSInformation [FromDHCP = false, SearchDomain = "domain.name"]).
8. ONVIF Client will invoke **SetDNS** request to restore the original settings of the 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.

## 7.2.5 SET DNS CONFIGURATION - DNSMANUAL IPV4

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

**Specification Coverage:** Set DNS settings (ONVIF Core Specification)

**Feature Under Test:** GetSystemDateAndTime

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requirement:** DHCP is disabled (see [Annex A.13](#)).

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetDNS** request to retrieve the original settings of the DUT.
4. ONVIF Client will invoke **SetDNS** request (FromDHCP = false, DNSManual = IPv4, "10.1.1.1").
5. Verify that the DUT sends **SetDNSResponse**.

6. Verify the DNS configurations in DUT through **GetDNS** request.
7. The DUT sends its DNS configurations in the **GetDNSResponse** message (DNSInformation [FromDHCP = false, DNSManual = IPv4, "10.1.1.1"]).
8. ONVIF Client will invoke **SetDNS** request 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](#) valid IPv4 address definition.

**Note:** See [Annex A.10](#) for valid expression in terms of empty IP address.

## 7.2.6 SET DNS CONFIGURATION - DNSMANUAL IPV6

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

**Specification Coverage:** Set DNS settings (ONVIF Core Specification)

**Feature Under Test:** SetDNS

**WSDL Reference:** devicemgmt.wsdl

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

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetDNS** request to retrieve the original settings of the DUT.



4. ONVIF Client will invoke **SetDNS** request (FromDHCP = false, DNSManual = IPv6, "2001:1:1:1:1:1:1:1").
5. Verify that the DUT sends **SetDNSResponse**.
6. Verify the DNS configurations at the DUT through **GetDNS** request.
7. The DUT sends its DNS configurations in the **GetDNSResponse** message (DNSInformation [FromDHCP = false, DNSManual = IPv6, "2001:1:1:1:1:1:1:1"]).
8. ONVIF Client will invoke **SetDNS** request 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.

## 7.2.7 SET DNS CONFIGURATION - FROMDHCP

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

**Specification Coverage:** Set DNS settings (ONVIF Core Specification)

**Feature Under Test:** SetDNS

**WSDL Reference:** devicemgmt.wsdl

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

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

3. ONVIF Client will invoke **GetDNS** request to retrieve the original settings of the DUT.
4. ONVIF Client will invoke **SetDNS** request (FromDHCP = true).
5. Verify that the DUT sends **SetDNSResponse**.
6. Verify the DNS configurations in the DUT through **GetDNS** request.
7. The DUT sends its DNS configurations in the **GetDNSResponse** message (DNSInformation [FromDHCP = true, DNSFromDHCP = list of DNS Servers obtained from DHCP]).
8. ONVIF Client will invoke **SetDNS** request 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, the list can be empty if the DHCP server does not deliver DNS Addresses]) in the **GetDNSResponse** message.

## 7.2.8 SET DNS CONFIGURATION - DNSMANUAL INVALID IPV4

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

**Specification Coverage:** Set DNS settings (ONVIF Core Specification)

**Feature Under Test:** SetDNS

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** DHCP is disabled (see [Annex A.13](#)).

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **SetDNS** request (FromDHCP = false, DNSManual = IPv4, "10.1.1").
4. The DUT returns **env:Sender/ter:InvalidArgVal/ter:InvalidIPv4Address** SOAP 1.2 fault.
5. Retrieve DNS configurations from DUT through **GetDNS** request.
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 (**env:Sender/ter:InvalidArgVal/ter: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](#) valid IPv4 address definition.

**Note:** See [Annex A.10](#) for valid expression in terms of empty IP address.

## 7.2.9 SET DNS CONFIGURATION - DNSMANUAL INVALID IPV6

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

**Specification Coverage:** Set DNS settings (ONVIF Core Specification)

**Feature Under Test:** SetDNS**WSDL Reference:** devicemgmt.wsdl**Test Purpose:** To verify behavior of the DUT for invalid DNS IPv6 address configuration.**Pre-Requisite:** DHCP is disabled (see [Annex A.15](#)). IPv6 is implemented by the DUT.**Test Configuration:** ONVIF Client and DUT**Test Procedure:**

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

**Test Result:****PASS –**

- 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 (**env:Sender/ter:InvalidArgVal/ter: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.

## 7.2.10 GET NTP CONFIGURATION

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

**Specification Coverage:** Get NTP settings (ONVIF Core Specification)

**Feature Under Test:** GetNTP

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requirement:** NTP is supported by DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

**Test Result:**

**PASS –**

- 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.

## 7.2.11 SET NTP CONFIGURATION - NTPMANUAL IPV4

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

**Specification Coverage:** Set NTP settings (ONVIF Core Specification)

**Feature Under Test:** SetNTP

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** DHCP is disabled (see [Annex A.13](#)).NTP is supported by DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetNTP** request to retrieve the original settings of the DUT.
4. ONVIF Client will invoke **SetNTP** request (FromDHCP = false, NTPManual [Type = IPv4, IPv4Address = "10.1.1.1"]).
5. Verify that the DUT sends **SetNTPResponse**.
6. Verify the NTP Server settings in DUT through **GetNTP** request.
7. DUT sends its NTP Server settings in the **GetNTPResponse** message (NTPInformation [FromDHCP= false, NTPManual [Type = IPv4, IPv4Address = "10.1.1.1"]]).
8. ONVIF Client will invoke **SetNTP** request to restore the original settings of DUT.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **SetNTPResponse** message.
- The DUT did not send **GetNTPResponse** message.
- 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.

**Note:** See [Annex A.6](#) valid IPv4 address definition.

**Note:** See [Annex A.10](#) for valid expression in terms of empty IP address.

## 7.2.12 SET NTP CONFIGURATION - NTPMANUAL IPV6

**Test Case ID:** DEVICE-2-1-13

**Specification Coverage:** Set NTP settings (ONVIF Core Specification)

**Feature Under Test:** SetNTP

**WSDL Reference:** devicemgmt.wsdl

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

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

**Test Configuration:** ONVIF Client and DUT

### Test Procedure:

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

### Test Result:

#### PASS –

- DUT passes all assertions.

#### FAIL –

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

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

## 7.2.13 SET NTP CONFIGURATION - FROMDHCP

**Test Case ID:** DEVICE-2-1-14

**Specification Coverage:** Set NTP settings (ONVIF Core Specification)

**Feature Under Test:** SetNTP

**WSDL Reference:** devicemgmt.wsdl

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

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

**Test Configuration:** ONVIF Client and DUT

### Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetNTP** request to retrieve the original settings of the DUT.
4. ONVIF Client will invoke **SetNTP** request (FromDHCP = true).
5. Verify that the DUT sends **SetNTPResponse**.
6. Verify the NTP Server settings in DUT through **GetNTP** request.
7. The DUT sends its NTP Server settings in the **GetNTPResponse** message (NTPInformation [FromDHCP= true, NTPFromDHCP = list of NTP Servers obtained from DHCP]).
8. ONVIF Client will invoke **SetNTP** request to restore the original settings of DUT.

### Test Result:

#### PASS –

- DUT passes all assertions.

#### FAIL –



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

## 7.2.14 SET NTP CONFIGURATION - NTPMANUAL INVALID IPV4

**Test Case ID:** DEVICE-2-1-15

**Specification Coverage:** Set NTP settings (ONVIF Core Specification)

**Feature Under Test:** SetNTP

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** DHCP is disabled (see [Annex A.13](#)). NTP is supported by DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **SetNTP** request (FromDHCP = false, NTPManual [Type = IPv4, IPv4Address = "10.1.1"]).
4. The DUT returns **env:Sender/ter:InvalidArgVal/ter:InvalidIPv4Address** SOAP 1.2 fault.
5. Retrieve NTP Server configurations from DUT through **GetNTP** request.
6. The DUT sends valid NTP Server configurations in the **GetNTPResponse** message (NTPInformation [FromDHCP = true or false, NTPFromDHCP = list of NTP Servers obtained from DHCP, NTPManual = list of NTP Servers manually configured]).

**Test Result:**

**PASS –**

- 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 (**env:Sender/ter:InvalidArgVal/ter: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](#) valid IPv4 address definition.

## 7.2.15 SET NTP CONFIGURATION - NTPMANUAL INVALID IPV6

**Test Case ID:** DEVICE-2-1-16

**Specification Coverage:** Set NTP settings (ONVIF Core Specification)

**Feature Under Test:** SetNTP

**WSDL Reference:** devicemgmt.wsdl

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

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

**Test Configuration:** ONVIF Client and DUT

#### Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **SetNTP** request (FromDHCP = false, NTPManual [Type = IPv6, IPv6Address = "FF02:1"]).

4. The DUT returns **env:Sender/ter:InvalidArgVal/ter:InvalidIPv4Address** SOAP 1.2 fault.
5. Retrieve NTP Server configurations from DUT through **GetNTP** request.
6. The DUT sends valid NTP Server configurations in the **GetNTPResponse** message (NTPInformation [FromDHCP = true or false, NTPFromDHCP = list of NTP Servers obtained from DHCP, NTPManual = list of NTP Servers manually configured]).

**Test Result:****PASS –**

- 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 (**env:Sender/ter:InvalidArgVal/ter:InvalidIPv4Address**).
- 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.

## 7.2.16 GET NETWORK INTERFACE CONFIGURATION

**Test Case ID:** DEVICE-2-1-17

**Specification Coverage:** Get network interface configuration (ONVIF Core Specification)

**Feature Under Test:** GetNetworkInterfaces

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

## 7.2.17 SET NETWORK INTERFACE CONFIGURATION - IPV4

**Test Case ID:** DEVICE-2-1-18

**Specification Coverage:** Set network interface configuration (ONVIF Core Specification)

**Feature Under Test:** SetNetworkInterfaces

**WSDL Reference:** devicemgmt.wsdl

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

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetNetworkInterfaces** request.
4. The DUT responds with **GetNetworkInterfacesResponse** message with parameters

- NetworkInterfaces list =: *networkInterfacesList1*
5. Set *defaultNetworkSettings* := first interface from *networkInterfacesList1*.
  6. ONVIF Client invokes **SetNetworkInterfaces** request with parameters
    - InterfaceToken := *defaultNetworkSettings.@token*
    - NetworkInterface.Enabled skipped
    - NetworkInterface.Link skipped
    - NetworkInterface.MTU skipped
    - NetworkInterface.IPv4.Enabled := true
    - NetworkInterface.IPv4.Manual.Address := another address in the same subnet as an original address
    - NetworkInterface.IPv4.Manual.PrefixLength := current IPv4 prefix of the DUT
    - NetworkInterface.IPv4.DHCP := false
    - NetworkInterface.IPv6 skipped
    - NetworkInterface.Extension skipped
  7. The DUT responds with **SetNetworkInterfacesResponse** message with parameters
    - RebootNeeded =: *rebootNeeded*
  8. If *rebootNeeded* = true:
    - 8.1. ONVIF Client invokes **SystemReboot** request.
    - 8.2. The DUT responds with **SystemRebootResponse** message with parameters
      - Message
  9. The DUT sends **Hello** message from a newly configured address:
    - EndpointReference
    - Types =: *types*
    - Scopes
    - XAddrs =: *xAddrs*

- MetadataVersion
10. If *types* is skipped or empty, FAIL the test and go to step 20.
  11. If *xAddrs* is skipped or empty, FAIL the test and go to step 20.
  12. ONVIF Client invokes **GetNetworkInterfaces** request.
  13. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
    - NetworkInterfaces list =: *networkInterfacesList2*
  14. Set *updatedNetworkSettings* := item from *networkInterfacesList2* list with @token = *defaultNetworkSettings.@token*.
  15. If *updatedNetworkSettings.IPv4* is skipped or empty, FAIL the test and go to step 20.
  16. If *updatedNetworkSettings.IPv4.Enabled* = false, FAIL the test and go to step 20.
  17. If *updatedNetworkSettings.IPv4.Config.Manual* is skipped, FAIL the test and go to step 20.
  18. If *updatedNetworkSettings.IPv4.Config.Manual* is not equal to address at step 6, FAIL the test and go to step 20.
  19. If *updatedNetworkSettings.IPv4.Config.DHCP* = true, FAIL the test and go to step 20.
  20. ONVIF Client restores the original settings by following the procedure mentioned in [Annex A.8](#) with the following input and output parameters
    - in *defaultNetworkSettings* - Original default network settings

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetNetworkInterfacesResponse** message.
- DUT did not send **SetNetworkInterfacesResponse** message.
- DUT did not send **SystemRebootResponse** message.
- DUT did not send **Hello** message during *rebootTimeout*.

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

## 7.2.18 SET NETWORK INTERFACE CONFIGURATION - IPV6

**Test Case ID:** DEVICE-2-1-19

**Specification Coverage:** Set network interface configuration (ONVIF Core Specification)

**Feature Under Test:** SetNetworkInterfaces

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetNetworkInterfaces** request.
4. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
  - NetworkInterfaces list := *networkInterfacesList1*
5. Set *defaultNetworkSettings* := interface from *networkInterfacesList1* list with specified IPv6.
6. ONVIF Client invokes **SetNetworkInterfaces** request with parameters
  - InterfaceToken := *defaultNetworkSettings.@token*
  - NetworkInterface.Enabled := true
  - NetworkInterface.Link skipped
  - NetworkInterface.MTU skipped
  - NetworkInterface.IPv4 skipped
  - NetworkInterface.IPv6.Enabled := true
  - NetworkInterface.IPv6.AcceptRouterAdvert := *defaultNetworkSettings.IPv6.Config.AcceptRouterAdvert*
  - NetworkInterface.IPv6.Manual.Address := "2001:1:1:1:1:1:1:1"

- NetworkInterface.IPv6.Manual.PrefixLength := 64
  - NetworkInterface.IPv6.DHCP := Off
  - NetworkInterface.Extension skipped
7. The DUT responds with **SetNetworkInterfacesResponse** message with parameters
- RebootNeeded =: *rebootNeeded*
8. If *rebootNeeded* = true:
- 8.1. ONVIF Client invokes **SystemReboot** request.
- 8.2. The DUT responds with **SystemRebootResponse** message with parameters
- Message
9. The DUT sends **Hello** message from a newly configured address:
- EndpointReference
  - Types =: *types*
  - Scopes
  - XAddrs =: *xAddrs*
  - MetadataVersion
10. If *types* is skipped or empty, FAIL the test and go to step 20.
11. If *xAddrs* is skipped or empty, FAIL the test and go to step 20.
12. ONVIF Client invokes **GetNetworkInterfaces** request.
13. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
- NetworkInterfaces list =: *networkInterfacesList2*
14. Set *updatedNetworkSettings* := item from *networkInterfacesList2* list with @token = *defaultNetworkSettings.@token*.
15. If *updatedNetworkSettings.IPv6* is skipped or empty, FAIL the test and go to step 20.
16. If *updatedNetworkSettings.IPv6.Enabled* = false, FAIL the test and go to step 20.
17. If *updatedNetworkSettings.IPv6.Config.Manual* is skipped, FAIL the test and go to step 20.



18. If *updatedNetworkSettings.IPv6.Config.Manual* is not equal to address at step 6, FAIL the test and go to step 20.

19. If *updatedNetworkSettings.IPv6.Config.DHCP* != Off, FAIL the test and go to step 20.

20. ONVIF Client restores the original settings by following the procedure mentioned in Annex A.8 with the following input and output parameters

- in *defaultNetworkSettings* - Original default network settings

#### Test Result:

##### PASS –

- DUT passes all assertions.

##### FAIL –

- DUT did not send **GetNetworkInterfacesResponse** message.
- DUT did not send **SetNetworkInterfacesResponse** message.
- DUT did not send **SystemRebootResponse** message.
- DUT did not send **Hello** message during *rebootTimeout*.

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

## 7.2.19 SET NETWORK INTERFACE CONFIGURATION - INVALID IPV4

**Test Case ID:** DEVICE-2-1-20

**Specification Coverage:** Set network interface configuration (ONVIF Core Specification)

**Feature Under Test:** SetNetworkInterfaces

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None.

**Test Configuration:** ONVIF Client and DUT

#### Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client invokes **GetNetworkInterfaces** request to retrieve the configured network interfaces of the device.
4. The DUT returns its network interface settings.
5. ONVIF Client will invoke **SetNetworkInterfaces** request (InterfaceToken = available network interface, NetworkInterface.IPv4.Enable = true, NetworkInterface.IPv4.Manual.Address = "10.1.1").
6. The DUT returns **env:Sender/ter:InvalidArgVal/ter:InvalidIPv4Address** SOAP 1.2 fault.
7. Retrieve network interface configurations from DUT through **GetNetworkInterfaces** request.
8. 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 (**env:Sender/ter:InvalidArgVal/ter: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](#) valid IPv4 address definition.

## 7.2.20 SET NETWORK INTERFACE CONFIGURATION - INVALID IPV6

**Test Case ID:** DEVICE-2-1-21

**Specification Coverage:** Set network interface configuration (ONVIF Core Specification)

**Feature Under Test:** SetNetworkInterfaces

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requirement:** None.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetNetworkInterfaces** request to get the network interface settings of the device.
4. The device returns its network interfaces.
5. ONVIF Client will invoke **SetNetworkInterfaces** request (InterfaceToken = available network interface, NetworkInterface.IPv6.Enable = true, NetworkInterface.IPv6.Manual.Address = "FF02:1").
6. The DUT returns **env:Sender/ter:InvalidArgVal/ter:InvalidIPv6Address** SOAP 1.2 fault.
7. Retrieve network interface configurations from DUT through **GetNetworkInterfaces** request.
8. The 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 (**env:Sender/ter:InvalidArgVal/ter: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.

## 7.2.21 GET NETWORK PROTOCOLS CONFIGURATION

**Test Case ID:** DEVICE-2-1-33

**Specification Coverage:** Get network protocols (ONVIF Core Specification)

**Feature Under Test:** GetNetworkProtocols

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

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

## 7.2.22 SET NETWORK PROTOCOLS CONFIGURATION

**Test Case ID:** DEVICE-2-1-34

**Specification Coverage:** Set network protocols (ONVIF Core Specification)

**Feature Under Test:** SetNetworkProtocols

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetNetworkProtocols** request to retrieve the original settings of the DUT.
4. If RTSP protocol is not supported, skip other steps and go to the next test.
5. ONVIF Client will invoke **SetNetworkProtocols** request (NetworkProtocols [Name = RTSP, Enabled = true, Port = 10554]).
6. Verify that the DUT sends **SetNetworkProtocolsResponse**.
7. Verify the network protocols settings in DUT through **GetNetworkProtocols** request.
8. The DUT sends its network protocols settings in the **GetNetworkProtocolsResponse** message (NetworkProtocols [Name = RTSP, Enabled = true, Port = 10554]).
9. ONVIF Client will invoke **SetNetworkProtocols** request (NetworkProtocols [Name = RTSP, Enabled = false, Port = 10554]).
10. Verify that the DUT sends **SetNetworkProtocolsResponse**.
11. Verify the network protocols settings in DUT through **GetNetworkProtocols** request.

12. DUT sends its network protocols settings in the **GetNetworkProtocolsResponse** message (NetworkProtocols [Name = RTSP, Enabled = false, Port = integer]).
13. ONVIF Client will invoke **SetNetworkProtocols** request 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 at step 8.
- The DUT did not send correct network protocols information (i.e. NetworkProtocols [Name = RTSP, Enabled = false, Port = integer]) in **GetNetworkProtocolsResponse** message at step 12.

## 7.2.23 SET NETWORK PROTOCOLS CONFIGURATION - UNSUPPORTED PROTOCOLS

**Test Case ID:** DEVICE-2-1-35

**Specification Coverage:** Set network protocols (ONVIF Core Specification)

**Feature Under Test:** SetNetworkProtocols

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** DUT does not support all protocols.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

3. ONVIF client will retrieve network protocol configurations from the DUT through **GetNetworkProtocols** request.
4. The DUT sends network protocol configurations in the **GetNetworkProtocolsResponse** message (NetworkProtocols = supported protocols).
5. If HTTPS is not on the list of supported protocols, select HTTPS for the test as Protocol1. Otherwise, if RTSP is not on the list of supported protocols, select RTSP for the test as Protocol1. Otherwise, skip other steps and go to the next test.
6. ONVIF Client will invoke **SetNetworkProtocols** request (NetworkProtocols [Name = Protocol1, Enabled = true, Port = 10554]).
7. The DUT returns **env:Sender/ter:InvalidArgVal/ter:ServiceNotSupported** SOAP 1.2 fault.
8. Retrieve network protocol configurations from DUT through **GetNetworkProtocols** request.
9. The 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 (**env:Sender/ter:InvalidArgVal/ter: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.

## 7.2.24 GET NETWORK DEFAULT GATEWAY CONFIGURATION

**Test Case ID:** DEVICE-2-1-25

**Specification Coverage:** Get default gateway (ONVIF Core Specification)

**Feature Under Test:** GetNetworkDefaultGateway

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

## 7.2.25 SET NETWORK DEFAULT GATEWAY CONFIGURATION - INVALID IPV4

**Test Case ID:** DEVICE-2-1-28

**Specification Coverage:** Set default gateway (ONVIF Core Specification)



**Feature Under Test:** SetNetworkDefaultGateway**WSDL Reference:** devicemgmt.wsdl**Test Purpose:** To verify behavior of DUT for invalid default gateway IPv4 address configuration.**Pre-Requirement:** None.**Test Configuration:** ONVIF Client and DUT**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **SetNetworkDefaultGateway** request (IPv4Address = "10.1.1").
4. The DUT returns **env:Sender/ter:InvalidArgVal/ter:InvalidGatewayAddress** SOAP 1.2 fault.
5. Retrieve default gateway configurations from DUT through **GetNetworkDefaultGateway** request.
6. The DUT sends valid default gateway configurations in the **GetNetworkDefaultGatewayResponse** message (IPv4Address = list of IPv4 default gateway address, IPv6Address = list of IPv6 default gateway addresses).

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send SOAP 1.2 fault message.
- The DUT did not send a correct fault code in the SOAP fault message (**env:Sender/ter:InvalidArgVal/ter: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](#) valid IPv4 address definition.

## 7.2.26 SET NETWORK DEFAULT GATEWAY CONFIGURATION - INVALID IPV6

**Test Case ID:** DEVICE-2-1-29

**Specification Coverage:** Set default gateway (ONVIF Core Specification)

**Feature Under Test:** SetNetworkDefaultGateway

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requirement:** IPv6 is implemented by DUT.

**Test Configuration:** ONVIF Client and DUT

### Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **SetNetworkDefaultGateway** request (IPv6Address = "FF02:1").
4. The DUT returns **env:Sender/ter:InvalidArgVal/ter:InvalidGatewayAddress** SOAP 1.2 fault.
5. Retrieve default gateway configurations from DUT through **GetNetworkDefaultGateway** request.
6. The DUT sends valid default gateway configurations in the **GetNetworkDefaultGatewayResponse** message (IPv4Address = list of IPv4 default gateway address, IPv6Address = list of IPv6 default gateway addresses).

### Test Result:

#### PASS –

- 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 (**env:Sender/ter:InvalidArgVal/ter: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.

## 7.2.27 SET NETWORK DEFAULT GATEWAY CONFIGURATION - IPV4

**Test Case ID:** DEVICE-2-1-30

**Specification Coverage:** Set default gateway (ONVIF Core Specification)

**Feature Under Test:** SetNetworkDefaultGateway

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will follow the procedure described in [Annex A.13](#) to turn off IPv4 DHCP and set manual IP configuration.
4. ONVIF Client will invoke **GetNetworkDefaultGateway** request to retrieve the original settings of DUT.
5. Verify that the DUT sends **GetNetworkDefaultGatewayResponse** (original settings).
6. ONVIF Client will invoke **SetNetworkDefaultGateway** request (IPv4Address = "10.1.1.1").
7. Verify that the DUT sends **SetNetworkDefaultGatewayResponse**.

8. Verify the configured default gateways settings in DUT through **GetNetworkDefaultGateway** request.
9. DUT sends its configured default gateways settings in the **GetNetworkDefaultGatewayResponse** message (IPv4Address = "10.1.1.1").
10. ONVIF Client will invoke **SetNetworkDefaultGateway** request to restore the original network default gateway settings of DUT.
11. ONVIF Client will follow the procedure described in [Annex A.8](#) to turn on IPv4 DHCP to restore IP configuration.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **SetNetworkDefaultGatewayResponse** message.
- The DUT did not send **GetNetworkDefaultGatewayResponse** message.
- The DUT did not send correct default gateway information (i.e. IPv4Address = "10.1.1.1") in **GetNetworkDefaultGatewayResponse** message.

**Note:** See [Annex A.6](#) valid IPv4 address definition.

**Note:** See [Annex A.10](#) for valid expression in terms of empty IP address.

## 7.2.28 SET NETWORK DEFAULT GATEWAY CONFIGURATION - IPV6

**Test Case ID:** DEVICE-2-1-31

**Specification Coverage:** Set default gateway (ONVIF Core Specification)

**Feature Under Test:** SetNetworkDefaultGateway

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** IPv6 is implemented by the DUT.

**Test Configuration:** ONVIF Client and DUT**Test Procedure:**

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

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **SetNetworkDefaultGatewayResponse** message.
- The DUT did not send **GetNetworkDefaultGatewayResponse** message.
- The DUT did not send correct default gateway information (i.e. IPv6Address = "2001:1:1:1:1:1:1:1") in **GetNetworkDefaultGatewayResponse** message.

## 7.2.29 NETWORK COMMAND SETHOSTNAME TEST

**Test Case ID:** DEVICE-2-1-32

**Specification Coverage:** Set hostname (ONVIF Core Specification)

**Feature Under Test:** SetHostname

**WSDL Reference:** devicemgmt.wsdl

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

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

**Test Configuration:** ONVIF Client and DUT

### Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetHostname** request to retrieve original settings of the DUT.
4. ONVIF Client will invoke **SetHostname** request (Name = "Onvif-Test0-oNvif-Onv12", whose length is equal to 23 bytes) to configure the hostname.
5. Verify that the DUT sends **SetHostnameResponse**.
6. Verify the hostname configurations in DUT through **GetHostname** request.
7. The DUT sends hostname configuration in the **GetHostnameResponse** message (FromDHCP = false, Name = "Onvif-Test0-oNvif-Onv12").
8. ONVIF Client invokes **SetHostname** request to restore original settings of the 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 a correct hostname (i.e. "Onvif-Test0-oNvif-Onv123-Onvif12") in the **GetHostnameResponse** message.

**Note:** See [Annex A.2](#) for valid host names.

## 7.2.30 GET DYNAMIC DNS CONFIGURATION

**Test Case ID:** DEVICE-2-1-36

**Specification Coverage:** Get dynamic DNS settings (ONVIF Core Specification)

**Feature Under Test:** GetDynamicDNS

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To retrieve dynamic DNS settings of DUT through GetDynamicDNS command.

**Pre-Requisite:** Device supports DynamicDNS feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetDynamicDNS** request.
4. The DUT responds with a **GetDynamicDNSResponse** message with parameters
  - DynamicDNSInformation

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send a **GetDynamicDNSResponse**.

## 7.2.31 SET DYNAMIC DNS CONFIGURATION

**Test Case ID:** DEVICE-2-1-37

**Specification Coverage:** Set dynamic DNS settings (ONVIF Core Specification)

**Feature Under Test:** SetDynamicDNS

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To configure dynamic DNS settings of DUT through SetDynamicDNS command.

**Pre-Requisite:** Device supports DynamicDNS feature.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetDynamicDNS** request.
4. The DUT responds with a **GetDynamicDNSResponse** message with parameters
  - DynamicDNSInformation =: *initialDynamicDNSSettings*
5. ONVIF Client invokes **SetDynamicDNS** with parameters
  - Type := NoUpdate
  - Name skipped
  - TTL := PT20S
6. The DUT responds with a **SetDynamicDNSResponse** message.
7. ONVIF Client invokes **GetDynamicDNS** request.
8. The DUT responds with a **GetDynamicDNSResponse** message with parameters
  - DynamicDNSInformation =: *updatedDynamicDNSSettings*
9. If *updatedDynamicDNSSettings.Type* value is not equal to "NoUpdate", FAIL the test and skip other steps.
10. If *updatedDynamicDNSSettings.TTL* value is not equal to PT20S, FAIL the test and skip other steps.
11. ONVIF Client invokes **SetDynamicDNS** with parameters
  - Type := ServerUpdates
  - Name skipped



- TTL := PT25S
12. The DUT responds with a **SetDynamicDNSResponse** message.
  13. ONVIF Client invokes **GetDynamicDNS** request.
  14. The DUT responds with a **GetDynamicDNSResponse** message with parameters
    - DynamicDNSInformation =: *updatedDynamicDNSSettings*
  15. If *updatedDynamicDNSSettings.Type* value is not equal to "ServerUpdates", FAIL the test and skip other steps.
  16. If *updatedDynamicDNSSettings.TTL* value is not equal to PT25S, FAIL the test and skip other steps.
  17. ONVIF Client invokes **SetDynamicDNS** with parameters
    - Type := ClientUpdates
    - Name := "Onvif-Test0-oNvif-Onv12"
    - TTL := PT30S
  18. The DUT responds with a **SetDynamicDNSResponse** message.
  19. ONVIF Client invokes **GetDynamicDNS** request.
  20. The DUT responds with a **GetDynamicDNSResponse** message with parameters
    - DynamicDNSInformation =: *updatedDynamicDNSSettings*
  21. If *updatedDynamicDNSSettings.Type* value is not equal to "ClientUpdates", FAIL the test and skip other steps.
  22. If *updatedDynamicDNSSettings.Name* value is not equal to "Onvif-Test0-oNvif-Onv12", FAIL the test and skip other steps.
  23. If *updatedDynamicDNSSettings.TTL* value is not equal to PT30S, FAIL the test and skip other steps.
  24. ONVIF Client invokes **SetDynamicDNS** to restore dynamic DNS settings with parameters
    - Type := *initialDynamicDNSSettings.Type*
    - Name := *initialDynamicDNSSettings.Name*
    - TTL := *initialDynamicDNSSettings.TTL*

25. The DUT responds with a **SetDynamicDNSResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send a **GetDynamicDNSResponse** message.
- The DUT did not send a **SetDynamicDNSResponse** message.

## 7.3 System

### 7.3.1 SYSTEM COMMAND GETSYSTEMDATEANDTIME

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

**Specification Coverage:** Get system date and time (ONVIF Core Specification)

**Feature Under Test:** GetSystemDateAndTime

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

**Test Result:**

**PASS –**

- 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 shall return `UTCDateTime` or `LocalDateTime` in the `GetSystemDateAndTimeResponse` message.

## 7.3.2 SYSTEM COMMAND SETSYSTEMDATEANDTIME TEST FOR INVALID TIMEZONE

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

**Specification Coverage:** Set system date and time (ONVIF Core Specification)

**Feature Under Test:** `SetSystemDateAndTime`

**WSDL Reference:** `devicemgmt.wsdl`

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

**Pre-Requirement:** None.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetSystemDateAndTime** request.
4. The DUT responds with **GetSystemDateAndTimeResponse** message with parameters
  - `SystemDateAndTime` := *initialSystemDateAndTime*
5. ONVIF Client invokes **SetSystemDateAndTime** request with parameters
  - `DateTimeType` := `Manual`
  - `DaylightSavings` := `true`

- `TimeZone.TZ := "INVALIDTIMEZONE"`
  - `UTCDateTime.Time.Hour := current UTC hour`
  - `UTCDateTime.Time.Minute := current UTC minute`
  - `UTCDateTime.Time.Second := current UTC second`
  - `UTCDateTime.Date.Year := current UTC year`
  - `UTCDateTime.Date.Month := current UTC month`
  - `UTCDateTime.Date.Day := current UTC day`
6. The DUT returns **env:Sender/ter:InvalidArgVal/ter:InvalidTimeZone** SOAP 1.2 fault.
  7. ONVIF Client invokes **GetSystemDateAndTime** request.
  8. The DUT responds with **GetSystemDateAndTimeResponse** message with parameters
    - `SystemDateAndTime =: systemDateAndTime`
  9. If `systemDateAndTime.TimeZone.TZ` is specified and equal to "INVALIDTIMEZONE", FAIL the test and go to step 13.
  10. If `systemDateAndTime.UTCDateTime` is skipped, FAIL the test and go to step 13.
  11. If `systemDateAndTime.UTCDateTime` has invalid value for date or time, FAIL the test and go to step 13.
  12. If `systemDateAndTime.LocalDateTime` is specified and has invalid value for date or time, FAIL the test and go to step 13.
  13. ONVIF Client invokes **SetSystemDateAndTime** request with parameters
    - `DateTimeType := initialSystemDateAndTime.DateTimeType`
    - `DaylightSavings := initialSystemDateAndTime.DaylightSavings`
    - `TimeZone := initialSystemDateAndTime.TimeZone`
    - If `initialSystemDateAndTime.DateTimeType = Manual`:
      - `UTCDateTime.Time.Hour := current UTC hour`
      - `UTCDateTime.Time.Minute := current UTC minute`
      - `UTCDateTime.Time.Second := current UTC second`

- UTCDateTime.Date.Year := current UTC year
- UTCDateTime.Date.Month := current UTC month
- UTCDateTime.Date.Day := current UTC day

otherwise, UTCDateTime is skipped

14. DUT responds with **SetSystemDateAndTimeResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetSystemDateAndTimeResponse** message.
- DUT did not send the **env:Sender/ter:InvalidArgVal/ter:InvalidTimeZone** SOAP 1.2 fault message.
- DUT did not send **SetSystemDateAndTimeResponse** message.

**Note:** See [Annex A.3](#) for Invalid TimeZone.

**Note:** See [Annex A.4](#) for invalid SOAP 1.2 fault message definitions.

### 7.3.3 SYSTEM COMMAND SETSYSTEMDATEANDTIME TEST FOR INVALID DATE

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

**Specification Coverage:** Set system date and time (ONVIF Core Specification)

**Feature Under Test:** SetSystemDateAndTime

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetSystemDateAndTime** request.
4. The DUT responds with **GetSystemDateAndTimeResponse** message with parameters
  - SystemDateAndTime := *initialSystemDateAndTime*
5. ONVIF Client invokes **SetSystemDateAndTime** request with parameters
  - DateTimeType := Manual
  - DaylightSavings := true
  - TimeZone.TZ := "PST8PDT,M3.2.0,M11.1.0"
  - UTCDateTime := invalid date and time
6. The DUT returns **env:Sender/ter:InvalidArgVal/ter:InvalidDateTime** SOAP 1.2 fault.
7. ONVIF Client invokes **GetSystemDateAndTime** request.
8. The DUT responds with **GetSystemDateAndTimeResponse** message with parameters
  - SystemDateAndTime := *systemDateAndTime*
9. If *systemDateAndTime.TimeZone.TZ* is specified and *systemDateAndTime.TimeZone.TZ* has invalid value, FAIL the test and go to step 13.
10. If *systemDateAndTime.UTCDateTime* is skipped, FAIL the test and go to step 13.
11. If *systemDateAndTime.UTCDateTime* has invalid value for date or time, FAIL the test and go to step 13.
12. If *systemDateAndTime.LocalDateTime* is specified and has invalid value for date or time, FAIL the test and go to step 13.
13. ONVIF Client invokes **SetSystemDateAndTime** request with parameters
  - DateTimeType := *initialSystemDateAndTime.DateTimeType*
  - DaylightSavings := *initialSystemDateAndTime.DaylightSavings*
  - TimeZone := *initialSystemDateAndTime.TimeZone*
  - If *initialSystemDateAndTime.DateTimeType* = Manual:

- UTCDateTime.Time.Hour := current UTC hour
- UTCDateTime.Time.Minute := current UTC minute
- UTCDateTime.Time.Second := current UTC second
- UTCDateTime.Date.Year := current UTC year
- UTCDateTime.Date.Month := current UTC month
- UTCDateTime.Date.Day := current UTC day

otherwise, UTCDateTime is skipped

14. DUT responds with **SetSystemDateAndTimeResponse** message.

#### Test Result:

##### PASS –

- DUT passes all assertions.

##### FAIL –

- DUT did not send **GetSystemDateAndTimeResponse** message.
- DUT did not send the **env:Sender/ter:InvalidArgVal/ter:InvalidDateTime** SOAP 1.2 fault message.
- DUT did not send **SetSystemDateAndTimeResponse** message.

**Note:** See [Annex A.4](#) for invalid SOAP 1.2 fault message definitions.

## 7.3.4 SYSTEM COMMAND FACTORY DEFAULT HARD

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

**Specification Coverage:** Factory default (ONVIF Core Specification)

**Feature Under Test:** SetSystemFactoryDefault

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **SetSystemFactoryDefault** request (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 shall be configured with an IPv4 address, shall be IP reachable in the test network and other relevant configurations to be done for further tests.

## 7.3.5 SYSTEM COMMAND FACTORY DEFAULT SOFT

**Test Case ID:** DEVICE-3-1-7

**Specification Coverage:** Factory default (ONVIF Core Specification)

**Feature Under Test:** XML namespaces definition

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**



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

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **SetSystemFactoryDefaultResponse** message.
- The DUT did not send **ProbeMatch** message (i.e. DUT cannot be discovered).

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

## 7.3.6 SYSTEM COMMAND REBOOT

**Test Case ID:** DEVICE-3-1-8

**Specification Coverage:** Reboot

**Feature Under Test:** SystemReboot

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To reboot the DUT through SystemReboot command.

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

3. ONVIF Client will invoke **SystemReboot** request to reset the DUT.
4. Verify that the DUT sends **SystemRebootResponse** message (example message string = "Rebooting in x seconds").
5. The DUT will send Multicast **Hello** message after it is successfully rebooted.
6. ONVIF Client will wait for **Hello** message sent from newly configured address by the DUT. Then ONVIF Client will verify the **Hello** message and start using this newly configured address for further communications with DUT.
7. ONVIF Client will send Unicast **Probe** message to discover the DUT.
8. The DUT will send a **ProbeMatch** message.
9. ONVIF Client will verify the **ProbeMatch** message sent by the DUT.

**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 **ProbeMatch** message.

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

## 7.3.7 SYSTEM COMMAND DEVICE INFORMATION

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

**Specification Coverage:** Device Information

**Feature Under Test:** GetDeviceInformation

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT**Test Procedure:**

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

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.3.8 SYSTEM COMMAND GETSYSTEMLOG

**Test Case ID:** DEVICE-3-1-10**Specification Coverage:** Get system logs**Feature Under Test:** GetSystemLog**WSDL Reference:** devicemgmt.wsdl**Test Purpose:** To retrieve the DUT system and access logs using GetSystemLog command.**Pre-Requisite:** None**Test Configuration:** ONVIF Client and DUT**Test Procedure:**

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client invokes **GetSystemLog** request (System) to retrieve system log from the DUT.
4. Verify the **GetSystemLogResponse** message from the DUT or **env:Sender/ter:InvalidArgs/ter:SystemlogUnavailable** SOAP 1.2 fault, if system log is unavailable.
5. ONVIF Client invokes **GetSystemLog** request (Access) to retrieve system log from the DUT.
6. Verify the **GetSystemLogResponse** message from the DUT or **env:Sender/ter:InvalidArgs/ter:AccesslogUnavailable** SOAP 1.2 fault, if system 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.).

## 7.3.9 SYSTEM COMMAND SETSYSTEMDATEANDTIME

**Test Case ID:** DEVICE-3-1-11**Specification Coverage:** Set system date and time**Feature Under Test:** SetSystemDateAndTime**WSDL Reference:** devicemgmt.wsdl**Test Purpose:** To set the DUT system date and time using SetSystemDateAndTime command, date and time are entered manually.**Pre-Requisite:** None**Test Configuration:** ONVIF Client and DUT**Test Procedure:**

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

3. ONVIF Client invokes **GetSystemDateAndTime** request to retrieve original system date and type settings of DUT.
4. ONVIF Client will invoke **SetSystemDateAndTime** request (DateTimeType = Manual, DayLightSavings = false, UTCDateTime = Hour:Min:Sec, Year:Month:Day) to configure the date and time in the DUT.
5. Verify that DUT sends **SetSystemDateAndTimeResponse** message.
6. Verify the DUT date and time configurations through **GetSystemDateAndTime** request.
7. DUT sends system date and time configurations in the **GetSystemDateAndTimeResponse** message (DateTimeType = Manual, DayLightSavings = true, TimeZone = POSIX 1003.1, UTCDateTime = Hour:Min:Sec, Year:Month:Day).
8. ONVIF Client invokes **SetSystemDateAndTime** request (original DateTimeType Settings, original DayLightSavings, original TimeZone, current UTCDateTime for synchronize time if DateTimeType = Manual) to restore original system date and type settings of DUT.

**Test Result:****PASS –**

- 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.

## 7.3.10 SYSTEM COMMAND SETSYSTEMDATEANDTIME USING NTP

**Test Case ID:** DEVICE-3-1-12**Specification Coverage:** Set system date and time**Feature Under Test:** SetSystemDateAndTime**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client, DUT and NTP

**Test Procedure:**

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

## 7.3.11 GET SYSTEM URIS

**Test Case ID:** DEVICE-3-1-13

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

**Feature Under Test:** GetSystemUris

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify Get System Uris.

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. If the DUT supports GetServices command:
  - 3.1. ONVIF Client gets the service capabilities by following the procedure mentioned in [Annex A.18](#) with the following input and output parameters
    - out *cap1* - Device Management Service Capabilities
  - 3.2. Go to the step [5](#)

4. If the DUT does not support GetServices command:
  - 4.1. ONVIF Client gets the device service capabilities by following the procedure mentioned in [Annex A.19](#) with the following input and output parameters
    - out *cap2* - Device Management Capabilities
5. ONVIF client invokes **GetSystemUris**
6. The DUT responds with **GetSystemUrisResponse** message with parameters
  - SystemLogUri list =: *systemLogUriList*
  - SupportInfoUri =: *supportInfoUri*
  - SystemBackupUri =: *systemBackupUri*
  - Extension
7. If ((DUT supports GetServices command) and (*cap1.System.HttpSystemLogging* = true)) or ((DUT does not support GetServices command) and (*cap2.System.Extension.HttpSystemLogging* = true)):
  - 7.1. If *systemLogUriList* is empty, FAIL the test and skip other steps.
  - 7.2. For each SystemLog.Uri *uri* from *systemLogUriList* repeat the following steps:
    - 7.2.1. ONVIF client invokes **HTTP GET** to *uri*.
    - 7.2.2. The DUT responds with **HTTP 200 OK** response:
      - HTTP Body =: *systemLog*
    - 7.2.3. If *systemLog* is empty, FAIL the test and skip other steps.
8. If ((DUT supports GetServices command) and (*cap1.System.HttpSupportInformation* = true)) or ((DUT does not support GetServices command) and (*cap2.System.Extension.HttpSupportInformation* = true)):
  - 8.1. If *supportInfoUri* is empty, FAIL the test and skip other steps.
  - 8.2. ONVIF client invokes **HTTP GET** to *supportInfoUri*.
  - 8.3. The DUT responds with **HTTP 200 OK** response:
    - HTTP Body =: *supportInf*
  - 8.4. If *supportInf* is empty, FAIL the test and skip other steps.



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

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

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

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

- HTTP Body =: *systemBackup*

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

#### Test Result:

##### PASS –

- DUT passes all assertions.

##### FAIL –

- The DUT did not send **GetSystemUrisResponse** message.
- The DUT did not send **HTTP 200 OK** response.

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

## 7.3.12 START SYSTEM RESTORE

**Test Case ID:** DEVICE-3-1-14

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

**Feature Under Test:** StartSystemRestore

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify Start System Restore.

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF client invokes **GetSystemUris**.
4. The DUT responds with **GetSystemUrisResponse** message with parameters
  - SystemLogUris list
  - SupportInfoUri
  - SystemBackupUri =: *systemBackupUri*
  - Extension
5. If *systemBackupUri* is empty, FAIL the test and skip other steps.
6. ONVIF client invokes **HTTP GET** to *systemBackupUri*.
7. The DUT responds with **HTTP 200 OK** message with parameters
  - HTTP Body =: *systemBackup*
8. If *systemBackup* is empty, FAIL the test and skip other steps.
9. ONVIF client invokes **StartSystemRestore**.
10. The DUT responds with **StartSystemRestoreResponse** message with parameters
  - UploadUri =: *uploadUri*
  - ExpectedDownTime =: *expectedDownTime*
11. ONVIF client invokes **HTTP POST** to *uploadUri* with parameters
  - HTTP Header [Content-Type] := "application/octet-stream"
  - HTTP Body := *systemBackup*
12. The DUT responds with **HTTP 200 OK** message.
13. ONVIF Client waits *expectedDownTime* + *timeout1* for **Hello** message sent from newly configured address by the DUT. Then ONVIF Client starts using this newly configured address for further communications with DUT.
14. ONVIF Client invokes Unicast **Probe** message with the following parameters
  - Types empty

- Scopes empty

15. The DUT responds with **ProbeMatch** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetSystemUriResponse** message.
- The DUT did not send **StartSystemRestoreResponse** message.
- The DUT did not send **HTTP 200 OK** response.
- The DUT did not send **ProbeMatch** response.

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

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

## 7.3.13 START SYSTEM RESTORE – INVALID BACKUP FILE

**Test Case ID:** DEVICE-3-1-15

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

**Feature Under Test:** StartSystemRestore

**WSDL Reference:** devicemgmt.wsdl

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

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

**Test Procedure:**

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

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

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

**Note:** HTTP POST request could require authentication.

## 7.4 Security

### 7.4.1 SECURITY COMMAND GETUSERS

**Test Case ID:** DEVICE-4-1-1

**Specification Coverage:** Get users (ONVIF Core Specification)

**Feature Under Test:** GetUsers

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify the behavior of GetUsers command.

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetUsers** request, 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.

## 7.4.2 SECURITY COMMAND CREATEUSERS ERROR CASE

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

**Specification Coverage:** Create Users (ONVIF Core Specification)

**Feature Under Test:** CreateUsers

**WSDL Reference:** devicemgmt.wsdl

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

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

## Test Configuration: ONVIF Client and DUT

### Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **CreateUsers** request (Username = "OnvifTest1" Password = "OnvifTest123" UserLevel = Operator), to create the user in the DUT.
4. Verify that the DUT sends **CreateUsersResponse** message.
5. ONVIF Client will invoke **GetUsers** request, to retrieve the user list from the DUT.
6. Verify that the DUT sends **GetUsersResponse** message (Username = "OnvifTest1" UserLevel = Operator)
7. ONVIF Client will invoke **CreateUsers** request (Username = "OnvifTest1" Password = "OnvifTest123" UserLevel = User) to create the user in the DUT.
8. The DUT returns **env:Sender/ter:OperationProhibited/ter:UsernameClash** SOAP 1.2 fault.
9. ONVIF Client will invoke **CreateUsers** request (Username = "OnvifTest4" Password = "OnvifTest123" UserLevel = Operator, Username = "OnvifTest1" Password = "OnvifTest123" UserLevel = User) to create the users in the DUT.
10. The DUT returns **env:Sender/ter:OperationProhibited/ter:UsernameClash** SOAP 1.2 fault.
11. ONVIF Client will invoke **GetUsers** request, to retrieve the user list from the DUT.
12. Verify that DUT sends **GetUsersResponse** message (Username = "OnvifTest1" UserLevel = Operator)
13. ONVIF Client will invoke **DeleteUsers** request (Username = "OnvifTest1") to delete the user in the DUT.
14. Verify that the DUT sends **DeleteUsersResponse** 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 (**env:Sender/ter:OperationProhibited/ter:UsernameClash**).
- The DUT did not send **GetUsersResponse** message.
- The DUT did not send **CreateUsersResponse** message.
- The DUT did not send **DeleteUsersResponse** message.
- The DUT creates the user "OnvifTest1" with Level = User.
- The DUT creates the user "OnvifTest4".

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

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

### 7.4.3 SECURITY COMMAND DELETEUSERS

**Test Case ID:** DEVICE-4-1-4

**Specification Coverage:** Delete users (ONVIF Core Specification)

**Feature Under Test:** DeleteUsers

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify the behavior of DeleteUsers command.

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **CreateUsers** request (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.
5. ONVIF Client will invoke **GetUsers** request, to retrieve the user list from the DUT.
6. Verify that DUT sends **GetUsersResponse** message (Username = "OnvifTest1" UserLevel = Operator, Username = "OnvifTest2" UserLevel = Operator, Username = "OnvifTest3" UserLevel = User).
7. ONVIF Client will invoke **DeleteUsers** request (Username = "OnvifTest1"), to delete the user.
8. Verify that DUT sends **DeleteUsersResponse** message.
9. ONVIF Client will invoke **GetUsers** request, to retrieve the user list from the DUT.
10. Verify that DUT sends the updated user list (Username = "OnvifTest2" UserLevel = Operator, Username = "OnvifTest3" UserLevel = User).
11. ONVIF Client will invoke **DeleteUsers** request (Username = "OnvifTest2", Username = "OnvifTest3") to delete the users.
12. Verify that DUT sends the **DeleteUsersResponse** message.
13. ONVIF Client will invoke **GetUsers** request, 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 **DeleteUsersResponse** message.
- The DUT did not create the users.
- The DUT did not send **GetUsersResponse** message.
- The DUT did not send **CreateUsersResponse** message.
- The DUT did not delete the users.

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

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



## 7.4.4 SECURITY COMMAND DELETEUSERS ERROR CASE

**Test Case ID:** DEVICE-4-1-5

**Specification Coverage:** Delete users (ONVIF Core Specification)

**Feature Under Test:** DeleteUsers

**WSDL Reference:** devicemgmt.wsdl

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

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **CreateUsers** request (Username = "OnvifTest1" Password = "OnvifTest123" UserLevel = Operator), to create the user in the DUT.
4. Verify that DUT sends **CreateUsersResponse** message.
5. ONVIF Client will invoke **DeleteUsers** request (Username = "OnvifTest1", Username = "OnvifTest2"), to delete user.
6. The DUT returns **env:Sender/ter:InvalidArgVal/ter:UsernameMissing** SOAP 1.2 fault.
7. ONVIF Client will invoke **GetUsers** request, to retrieve the user list from the DUT.
8. Verify that DUT sends **GetUsersResponse** message (Username = "OnvifTest1" UserLevel = Operator).
9. ONVIF Client will invoke **DeleteUsers** request (Username = "OnvifTest1"), to delete the user.
10. Verify that DUT sends **DeleteUsersResponse** message.
11. ONVIF Client will invoke **GetUsers** request, 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 (**env:Sender/ter:InvalidArgVal/ter:UsernameMissing**).
- The DUT deletes the user "OnvifTest1" at step 5.
- The DUT did not send **DeleteUsersResponse** message.
- The DUT did not send **CreateUsersResponse** message.
- The DUT did not send **GetUsersResponse** message.

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

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

## 7.4.5 SECURITY COMMAND DELETEUSERS DELETE ALL USERS

**Test Case ID:** DEVICE-4-1-6

**Specification Coverage:** Delete users (ONVIF Core Specification)

**Feature Under Test:** DeleteUsers

**WSDL Reference:** devicemgmt.wsdl

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

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

3. ONVIF Client will invoke **GetUsers** request, to retrieve the user list from the DUT.
4. Verify that the DUT sends **GetUsersResponse** message (Returns default users, if any).
5. ONVIF Client will invoke **DeleteUsers** request (Username = Default Users), to delete the default users if there are any.
6. Verify that the DUT sends **DeleteUsersResponse** message or **env:Sender/ter:InvalidArgVal/ter:FixedUser** SOAP 1.2 fault message. Depending upon the implementation DUT may respond with SOAP 1.2 fault message or send empty response.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send correct SOAP 1.2 fault message (**env:Sender/ter:InvalidArgVal/ter:FixedUser**).
- The DUT did not send **GetUsersResponse** message.
- The DUT did not send **DeleteUsersResponse** message.

**Note:** The DUT may return the default users, if any.

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

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

## 7.4.6 SECURITY COMMAND SETUSER

**Test Case ID:** DEVICE-4-1-7

**Specification Coverage:** Set users (ONVIF Core Specification)

**Feature Under Test:** SetUser

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify the behavior of SetUser command.

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **CreateUsers** request (Username = "OnvifTest1" Password = "OnvifTest123" UserLevel = Operator, Username = "OnvifTest2" Password = "OnvifTest123" UserLevel = Operator), to create the user in the DUT.
4. Verify that the DUT sends **CreateUsersResponse** message.
5. ONVIF Client will invoke **GetUsers** request, to retrieve the user list from the DUT.
6. Verify that the DUT sends **GetUsersResponse** message (Username = "OnvifTest1" UserLevel = Operator, Username = "OnvifTest2" UserLevel = Operator).
7. ONVIF Client will invoke **SetUser** request (Username = "OnvifTest1" Password = "OnvifTest321" UserLevel = "Operator"), to update the user settings in the DUT.
8. Verify that the DUT sends **SetUserResponse** message.
9. ONVIF Client will invoke **GetUsers** request, to retrieve the user list from the DUT.
10. Verify that the DUT sends **GetUsersResponse** message (Username= "OnvifTest1" UserLevel = Operator, Username = "OnvifTest2" UserLevel = Operator).
11. ONVIF Client will invoke **SetUser** request (Username = "OnvifTest1" Password = "OnvifTest123" UserLevel = Operator, Username = "OnvifTest2" Password = "OnvifTest321" UserLevel = Operator), to update user settings in the DUT.
12. Verify that the DUT sends **SetUserResponse** message.
13. ONVIF Client will invoke **GetUsers** request, to retrieve the user list from the DUT.
14. Verify that the DUT sends **GetUsersResponse** message (Username = OnvifTest1 UserLevel = Operator, Username = OnvifTest2 UserLevel = User).
15. ONVIF Client will invoke **DeleteUsers** request (Username = "OnvifTest1", Username = "OnvifTest2") to delete the user in the DUT.
16. Verify that the DUT sends **DeleteUsersResponse** 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:** The DUT may return SOAP Fault 1.2 "TooManyUsers" for the CreateUsers command. In this case test shall be executed with the default users if any.

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

## 7.4.7 SECURITY COMMAND USER MANAGEMENT ERROR CASE

**Test Case ID:** DEVICE-4-1-8

**Specification Coverage:** Set users (ONVIF Core Specification)

**Feature Under Test:** SetUser

**WSDL Reference:** devicemgmt.wsdl

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

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **CreateUsers** request (Username = "OnvifTest1" Password = "OnvifTest123" UserLevel = Operator), to create the user in the DUT.
4. Verify that the DUT sends **CreateUsersResponse** message.

5. ONVIF Client will invoke **GetUsers** request to retrieve the user list from the DUT
6. Verify that the DUT sends the **GetUsersResponse** message (Username = "OnvifTest1" UserLevel = Operator).
7. ONVIF Client will invoke **SetUser** request (Username = "OnvifTest1" Password = "OnvifTest123" UserLevel = User, Username = "OnvifTest5" Password = "OnvifTest123" UserLevel = User) to update the user settings in the DUT.
8. The DUT returns **env:Sender/ter:InvalidArgVal/ter:UsernameMissing** SOAP 1.2 fault.
9. ONVIF Client will invoke **GetUsers** request, to retrieve the user list from the DUT.
10. Verify that the DUT sends the **GetUsersResponse** message (Username = "OnvifTest1" UserLevel = Operator).
11. ONVIF Client will invoke **DeleteUsers** request (Username = "OnvifTest1"), to delete the user in the DUT.
12. Verify that the DUT sends **DeleteUsersResponse** message.
13. ONVIF Client will invoke **GetUsers** request to retrieve the user list form the DUT
14. Verify that the DUT sends the **GetUsersResponse** message (returns default users, if any).

**Test Result:****PASS –**

- 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 (**env:Sender/ter:InvalidArgVal/ter:UsernameMissing**).
- The DUT did not send **GetUsersResponse** message.
- The DUT did not send **SetUserResponse** message.
- The DUT did not send **CreateUsersResponse** message.
- The DUT did not send **DeleteUsersResponse** message.
- The DUT updates the settings of the user "OnvifTest1".

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

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

## 7.4.8 SECURITY COMMAND CREATEUSERS

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

**Specification Coverage:** Create users (ONVIF Core Specification)

**Feature Under Test:** CreateUsers

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify the behavior of CreateUsers command.

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

**Test Configuration:** ONVIF Client and DUT

### Test Procedure:

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

11. ONVIF Client will invoke **GetUsers** request, to retrieve the user list from the DUT.
12. Verify that DUT sends the **GetUsersResponse** message (Username = "OnvifTest1" UserLevel = Operator, Username = "OnvifTest2" UserLevel = User).
13. ONVIF Client will invoke **DeleteUsers** request (Username = "OnvifTest1", Username = "OnvifTest2") to delete the users in the DUT.
14. Verify that DUT sends **DeleteUsersResponse** message.
15. ONVIF Client will invoke **CreateUsers** request (Username = "OnvifTest1" Password = "OnvifTest123" UserLevel = Administrator), to create the user in the DUT.
16. Verify that the DUT sends **CreateUsersResponse** message.
17. ONVIF Client will invoke **GetUsers** request to retrieve the user list from the DUT.
18. Verify that the DUT sends **GetUsersResponse** message (Username = "OnvifTest1", UserLevel = Administrator).
19. ONVIF Client will invoke **DeleteUsers** request (Username = "OnvifTest1") to delete the users in the DUT.
20. Verify that the DUT sends **DeleteUsersResponse** message.

**Test Result:****PASS –**

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

**FAIL –**

- The DUT did not send **GetUsersResponse** message.
- The DUT did not send **CreateUsersResponse** message.
- The DUT did not send **DeleteUsersResponse** message.

**Note:** The DUT may return SOAP Fault 1.2 "TooManyUsers" for the CreateUsers command if the capability MaxUsers is not present or equals the size of the users list. Such SOAP 1.2 fault message shall be treated as PASS case in this case. See [Annex A.17 TooManyUsers](#) fault check for details.

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



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

## 7.4.9 GET REMOTE USER

**Test Case ID:** DEVICE-4-1-10

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

**Feature Under Test:** GetRemoteUser

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify Get Remote User.

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.4.10 SET REMOTE USER

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

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

**Feature Under Test:** SetRemoteUser

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify Set Remote User.

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF client invokes **SetRemoteUser** request with parameters
  - RemoteUser.Username := "ONVIFRemoteUser"
  - RemoteUser.Password := "ONVIFRemotePassword"
  - RemoteUser.UseDerivedPassword := true
4. The DUT responds with **SetRemoteUserResponse** message.
5. ONVIF client invokes **GetRemoteUser** request.
6. The DUT responds with **GetRemoteUserResponse** message with parameters.
  - RemoteUser =: *remoteUser*
7. If *remoteUser* is empty, FAIL the test and skip other steps.
8. If *remoteUser*.User is not equal to "ONVIFRemoteUser", FAIL the test and skip other steps.
9. If *remoteUser*.UseDerivedPassword is not equal to true, FAIL the test and skip other steps.
10. If *remoteUser*.Password is not skkiped, FAIL the test and skip other steps.
11. ONVIF client invokes **SetRemoteUser** request with parameters
  - RemoteUser.Username := "ONVIFRemoteUser"
  - RemoteUser.Password := "ONVIFRemotePassword"
  - RemoteUser.UseDerivedPassword := false

12. The DUT responds with **SetRemoteUserResponse** message.
13. ONVIF client invokes **GetRemoteUser** request.
14. The DUT responds with **GetRemoteUserResponse** message with parameters
  - RemoteUser =: *remoteUser*
15. If *remoteUser* is empty, FAIL the test and skip other steps.
16. If *remoteUser*.User is not equal to "ONVIFRemoteUser", FAIL the test and skip other steps.
17. If *remoteUser*.UseDerivedPassword is not equal to false, FAIL the test and skip other steps.
18. If *remoteUser*.Password is not skkiped, FAIL the test and skip other steps.
19. ONVIF client invokes **SetRemoteUser** request with parameters
  - RemoteUser skipped
20. The DUT responds with **SetRemoteUserResponse** message.
21. ONVIF client invokes **GetRemoteUser** request.
22. The DUT responds with **GetRemoteUserResponse** message with parameters.
  - RemoteUser =: *remoteUser*
23. If *remoteUser* is not empty, FAIL the test.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.5 I/O

### 7.5.1 IO COMMAND GETRELAYOUTPUTS

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

**Specification Coverage:** Get relay outputs (ONVIF Core Specification)

**Feature Under Test:** GetRelayOutputs

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** Relay Outputs supported by DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetRelayOutputs** request 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.

## 7.5.2 RELAY OUTPUTS COUNT IN GETRELAYOUTPUTS AND GETCAPABILITIES

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

**Specification Coverage:** Get relay outputs, Capability exchange (ONVIF Core Specification)

**Feature Under Test:** GetRelayOutputs

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** Relay Outputs supported by DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetCapabilities** request (Device) to retrieve device capabilities.
4. Verify the **GetCapabilitiesResponse** message from the DUT.
5. ONVIF Client invokes **GetRelayOutputs** request 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, its number in **GetRelayOutputsResponse** message differs from the one in Device.IO.RelayOutputs from **GetCapabilitiesResponse** message.

## 7.5.3 IO COMMAND SETRELAYOUTPUTSETTINGS

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

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

**Feature Under Test:** GetRelayOutputs, SetRelayOutputSettings

**WSDL Reference:** devicemgmt.wsdl

**Test Purpose:** To verify the behavior of SetRelayOutputSettings command.

**Pre-Requirement:** Relay Outputs supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetRelayOutputs** request to retrieve a list of all available relay outputs and their settings.
4. The DUT sends the **GetRelayOutputsResponse** message with list of all available relay outputs and their settings.
5. Verify the **GetRelayOutputsResponse** message from the DUT.
6. If the DUT supports Bistable Mode with Open Idle state, ONVIF Client invokes **SetRelayOutputSettings** request (RelayOutputToken, Mode = Bistable, DelayTime = time1, IdleState = open) to set new relay output settings. If the DUT does not support Bistable Mode with Open Idle state, then go to the step 10.
7. Verify the **SetRelayOutputSettingsResponse** message.
8. ONVIF Client invokes **GetRelayOutputs** request to retrieve a list of all available relay outputs and their settings.
9. Verify **GetRelayOutputsResponse** message from the DUT. Check that values correspond the values from step 6 (except DelayTime value).
10. If the DUT supports Bistable Mode with Closed Idle state, ONVIF Client invokes **SetRelayOutputSettings** request (RelayOutputToken, Mode = Bistable, DelayTime = time1, IdleState = closed) to set new relay output settings. If the DUT does not support Bistable Mode with Closed Idle state, then go to the step 14.
11. Verify the **SetRelayOutputSettingsResponse** message.
12. ONVIF Client invokes **GetRelayOutputSettings** request to retrieve a list of all available relay outputs and their settings.

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

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetRelayOutputsResponse** message.
- The DUT did not send valid **GetRelayOutputsResponse** message.
- The DUT did not send **SetRelayOutputSettingsResponse** message.
- The DUT did not send valid **SetRelayOutputSettingsResponse** message.
- The DUT did not send correct changed settings in **GetRelayOutputsResponse** message.

- The DUT did not support at least one of the following: Bistable Mode with Open Idle state, Bistable Mode with Closed Idle state, Monostable Mode with Open Idle state, Monostable Mode with Closed Idle state

## 7.5.4 IO COMMAND SETRELAYOUTPUTSTATE – BISTABLE MODE (OPENED IDLE STATE)

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

**Specification Coverage:** Trigger relay output (ONVIF Core Specification)

**Feature Under Test:** SetRelayOutputState

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requirement:** Relay Outputs supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetRelayOutputs** request to retrieve a list of all available relay outputs and their settings.
4. The DUT sends the **GetRelayOutputsResponse** message with list of all available relay outputs and their settings.
5. ONVIF Client invokes **SetRelayOutputSettings** request (RelayOutputToken, Mode = Bistable, DelayTime = time1, IdleState = open).
6. The DUT sends the **SetRelayOutputSettingsResponse** message.
7. ONVIF Client invokes **SetRelayOutputState** request (RelayOutputToken, LogicalState = active)
8. Verify the **SetRelayOutputStateResponse** message from the DUT.
9. ONVIF Client invokes **SetRelayOutputState** request (RelayOutputToken, LogicalState = inactive).



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

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetRelayOutputsResponse** message.
- The DUT did not send **SetRelayOutputSettingsResponse** message.
- The DUT did not send **SetRelayOutputStateResponse** message.
- The DUT did not send a valid **SetRelayOutputStateResponse** message.

## 7.5.5 IO COMMAND SETRELAYOUTPUTSTATE – BISTABLE MODE (CLOSED IDLE STATE)

**Test Case ID:** DEVICE-5-1-6

**Specification Coverage:** Trigger relay output (ONVIF Core Specification)

**Feature Under Test:** SetRelayOutputState

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** Relay Outputs supported by DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetRelayOutputs** request to retrieve a list of all available relay outputs and their settings.
4. The DUT sends the **GetRelayOutputsResponse** message with list of all available relay outputs and their settings.

5. ONVIF Client invokes **SetRelayOutputSettings** request (RelayOutputToken, Mode = Bistable, DelayTime = time1, IdleState = closed).
6. The DUT sends the **SetRelayOutputSettingsResponse** message.
7. ONVIF Client invokes **SetRelayOutputState** request (RelayOutputToken, LogicalState = active).
8. Verify the **SetRelayOutputStateResponse** message from the DUT.
9. ONVIF Client invokes **SetRelayOutputState** request (RelayOutputToken, LogicalState = inactive).
10. Verify the **SetRelayOutputStateResponse** message from the DUT.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetRelayOutputsResponse** message.
- The DUT did not send **SetRelayOutputSettingsResponse** message.
- The DUT did not send valid **SetRelayOutputSettingsResponse** message.
- The DUT did not send **SetRelayOutputStateResponse** message.

## 7.5.6 IO COMMAND SETRELAYOUTPUTSTATE – MONOSTABLE MODE (OPENED IDLE STATE)

**Test Case ID:** DEVICE-5-1-7

**Specification Coverage:** Trigger relay output (ONVIF Core Specification)

**Feature Under Test:** SetRelayOutputState

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** Relay Outputs supported by DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetRelayOutputs** request to retrieve a list of all available relay outputs and their settings.
4. The DUT sends the **GetRelayOutputsResponse** message with list of all available relay outputs and their settings.
5. ONVIF Client invokes **SetRelayOutputSettings** request (RelayOutputToken, Mode = Monostable, DelayTime = time1, IdleState = open).
6. The DUT sends the **SetRelayOutputSettingsResponse** message.
7. ONVIF Client invokes **SetRelayOutputState** request (RelayOutputToken, LogicalState = active).
8. Verify the **SetRelayOutputStateResponse** message from the DUT. Wait until time out time1 expires.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetRelayOutputsResponse** message.
- The DUT did not send **SetRelayOutputSettingsResponse** message.
- The DUT did not send valid **SetRelayOutputSettingsResponse** message.
- The DUT did not send **SetRelayOutputStateResponse** message.

## 7.5.7 IO COMMAND SETRELAYOUTPUTSTATE – MONOSTABLE MODE (CLOSED IDLE STATE)

**Test Case ID:** DEVICE-5-1-8

**Specification Coverage:** Trigger relay output (ONVIF Core Specification)

**Feature Under Test:** SetRelayOutputState

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** Relay Outputs supported by DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetRelayOutputs** request to retrieve a list of all available relay outputs and their settings
4. The DUT sends the **GetRelayOutputsResponse** message with list of all available relay outputs and their settings.
5. ONVIF Client invokes **SetRelayOutputSettings** request (RelayOutputToken, Mode = Monostable, DelayTime = time1, IdleState = closed).
6. The DUT sends the **SetRelayOutputSettingsResponse** message.
7. ONVIF Client invokes **SetRelayOutputState** request (RelayOutputToken, LogicalState = active).
8. Verify the **SetRelayOutputStateResponse** message from the DUT.
9. Wait until time out time1 expires.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetRelayOutputsResponse** message.
- The DUT did not send **SetRelayOutputSettingsResponse** message.
- The DUT did not send valid **SetRelayOutputSettingsResponse** message.
- The DUT did not send **SetRelayOutputStateResponse** message.

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

**Test Case ID:** DEVICE-5-1-9

**Specification Coverage:** Trigger relay output (ONVIF Core Specification)

**Feature Under Test:** SetRelayOutputState

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requirement:** Relay Outputs are supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

### Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetRelayOutputs** request to retrieve a list of all available relay outputs and their settings.
4. The DUT sends the **GetRelayOutputsResponse** message with list of all available relay outputs and their settings.
5. If the DUT supports Monostable Mode with Open Idle state, ONVIF Client invokes **SetRelayOutputSettings** request (RelayOutputToken, Mode = Monostable, DelayTime = time1, IdleState = open) to set new relay output settings. If the DUT does not support Monostable Mode with Open Idle state, then go to the step 12.
6. The DUT sends the **SetRelayOutputSettingsResponse** message.
7. ONVIF Client invokes **SetRelayOutputState** request (RelayOutputToken, LogicalState = active).
8. Verify the **SetRelayOutputStateResponse** message from the DUT.
9. ONVIF Client invokes **SetRelayOutputState** request (RelayOutputToken, LogicalState = inactive), when time1 is not expired yet.
10. Verify the **SetRelayOutputStateResponse** message from the DUT.

11. Wait until timeout time1 expires.
12. If the DUT supports Monostable Mode with Closed Idle state, ONVIF Client invokes **SetRelayOutputSettings** request (RelayOutputToken, Mode = Monostable, DelayTime = time1, IdleState = closed) to set new relay output settings. If the DUT does not support Monostable Mode with Open Idle state, then skip other steps of the test.
13. The DUT sends the **SetRelayOutputSettingsResponse** message.
14. ONVIF Client invokes **SetRelayOutputState** request (RelayOutputToken, LogicalState = active).
15. Verify the **SetRelayOutputStateResponse** message from the DUT.
16. ONVIF Client invokes **SetRelayOutputState** request (RelayOutputToken, LogicalState = inactive), when time1 is not expired yet.
17. Verify the **SetRelayOutputStateResponse** message from the DUT.
18. Wait until time out time1 will expire.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetRelayOutputsResponse** message.
- The DUT did not send **SetRelayOutputSettingsResponse** message.
- The DUT did not send valid **SetRelayOutputSettingsResponse** message.
- The DUT did not send **SetRelayOutputStateResponse** message.
- The DUT did not change relay state as expected.

## 7.5.9 IO COMMAND SETRELAYOUTPUTSETTINGS – INVALID TOKEN

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

**Specification Coverage:** Set relay output settings (ONVIF Core Specification)

**Feature Under Test:** SetRelayOutputSettings

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** Relay Outputs supported by DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client sends a **GetRelayOutputs** request to retrieve the available relay outputs.
4. The DUT returns its relay outputs.
5. ONVIF Client invokes **SetRelayOutputSettings** request (invalid RelayOutputToken).
6. The DUT returns **env:Sender/ter:InvalidArgVal/ter:RelayToken** SOAP 1.2 fault.

**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 though the specified are preferable.

**Note:** See [Annex A.16](#) for Name and Token Parameters Length limitations.

## 7.5.10 IO COMMAND SETRELAYOUTPUTSTATE – INVALID TOKEN

**Test Case ID:** DEVICE-5-1-12

**Specification Coverage:** Trigger relay output (ONVIF Core Specification)

**Feature Under Test:** SetRelayOutputState

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** Relay Outputs supported by DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **SetRelayOutputState** request (invalid RelayOutputToken).
4. The DUT returns **env:Sender/ter:InvalidArgVal/ter:RelayToken** SOAP 1.2 fault.

**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 though the specified are preferable.

**Note:** See [Annex A.16](#) for Name and Token Parameters Length limitations.

## 7.6 Namespace Handling

### 7.6.1 DEVICE MANAGEMENT - NAMESPACES (DEFAULT NAMESPACES FOR EACH TAG)

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

**Specification Coverage:** None.

**Feature Under Test:** XML namespaces definition

**WSDL Reference:** devicemgmt.wsdl

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



**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetDNS** message to retrieve the original settings of the DUT.
4. Verify the **GetDNSResponse** message from the DUT.
5. ONVIF Client invokes **SetDNS** message (FromDHCP = false, DNSManual = ["IPv4", "DNS IP"]) to retrieve the original settings of the DUT.
6. Verify the **SetDNSResponse** message from the DUT.
7. ONVIF Client invokes **GetDNS** message to retrieve the settings of the DUT.
8. Verify the **GetDNSResponse** message from the DUT and verify DNS configurations in the DUT.
9. ONVIF Client will invoke **SetDNS** 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 the correct information in the **GetDNSResponse** message (i.e. FromDHCP = false, DNSManual = ["IPv4", "DNS IP"]).

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

## 7.6.2 DEVICE MANAGEMENT - NAMESPACES (DEFAULT NAMESPACES FOR PARENT TAG)

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

**Specification Coverage:** None.

**Feature Under Test:** XML namespaces definition

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetDNS** message to retrieve the original settings of the DUT.
4. Verify the **GetDNSResponse** message from the DUT.
5. ONVIF Client invokes **SetDNS** message (FromDHCP = false, DNSManual = ["IPv4", "DNS IP"]) to retrieve the original settings of the DUT.
6. Verify the **SetDNSResponse** message from the DUT.
7. ONVIF Client invokes **GetDNS** message to retrieve the settings of the DUT.
8. Verify the **GetDNSResponse** message from the DUT and verify DNS configurations in the DUT.
9. ONVIF Client will invoke **SetDNS** 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 the correct information in the **GetDNSResponse** message (i.e. FromDHCP = false, DNSManual = ["IPv4", "DNS IP"]).

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

## 7.6.3 DEVICE MANAGEMENT - NAMESPACES (NOT STANDARD PREFIXES)

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

**Specification Coverage:** None.

**Feature Under Test:** XML namespaces definition

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetDNS** message to retrieve the original settings of the DUT.
4. Verify the **GetDNSResponse** message from the DUT.
5. ONVIF Client invokes **SetDNS** message (FromDHCP = false, DNSManual = ["IPv4", "DNS IP"]) to retrieve the original settings of the DUT.
6. Verify the **SetDNSResponse** message from the DUT.
7. ONVIF Client invokes **GetDNS** message to retrieve the settings of the DUT.
8. Verify the **GetDNSResponse** message from the DUT and verify DNS configurations in the DUT.
9. ONVIF Client will invoke **SetDNS** 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 the correct information in the **GetDNSResponse** message (i.e. FromDHCP = false, DNSManual = ["IPv4", "DNS IP"]).

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

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

**Test Case ID:** DEVICE-6-1-4

**Specification Coverage:** None.

**Feature Under Test:** XML namespaces definition

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetDNS** message to retrieve the original settings of the DUT.
4. Verify the **GetDNSResponse** message from the DUT.
5. ONVIF Client invokes **SetDNS** message (FromDHCP = false, DNSManual = ["IPv4", "DNS IP"]) to retrieve the original settings of the DUT.
6. Verify the **SetDNSResponse** message from the DUT.

7. ONVIF Client invokes **GetDNS** message to retrieve the settings of the DUT.
8. Verify the **GetDNSResponse** message from the DUT and verify DNS configurations in the DUT.
9. ONVIF Client will invoke **SetDNS** 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 the correct information in the **GetDNSResponse** message (i.e. FromDHCP = false, DNSManual = ["IPv4", "DNS IP"]).

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

## 7.6.5 DEVICE MANAGEMENT - NAMESPACES (THE SAME PREFIX FOR DIFFERENT NAMESPACES)

**Test Case ID:** DEVICE-6-1-5

**Specification Coverage:** None.

**Feature Under Test:** XML namespaces definition

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

3. ONVIF Client invokes **GetDNS** message to retrieve the original settings of the DUT.
4. Verify the **GetDNSResponse** message from the DUT.
5. ONVIF Client invokes **SetDNS** message (FromDHCP = false, DNSManual = ["IPv4", "DNS IP"]) to retrieve the original settings of the DUT.
6. Verify the **SetDNSResponse** message from the DUT.
7. ONVIF Client invokes **GetDNS** message to retrieve the settings of the DUT.
8. Verify the **GetDNSResponse** message from the DUT and verify DNS configurations in the DUT.
9. ONVIF Client will invoke **SetDNS** 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 the correct information in the **GetDNSResponse** message (i.e. FromDHCP = false, DNSManual = ["IPv4", "DNS IP"]).

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

## 7.7 IP Filtering

### 7.7.1 GET IP ADDRESS FILTER

**Test Case ID:** DEVICE-7-1-1

**Specification Coverage:** Get IP address filter (ONVIF Core Specification)

**Feature Under Test:** GetIPAddressFilter

**WSDL Reference:** devicemgmt.wsdl

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

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetIPAddressFilter** request to retrieve IP address filter settings of the DUT.
4. Verify the **GetIPAddressFilterResponse** from the DUT (IPAddressFilter [Type = 'Allow' or 'Deny', IPv4Address = list of IPv4 filter addresses, IPv6Address = list of IPv6 filter addresses]).
5. If the DUT returned IPv4Address, validate IPv4Address.Address and IPv4Address.PrefixLength. Check that IPv4Address.Address represented at dot-decimal notation format. Check that  $0 \geq \text{IPv4Address.PrefixLength} \geq 32$ .
6. Check that IPv4Address.Address along with IPv4Address.PrefixLength is a valid range of IPv4 addresses according to Classless Inter-Domain Routing (CIDR) method.
7. If the DUT returned IPv6Address, validate IPv6Address.Address and IPv6Address.PrefixLength. Check that IPv6Address.Address represented as eight groups of four hexadecimal digits (or simplified). Check that  $0 \geq \text{IPv6Address.PrefixLength} \geq 128$ .
8. Check that IPv6Address.Address along with IPv6Address.PrefixLength is a valid range of IPv4 addresses according to Classless Inter-Domain Routing (CIDR) method.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetIPAddressFilterResponse** message.
- The DUT did not send correct information (i.e. IPAddressFilter [Type = Allow or Deny, IPv4Address = list of IPv4 filter addresses, IPv6Address = list of IPv6 filter addresses]) in the GetIPAddressFilterResponse message.
- The DUT did not send correct IPv4Address (IPv4Address.Address represented at dot-decimal notation format,  $0 \geq \text{IPv4Address.PrefixLength} \geq 32$ )

- The DUT did not send correct IPv4Address range.
- The DUT did not send correct IPv6Address (IPv6Address.Address represented as eight groups of four hexadecimal digits (or simplified,  $0 \geq \text{IPv6Address.PrefixLength} \geq 128$ ))
- The DUT did not send correct IPv6Address range.

**Note:** See [Annex A.10](#) for valid expression in terms of empty IP address.

## 7.7.2 SET IP ADDRESS FILTER – IPv4

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

**Specification Coverage:** Set IP address filter (ONVIF Core Specification)

**Feature Under Test:** SetIPAddressFilter

**WSDL Reference:** devicemgmt.wsdl

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

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetIPAddressFilter** request to retrieve the original settings of the DUT.
4. Verify **GetIPAddressFilterResponse** message from the DUT.
5. ONVIF Client will invoke **SetIPAddressFilter** request (IPAddressFilter [Type = Deny, IPv4Address = IPv4Address1]).
6. Verify that the DUT sends **SetIPAddressFilterResponse**
7. ONVIF Client will invoke **GetIPAddressFilter** request to retrieve updated settings of the DUT.
8. Verify **GetIPAddressFilterResponse** message from the DUT. Check that the settings were applied.



9. ONVIF Client will invoke **SetIPAddressFilter** request (IPAddressFilter [Type = Allow, IPv4Address = IPv4Address2]).
10. Verify that the DUT sends **SetIPAddressFilterResponse**.
11. ONVIF Client will invoke **GetIPAddressFilter** request to retrieve updated settings of DUT.
12. Verify **GetIPAddressFilterResponse** message from the DUT. Check that the settings were applied.
13. ONVIF Client restores the original settings of DUT.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetIPAddressFilterResponse** message.
- The DUT did not send **SetIPAddressFilterResponse** message.
- The DUT did not apply new settings after sending **SetIPAddressFilter** command.

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

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

### 7.7.3 ADD IP ADDRESS FILTER – IPv4

**Test Case ID:** DEVICE-7-1-3

**Specification Coverage:** Add IP address to filter (ONVIF Core Specification)

**Feature Under Test:** AddIPAddressFilter

**WSDL Reference:** devicemgmt.wsdl

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

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

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

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetIPAddressFilterResponse** message.
- The DUT did not send **AddIPAddressFilterResponse** message.
- The DUT did not return a new IP address at step 9.

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

## 7.7.4 REMOVE IP ADDRESS FILTER – IPv4

**Test Case ID:** DEVICE-7-1-4

**Specification Coverage:** Remove IP address from filter (ONVIF Core Specification)

**Feature Under Test:** RemoveIPAddressFilter

**WSDL Reference:** devicemgmt.wsdl

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

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

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetIPAddressFilter** request to retrieve IP address filter settings of the DUT.
4. Verify the **GetIPAddressFilterResponse** from the DUT (IPAddressFilter [Type = CurrentType (Allow or Deny), IPv4Address = list of IPv4 filter addresses, IPv6Address = list of IPv6 filter addresses]).
5. ONVIF Client will invoke **SetIPAddressFilter** request (IPAddressFilter [Type = Deny, IPv4Address = IPv4Address1]).
6. Verify that the DUT sends **SetIPAddressFilterResponse**.
7. ONVIF Client will invoke **GetIPAddressFilter** request to retrieve updated settings of the DUT.
8. Verify **GetIPAddressFilterResponse** message from the DUT. Check that the settings were applied.
9. ONVIF Client will invoke **RemoveIPAddressFilter** request (IPAddressFilter [Type = Allow, IPv4Address = IPv4Address1]) to remove IP filter address in the list.
10. Verify that the DUT sends **RemoveIPAddressFilterResponse**.
11. ONVIF Client will invoke **GetIPAddressFilter** request to retrieve updated IP address filter settings of the DUT.
12. Verify the **GetIPAddressFilterResponse** from the DUT (IPAddressFilter [Type = Deny, IPv4Address = empty list]).

13. ONVIF Client restores the original settings of the DUT.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **SetIPAddressFilterResponse** message.
- The DUT did not send **GetIPAddressFilterResponse** message.
- The DUT did not send **RemoveIPAddressFilterResponse** message.
- The DUT did not send an empty IPv4Address list at step 12.

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

## 7.8 Auxiliary Operation

### 7.8.1 AUXILIARY COMMANDS

**Test Case ID:** DEVICE-8-1-1

**Specification Coverage:** Auxiliary operation (ONVIF Core Specification)

**Feature Under Test:** SendAuxiliaryCommand, tt:Wiper, tt:Washer, tt:WashingProcedure, tt:IRLamp

**WSDL Reference:** devicemgmt.wsdl

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

**Pre-Requisite:** GetServices command is supported by the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServiceCapabilities** request.
4. The DUT responds with **GetServiceCapabilitiesResponse** message with parameters

- Capabilities =: *capabilities*
5. For each auxiliary command (*aux*) from the list {tt:Wiper|On, tt:Wiper|Off, tt:Washer|On, tt:Washer|Off, tt:WashingProcedure|On, tt:WashingProcedure|Off, tt:IRLamp|On, tt:IRLamp|Off, tt:IRLamp|Auto} which are supported according to *capabilities.Misc.AuxiliaryCommands* repeat the following steps:
    - 5.1. ONVIF Client invokes **SendAuxiliaryCommand** request with parameters
      - AuxiliaryCommand := *aux*
    - 5.2. The DUT responds with **SendAuxiliaryCommandResponse** message with parameters
      - AuxiliaryCommandResponse

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

## 7.9 Monitoring Events

### 7.9.1 Processor Usage event

**Test Case ID:** DEVICE-9-1-1

**Specification Coverage:** Monitoring Event ProcessorUsage

**Feature Under Test:** GetServices, GetEventProperties, CreatePullPointSubscription, PullMessages, Monitoring/ProcessorUsage event

**WSDL Reference:** devicemgmt.wsdl and event.wsdl

**Test Purpose:** To verify tns1:Monitoring/ProcessorUsage event generation after subscription and to verify tns1:Monitoring/ProcessorUsage event format.

**Pre-Requisite:** Event Service was received from the DUT. tns1:Monitoring/ProcessorUsage event is supported by the DUT as indicated by the GetEventPropertiesResponse.

**Test Configuration:** ONVIF Client and DUT

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
  - TopicNamespaceLocation list
  - FixedTopicSet
  - TopicSet =: *topicSet*
  - TopicExpressionDialect list
  - MessageContentFilterDialect list
  - MessageContentSchemaLocation list
5. If *topicSet* does not contain **tns1:Monitoring/ProcessorUsage** topic, FAIL the test and skip other steps.
6. ONVIF Client verifies **tns1:Monitoring/ProcessorUsage** topic (*processorUsageTopic*) from *topicSet*:
  - 6.1. If *processorUsageTopic.MessageDescription.IsProperty* is skipped or equals false, FAIL the test and skip other steps.
  - 6.2. If *processorUsageTopic* does not contain *MessageDescription.Source.SimpleItemDescription* item with Name = "Token", FAIL the test and skip other steps.
  - 6.3. If *processorUsageTopic.MessageDescription.Source.SimpleItemDescription* with Name = "Token" does not have Type = "tt:ReferenceToken", FAIL the test and skip other steps.
  - 6.4. If *processorUsageTopic* does not contain *MessageDescription.Data.SimpleItemDescription* item with Name = "Value", FAIL the test and skip other steps.
  - 6.5. If *processorUsageTopic.MessageDescription.Data.SimpleItemDescription* item with Name = "Value" does not have Type = "xs:float", FAIL the test and skip other steps.
7. ONVIF Client invokes **CreatePullPointSubscription** with parameters

- Filter.TopicExpression := "tns1:Monitoring/ProcessorUsage"
  - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
- SubscriptionReference =: s
  - CurrentTime
  - TerminationTime
9. Until *timeout1* timeout expires, repeat the following steps:
- 9.1. ONVIF Client invokes **PullMessages** to the subscription endpoint s with parameters
- Timeout := PT60S
  - MessageLimit := 1
- 9.2. The DUT responds with **PullMessagesResponse** message with parameters
- CurrentTime
  - TerminationTime
  - NotificationMessage =: m
- 9.3. If m is not null and m Message.Message.PropertyOperation = Initialized ONVIF Client verifies m:
- 9.3.1. If m.Topic does not equal to **tns1:Monitoring/ProcessorUsage**, FAIL the test and go to the step 10.
- 9.3.2. If m does not contain Message.Message.Source.SimpleItem.Token, FAIL the test and go to the step 10.
- 9.3.3. If m.Message.Message.Source.SimpleItem.Token has value type different from tt:ReferenceToken type, FAIL the test and go to the step 10.
- 9.3.4. If m does not contain Message.Message.Data.SimpleItem.Value, FAIL the test and go to the step 10.
- 9.3.5. If m.Message.Message.Data.SimpleItem.Value has value type different from xs:float type, FAIL the test and go to the step 10.

9.3.6. If *m.Message.Message.Data.SimpleItem.Value* value is outside of range 0 to 100, FAIL the test and go to the step 10.

9.3.7. Go to the step 10.

9.4. If *timeout1* timeout expires for step 9 without Notification with *PropertyOperation = Initialized*, FAIL the test and go to the step 10.

10. ONVIF Client invokes **Unsubscribe** to the subscription endpoint *s*.

11. The DUT responds with **UnsubscribeResponse** message.

#### Test Result:

##### PASS –

- DUT passes all assertions.

##### FAIL –

- The DUT did not send **GetEventPropertiesResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message(s).
- The DUT did not send **UnsubscribeResponse** message.

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

## 7.9.2 Last Reset event

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

**Specification Coverage:** Monitoring Event LastReset

**Feature Under Test:** GetServices, GetEventProperties, CreatePullPointSubscription, PullMessages, Monitoring/OperatingTime/LastReset event

**WSDL Reference:** devicemgmt.wsdl and event.wsdl

**Test Purpose:** To verify *tns1:Monitoring/OperatingTime/LastReset* event generation after subscription and to verify *tns1:Monitoring/OperatingTime/LastReset* event format.

**Pre-Requisite:** Event Service was received from the DUT. *tns1:Monitoring/OperatingTime/LastReset* event is supported by the DUT as indicated by the *GetEventPropertiesResponse*.



**Test Configuration:** ONVIF Client and DUT

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
  - TopicNamespaceLocation list
  - FixedTopicSet
  - TopicSet =: *topicSet*
  - TopicExpressionDialect list
  - MessageContentFilterDialect list
  - MessageContentSchemaLocation list
5. If *topicSet* does not contain **tns1:Monitoring/OperatingTime/LastReset** topic, FAIL the test and skip other steps.
6. ONVIF Client verifies **tns1:Monitoring/OperatingTime/LastReset** topic (*lastResetTopic*) from *topicSet*:
  - 6.1. If *lastResetTopic* MessageDescription.IsProperty is skipped or equals to false, FAIL the test and skip other steps.
  - 6.2. If *lastResetTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "Status", FAIL the test and skip other steps.
  - 6.3. If *lastResetTopic* MessageDescription.Data.SimpleItemDescription with Name = "Status" does not have Type = "xs:dateTime", FAIL the test and skip other steps.
7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
  - Filter.TopicExpression := "tns1:Monitoring/OperatingTime/LastReset"
  - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
  - SubscriptionReference =: s

- CurrentTime
  - TerminationTime
9. Until *timeout1* timeout expires, repeat the following steps:
- 9.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
- Timeout := PT60S
  - MessageLimit := 1
- 9.2. The DUT responds with **PullMessagesResponse** message with parameters
- CurrentTime
  - TerminationTime
  - NotificationMessage =: *m*
- 9.3. If *m* is not null and *m* Message.Message.PropertyOperation = Initialized ONVIF Client verifies *m*:
- 9.3.1. If *m* Topic does not equal to **tns1:Monitoring/OperatingTime/LastReset**, FAIL the test and go to the step 10.
- 9.3.2. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 10.
- 9.3.3. If *m* Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 10.
- 9.3.4. Go to the step 10.
- 9.4. If *timeout1* timeout expires for step 9 without Notification with PropertyOperation = Initialized, FAIL the test and go to the step 10.
10. ONVIF Client invokes **Unsubscribe** to the subscription endpoint *s*.
11. The DUT responds with **UnsubscribeResponse** message.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetEventPropertiesResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message(s).
- The DUT did not send **UnsubscribeResponse** message.

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

### 7.9.3 Last Reboot event

Test Label: Last Reboot event **Test Case ID:** DEVICE-9-1-3

**Specification Coverage:** Monitoring Event LastReboot

**Feature Under Test:** GetServices, GetEventProperties, CreatePullPointSubscription, PullMessages, Monitoring/OperatingTime/LastReboot event

**WSDL Reference:** devicemgmt.wsdl and event.wsdl

**Test Purpose:** To verify tns1:Monitoring/OperatingTime/LastReboot event generation after subscription and to verify tns1:Monitoring/OperatingTime/LastReboot event format.

**Pre-Requisite:** Event Service was received from the DUT. tns1:Monitoring/OperatingTime/LastReboot event is supported by the DUT as indicated by the GetEventPropertiesResponse.

**Test Configuration:** ONVIF Client and DUT

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
  - TopicNamespaceLocation list
  - FixedTopicSet
  - TopicSet =: *topicSet*
  - TopicExpressionDialect list
  - MessageContentFilterDialect list

- MessageContentSchemaLocation list
5. If *topicSet* does not contain **tns1:Monitoring/OperatingTime/LastReboot** topic, FAIL the test and skip other steps.
  6. ONVIF Client verifies **tns1:Monitoring/OperatingTime/LastReboot** topic (*lastRebootTopic*) from *topicSet*:
    - 6.1. If *lastRebootTopic.MessageDescription.IsProperty* is skipped or equals to false, FAIL the test and skip other steps.
    - 6.2. If *lastRebootTopic* does not contain *MessageDescription.Data.SimpleItemDescription* item with Name = "Status", FAIL the test and skip other steps.
    - 6.3. If *lastRebootTopic.MessageDescription.Data.SimpleItemDescription* with Name = "Status" does not have Type = "xs:dateTime", FAIL the test and skip other steps.
  7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
    - Filter.TopicExpression := "tns1:Monitoring/OperatingTime/LastReboot"
    - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
  8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
    - SubscriptionReference =: s
    - CurrentTime
    - TerminationTime
  9. Until *timeout1* timeout expires, repeat the following steps:
    - 9.1. ONVIF Client invokes **PullMessages** to the subscription endpoint s with parameters
      - Timeout := PT60S
      - MessageLimit := 1
    - 9.2. The DUT responds with **PullMessagesResponse** message with parameters
      - CurrentTime
      - TerminationTime

- NotificationMessage =: *m*
- 9.3. If *m* is not null and *m* Message.Message.PropertyOperation = Initialized ONVIF Client verifies *m*:
- 9.3.1. If *m* Topic does not equal to **tns1:Monitoring/OperatingTime/LastReboot**, FAIL the test and go to the step 10.
  - 9.3.2. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 10.
  - 9.3.3. If *m* Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 10.
  - 9.3.4. Go to the step 10.
- 9.4. If *timeout1* timeout expires for step 9 without Notification with PropertyOperation = Initialized, FAIL the test and go to the step 10.
10. ONVIF Client invokes **Unsubscribe** to the subscription endpoint *s*.
11. The DUT responds with **UnsubscribeResponse** message.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetEventPropertiesResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message(s).
- The DUT did not send **UnsubscribeResponse** message.

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

## 7.9.4 Last Reboot event (Status change)

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

**Specification Coverage:** Monitoring Event LastReboot

**Feature Under Test:** CreatePullPointSubscription, PullMessages, SystemReboot, Monitoring/OperatingTime/LastReboot event

**WSDL Reference:** devicemgmt.wsdl and event.wsdl

**Test Purpose:** To verify that the last reboot event signals correct values.

**Pre-Requirement:** Event Service was received from the DUT. tns1:Monitoring/OperatingTime/LastReboot event is supported by the DUT as indicated by the GetEventPropertiesResponse. NTP server in network and configured on the DUT.

**Test Configuration:** ONVIF Client and DUT

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **CreatePullPointSubscription** with parameters
  - Filter.TopicExpression := "tns1:Monitoring/OperatingTime/LastReboot"
  - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
4. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
  - SubscriptionReference =: s
  - CurrentTime
  - TerminationTime
5. Until *timeout1* timeout expires, repeat the following steps:
  - 5.1. ONVIF Client invokes **PullMessages** to the subscription endpoint s with parameters
    - Timeout := PT60S
    - MessageLimit := 1
  - 5.2. The DUT responds with **PullMessagesResponse** message with parameters
    - CurrentTime
    - TerminationTime
    - NotificationMessage =: m

- 5.3. If *m* is not null and *m* Message.Message.PropertyOperation = Initialized:
  - 5.3.1. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 6.
  - 5.3.2. If *m* Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 17.
  - 5.3.3. Set *lastRebootTime* := *m* Message.Message.Data.SimpleItem.Status
  - 5.3.4. Go to the step 6.
- 5.4. If *timeout1* timeout expires for step 5 without Notification with PropertyOperation = Initialized, FAIL the test and go to the step 17.
6. ONVIF Client invokes **Unsubscribe** to the subscription endpoint *s*. The test should not FAIL even if Unsubscribe fails.
7. The DUT responds with **UnsubscribeResponse** message. The test should not FAIL even if **UnsubscribeResponse** message is not received.
8. ONVIF Client invokes **SystemReboot**.
9. The DUT responds with **SystemRebootResponse** message with parameters
  - Message
10. Until *timeout2* timeout expires repeat the following steps:
  - 10.1. The DUT will send Multicast **Hello** message after it is successfully rebooted with parameters:
    - EndpointReference.Address equal to unique endpoint reference of the DUT
    - Types list
    - Scopes list
    - XAddrs list := *xaddrsList*
    - MetadataVersion
  - 10.2. If *xaddrsList* contains URI address with not a LinkLocal IPv4 address from ONVIF Client subnet, go to step 12.
11. If *timeout2* timeout expires for step 10 without **Hello** with URI address with not a LinkLocal IPv4 address from ONVIF Client subnet, FAIL the test and skip other steps.

12. ONVIF client waits for 5 seconds after **Hello** was received.
13. ONVIF Client invokes **CreatePullPointSubscription** with parameters
  - Filter.TopicExpression := "tns1:Monitoring/OperatingTime/LastReboot"
  - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
14. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
  - SubscriptionReference =: *s*
  - CurrentTime
  - TerminationTime
15. Until *timeout1* timeout expires, repeat the following steps:
  - 15.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
    - Timeout := PT60S
    - MessageLimit := 1
  - 15.2. The DUT responds with **PullMessagesResponse** message with parameters
    - CurrentTime
    - TerminationTime
    - NotificationMessage =: *m*
  - 15.3. If *m* is not null and *m* Message.Message.PropertyOperation = Initialized:
    - 15.3.1. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 16.
    - 15.3.2. If *m* Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 17.
    - 15.3.3. Set *updatedRebootTime* := *m* Message.Message.Data.SimpleItem.Status
    - 15.3.4. Go to the step 16.
  - 15.4. If *timeout1* timeout expires for step 15 without Notification with PropertyOperation = Initialized, FAIL the test and go to the step 17.



16. If *updatedRebootTime* <= *lastRebootTime*, FAIL the test and go to the next step.

17. ONVIF Client invokes **Unsubscribe** to the subscription endpoint s.

18. The DUT responds with **UnsubscribeResponse** message.

#### Test Result:

##### PASS –

- DUT passes all assertions.

##### FAIL –

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message(s).
- The DUT did not send **SystemRebootResponse** message.

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

**Note:** *timeout2* will be taken from Reboot Timeout field of ONVIF Device Test Tool.

**Note:** IPv4 address from Hello shall be used for further test cases. Previous Device service address will be selected if it is present on the XAddrs list of Hello message.

## 7.9.5 Last Clock Synchronization event

**Test Case ID:** DEVICE-9-1-5

**Specification Coverage:** Monitoring Event LastClockSynchronization

**Feature Under Test:** GetServices, GetEventProperties, CreatePullPointSubscription, PullMessages, Monitoring/OperatingTime/LastClockSynchronization event

**WSDL Reference:** devicemgmt.wsdl and event.wsdl

**Test Purpose:** To verify tns1:Monitoring/OperatingTime/LastClockSynchronization event generation after subscription and to verify tns1:Monitoring/OperatingTime/LastClockSynchronization event format.

**Pre-Requisite:** Event Service was received from the DUT. tns1:Monitoring/OperatingTime/LastClockSynchronization event is supported by the DUT as indicated by the GetEventPropertiesResponse.

**Test Configuration:** ONVIF Client and DUT

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
  - TopicNamespaceLocation list
  - FixedTopicSet
  - TopicSet =: *topicSet*
  - TopicExpressionDialect list
  - MessageContentFilterDialect list
  - MessageContentSchemaLocation list
5. If *topicSet* does not contain **tns1:Monitoring/OperatingTime/LastClockSynchronization** topic, FAIL the test and skip other steps.
6. ONVIF Client verifies **tns1:Monitoring/OperatingTime/LastClockSynchronization** topic (*lastClockSynchronizationTopic*) from *topicSet*:
  - 6.1. If *lastClockSynchronizationTopic.MessageDescription.IsProperty* is skipped or equals to false, FAIL the test and skip other steps.
  - 6.2. If *lastClockSynchronizationTopic* does not contain *MessageDescription.Data.SimpleItemDescription* item with Name = "Status", FAIL the test and skip other steps.
  - 6.3. If *lastClockSynchronizationTopic.MessageDescription.Data.SimpleItemDescription* with Name = "Status" does not have Type = "xs:dateTime", FAIL the test and skip other steps.
7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
  - Filter.TopicExpression := "tns1:Monitoring/OperatingTime/LastClockSynchronization"
  - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
  - SubscriptionReference =: s
  - CurrentTime

- TerminationTime
9. Until *timeout1* timeout expires, repeat the following steps:
- 9.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
- Timeout := PT60S
  - MessageLimit := 1
- 9.2. The DUT responds with **PullMessagesResponse** message with parameters
- CurrentTime
  - TerminationTime
  - NotificationMessage =: *m*
- 9.3. If *m* is not null and *m* Message.Message.PropertyOperation = Initialized ONVIF Client verifies *m*:
- 9.3.1. If *m* Topic does not equal to **tns1:Monitoring/OperatingTime/LastClockSynchronization**, FAIL the test and go to the step 10.
- 9.3.2. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 10.
- 9.3.3. If *m* Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 10.
- 9.3.4. If *m* Message.Message.Data.SimpleItem.Status has value time in format different from utc format with including the 'Z' indicator, FAIL the test and go to the step 10.
- 9.3.5. Go to the step 10.
- 9.4. If *timeout1* timeout expires for step 9 without Notification with PropertyOperation = Initialized, FAIL the test and go to the step 10.
10. ONVIF Client invokes **Unsubscribe** to the subscription endpoint *s*.
11. The DUT responds with **UnsubscribeResponse** message.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetEventPropertiesResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message(s).
- The DUT did not send **UnsubscribeResponse** message.

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

## 7.9.6 Last Clock Synchronization change event (SetSystemDateAndTime)

**Test Case ID:** DEVICE-9-1-6

**Specification Coverage:** Monitoring Event LastClockSynchronization

**Feature Under Test:** GetServices, CreatePullPointSubscription, PullMessages, SetSystemDateAndTime, Monitoring/OperatingTime/LastClockSynchronization event

**WSDL Reference:** devicemgmt.wsdl and event.wsdl

**Test Purpose:** To verify tns1:Monitoring/OperatingTime/LastClockSynchronization event generation after property was changed via SetSystemDateAndTime call and to verify tns1:Monitoring/OperatingTime/LastClockSynchronization event format. To verify that the clock synchronization event signals correct values.

**Pre-Requirement:** Event Service was received from the DUT. tns1:Monitoring/OperatingTime/LastClockSynchronization event is supported by the DUT as indicated by the GetEventPropertiesResponse.

**Test Configuration:** ONVIF Client and DUT

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **CreatePullPointSubscription** with parameters
  - Filter.TopicExpression := "tns1:Monitoring/OperatingTime/LastClockSynchronization"

- Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
4. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
    - SubscriptionReference =: *s*
    - CurrentTime
    - TerminationTime
  5. ONVIF Client invokes **GetSystemDateAndTime**.
  6. The DUT responds with a **GetSystemDateAndTimeResponse** message with parameters
    - SystemDateAndTime =: *initialSystemDateAndTime*
  7. Until *timeout1* timeout expires, repeat the following steps:
    - 7.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
      - Timeout := PT60S
      - MessageLimit := 1
    - 7.2. The DUT responds with **PullMessagesResponse** message with parameters
      - CurrentTime
      - TerminationTime
      - NotificationMessage =: *m*
    - 7.3. If *m* is not null and *m* Message.Message.PropertyOperation = Initialized ONVIF Client verifies *m*:
      - 7.3.1. If *m* TopicDialect does not equal to **tns1:Monitoring/OperatingTime/LastClockSynchronization**, FAIL the test and go to the step 14.
      - 7.3.2. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 14.
      - 7.3.3. If *m* Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 14.
      - 7.3.4. Go to the step 8.

- 7.4. If *timeout1* timeout expires for step 7 without Notification with PropertyOperation = Initialized, FAIL the test and go to the step 14.
8. Set *initialClockSynchronization* := *m* Message.Message.Data.SimpleItem.Status
9. ONVIF Client waits for 1 second.
10. ONVIF Client invokes **SetSystemDateAndTime** with parameters
- DateTimeType := Manual
  - DaylightSavings := false
  - TimeZone skipped
  - UTCDateTime := *initialSystemDateAndTime*.UTCDateTime received from DUT on step 6. If *initialSystemDateAndTime*.UTCDateTime is empty, then system date and time should be used to populate this parameter.
11. The DUT responds with a **SetSystemDateAndTimeResponse** message.
12. Until *timeout1* timeout expires, repeat the following steps:
- 12.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
- Timeout := PT60S
  - MessageLimit := 1
- 12.2. The DUT responds with **PullMessagesResponse** message with parameters
- CurrentTime
  - TerminationTime
  - NotificationMessage =: *m*
- 12.3. If *m* is not null and *m* Message.Message.PropertyOperation="Changed" ONVIF Client verifies *m*:
- 12.3.1. If *m* TopicDialect does not equal to **tns1:Monitoring/OperatingTime/LastClockSynchronization**, FAIL the test and go to the step 14.
- 12.3.2. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 14.

12.3.3. If *m* Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 14.

12.3.4. Set *lastClockSynchronization* := *m*  
Message.Message.Data.SimpleItem.Status

12.3.5. Go to the step 13.

12.4. If *timeout1* timeout expires for step 11 without Notification with PropertyOperation="Changed", FAIL the test and go to the step 14.

13. If *lastClockSynchronization* is less or equal to *initialClockSynchronization*, FAIL the test and go to the next step.

14. ONVIF Client restores Default System Date and Time by following the procedure mentioned in Annex A.20 with the following input and output parameters

- in *initialSystemDateAndTime* - initial system date and time settings

15. ONVIF Client invokes **Unsubscribe** to the subscription endpoint *s*.

16. The DUT responds with **UnsubscribeResponse** message.

#### Test Result:

##### PASS –

- DUT passes all assertions.

##### FAIL –

- The DUT did not send **GetSystemDateAndTimeResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message(s).
- The DUT did not send **UnsubscribeResponse** message.

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

## 7.9.7 Last Clock Synchronization change event (NTP message)

**Test Case ID:** DEVICE-9-1-7

**Specification Coverage:** Monitoring Event LastClockSynchronization

**Feature Under Test:** GetServices, CreatePullPointSubscription, PullMessages, SetSystemDateAndTime, SetNTP, Monitoring/OperatingTime/LastClockSynchronization event

**WSDL Reference:** devicemgmt.wsdl and event.wsdl

**Test Purpose:** To verify tns1:Monitoring/OperatingTime/LastClockSynchronization event generation after property was changed via an NTP message and to verify tns1:Monitoring/OperatingTime/LastClockSynchronization event format. To verify that the clock synchronization event signals correct values.

**Pre-Requirement:** Event Service was received from the DUT. tns1:Monitoring/OperatingTime/LastClockSynchronization event is supported by the DUT as indicated by the GetEventPropertiesResponse. NTP is supported by the DUT as indicated by the Network.NTP capability. A valid NTP server address should be configured in the DUT.

**Test Configuration:** ONVIF Client and DUT

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetSystemDateAndTime**.
4. The DUT responds with a **GetSystemDateAndTimeResponse** message with parameters
  - SystemDateAndTime := *initialSystemDateAndTime*
5. ONVIF Client invokes **GetNTP**.
6. The DUT responds with a **GetNTPResponse** message with parameters
  - NTPInformation := *initialNTPInformation*
7. If *initialSystemDateAndTime.DateTimeType* = NTP, ONVIF Client invokes **SetSystemDateAndTime** with parameters
  - DateTimeType := Manual
  - DaylightSavings := false
  - TimeZone skipped
  - UTCDateTime := *initialSystemDateAndTime.UTCDateTime* received from DUT on step 4. If *initialSystemDateAndTime.UTCDateTime* is empty, then system date and time should be used to populate this parameter.
8. The DUT responds with a **SetSystemDateAndTimeResponse** message.



9. ONVIF Client configures DUT with proper NTP server by following the procedure mentioned in [Annex A.21](#).
10. ONVIF Client invokes **CreatePullPointSubscription** with parameters
  - Filter.TopicExpression := "tns1:Monitoring/OperatingTime/LastClockSynchronization"
  - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
11. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
  - SubscriptionReference =: *s*
  - CurrentTime
  - TerminationTime
12. Until *timeout1* timeout expires, repeat the following steps:
  - 12.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
    - Timeout := PT60S
    - MessageLimit := 1
  - 12.2. The DUT responds with **PullMessagesResponse** message with parameters
    - CurrentTime
    - TerminationTime
    - NotificationMessage =: *m*
  - 12.3. If *m* is not null and *m* Message.Message.PropertyOperation = Initialized ONVIF Client verifies *m*:
    - 12.3.1. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step [17](#).
    - 12.3.2. If *m* Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step [17](#).
    - 12.3.3. Go to the step [13](#).
  - 12.4. If *timeout1* timeout expires for step [12](#) without Notification with PropertyOperation = Initialized, FAIL the test and go to the step [17](#).

13. Set *initialClockSynchronization* := *m* Message.Message.Data.SimpleItem.Status

14. ONVIF Client waits for 1 second.

15. ONVIF Client invokes **SetSystemDateAndTime** with parameters

- DateTimeType := NTP
- DaylightSavings := false
- TimeZone := POSIX 1003.1
- UTCDateTime skipped

16. The DUT responds with a **SetSystemDateAndTimeResponse** message.

17. Until *timeout1* timeout expires, repeat the following steps:

17.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters

- Timeout := PT60S
- MessageLimit := 1

17.2. The DUT responds with **PullMessagesResponse** message with parameters

- CurrentTime
- TerminationTime
- NotificationMessage =: *m*

17.3. If *m* is not null and *m* Message.Message.PropertyOperation="Changed" ONVIF Client verifies *m*:

17.3.1. If *m* TopicDialect does not equal to **tns1:Monitoring/OperatingTime/LastClockSynchronization**, FAIL the test and go to the step 19.

17.3.2. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 19.

17.3.3. If *m* Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 19.

17.3.4. Set *lastClockSynchronization* := *m*  
Message.Message.Data.SimpleItem.Status

17.3.5. Go to the step 18.

17.4. If *timeout1* timeout expires for step 10 without Notification with PropertyOperation="Changed", FAIL the test and go to the step 19.

18. If *lastClockSynchronization* is less or equal to *initialClockSynchronization*, FAIL the test and go to the next step.

19. ONVIF Client restores Default System Date and Time by following the procedure mentioned in Annex A.20 with the following input and output parameters

- in *initialSystemDateAndTime* - initial system date and time settings

20. ONVIF Client configures DUT with proper NTP server by following the procedure mentioned in Annex A.22 with the following input and output parameters

- in *initialNTPInformation* - initial NTP settings

21. ONVIF Client invokes **Unsubscribe** to the subscription endpoint s

22. The DUT responds with **UnsubscribeResponse** message.

#### Test Result:

##### PASS –

- DUT passes all assertions.

##### FAIL –

- The DUT did not send **GetSystemDateAndTimeResponse** message.
- The DUT did not send **GetNTPResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **SetSystemDateAndTimeResponse** message.
- The DUT did not send **PullMessagesResponse** message(s).
- The DUT did not send **UnsubscribeResponse** message.

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

## 7.9.8 Last Backup event

**Test Case ID:** DEVICE-9-1-8

**Specification Coverage:** Monitoring Event Last Backup

**Feature Under Test:** GetServices, GetEventProperties, CreatePullPointSubscription, PullMessages, Monitoring/Backup/Last event

**WSDL Reference:** devicemgmt.wsdl and event.wsdl

**Test Purpose:** To verify tns1:Monitoring/Backup/Last initial event generation and to verify tns1:Monitoring/Backup/Last event format.

**Pre-Requisite:** Event Service was received from the DUT. tns1:Monitoring/Backup/Last event is supported by the DUT as indicated by the GetEventPropertiesResponse.

**Test Configuration:** ONVIF Client and DUT

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
  - TopicNamespaceLocation list
  - FixedTopicSet
  - TopicSet =: *topicSet*
  - TopicExpressionDialect list
  - MessageContentFilterDialect list
  - MessageContentSchemaLocation list
5. If *topicSet* does not contain **tns1:Monitoring/Backup/Last** topic, FAIL the test and skip other steps.
6. ONVIF Client verifies **tns1:Monitoring/Backup/Last** topic (*lastBackupTopic*) from *topicSet*:
  - 6.1. If *lastBackupTopic*.MessageDescription.IsProperty is skipped or equals to false, FAIL the test and skip other steps.
  - 6.2. If *lastBackupTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "Status", FAIL the test and skip other steps.
  - 6.3. If *lastBackupTopic*.MessageDescription.Data.SimpleItemDescription item with Name = "Status" does not have Type = "xs:dateTime", FAIL the test and skip other steps.

7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
  - Filter.TopicExpression := "tns1:Monitoring/Backup/Last"
  - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
  - SubscriptionReference =: *s*
  - CurrentTime
  - TerminationTime
9. Until *timeout1* timeout expires, repeat the following steps:
  - 9.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
    - Timeout := PT60S
    - MessageLimit := 1
  - 9.2. The DUT responds with **PullMessagesResponse** message with parameters
    - CurrentTime
    - TerminationTime
    - NotificationMessage =: *m*
  - 9.3. If *m* is not null and *m* Message.Message.PropertyOperation = Initialized ONVIF Client verifies *m*:
    - 9.3.1. If *m* Topic does not equal to **tns1:Monitoring/Backup/Last**, FAIL the test and go to the step 10.
    - 9.3.2. If *m* does not contain Message.Message.Data.SimpleItem.Status, FAIL the test and go to the step 10.
    - 9.3.3. If *m* Message.Message.Data.SimpleItem.Status has value type different from xs:dateTime type, FAIL the test and go to the step 10.
    - 9.3.4. Go to the step 10.
  - 9.4. If *timeout1* timeout expires for step 9 without Notification with PropertyOperation = Initialized, FAIL the test and go to the step 10.

10. ONVIF Client invokes **Unsubscribe** to the subscription endpoint s

11. The DUT responds with **UnsubscribeResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetEventPropertiesResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message(s).
- The DUT did not send **UnsubscribeResponse** message.

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

## 7.9.9 Fan Failure event

**Test Case ID:** DEVICE-9-1-9

**Specification Coverage:** Monitoring Event Fan Failure

**Feature Under Test:** GetServices, GetEventProperties, CreatePullPointSubscription, PullMessages, Device/HardwareFailure/FanFailure event format

**WSDL Reference:** devicemgmt.wsdl and event.wsdl

**Test Purpose:** To verify tns1:Device/HardwareFailure/FanFailure event generation and to verify tns1:Device/HardwareFailure/FanFailure event format.

**Pre-Requisite:** Event Service was received from the DUT. tns1:Device/HardwareFailure/FanFailure event is supported by the DUT as indicated by the GetEventPropertiesResponse.

**Test Configuration:** ONVIF Client and DUT

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
  - TopicNamespaceLocation list

- FixedTopicSet
  - TopicSet =: *topicSet*
  - TopicExpressionDialect list
  - MessageContentFilterDialect list
  - MessageContentSchemaLocation list
5. If *topicSet* does not contain **tns1:Device/HardwareFailure/FanFailure** topic, FAIL the test and skip other steps.
  6. ONVIF Client verifies **tns1:Device/HardwareFailure/FanFailure** topic (*fanFailureTopic*) from *topicSet*:
    - 6.1. If *fanFailureTopic*.MessageDescription.IsProperty is skipped or equals to false, FAIL the test and skip other steps.
    - 6.2. If *fanFailureTopic* does not contain MessageDescription.Source.SimpleItemDescription item with Name = "Token", FAIL the test and skip other steps.
    - 6.3. If *fanFailureTopic*.MessageDescription.Source.SimpleItemDescription item with Name = "Token" does not have Type = "tt:ReferenceToken", FAIL the test and skip other steps.
    - 6.4. If *fanFailureTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "Failed", FAIL the test and skip other steps.
    - 6.5. If *fanFailureTopic*.MessageDescription.Data.SimpleItemDescription item with Name = "Failed" does not have Type = "xs:boolean", FAIL the test and skip other steps.
  7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
    - Filter.TopicExpression := "tns1:Device/HardwareFailure/FanFailure"
    - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
  8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
    - SubscriptionReference =: *s*
    - CurrentTime

- TerminationTime
9. Until *timeout1* timeout expires, repeat the following steps:
    - 9.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
      - Timeout := PT60S
      - MessageLimit := 1
    - 9.2. The DUT responds with **PullMessagesResponse** message with parameters
      - CurrentTime
      - TerminationTime
      - NotificationMessage =: *m*
    - 9.3. If *m* is not null and *m* Message.Message.PropertyOperation = Initialized ONVIF Client verifies *m*:
      - 9.3.1. If *m* Topic does not equal to **tns1:Device/HardwareFailure/FanFailure**, FAIL the test and go to the step 10.
      - 9.3.2. If *m* does not contain Message.Message.Source.SimpleItem.Token, FAIL the test and go to the step 10.
      - 9.3.3. If *m* Message.Message.Source.SimpleItem.Token has value type different from tt:ReferenceToken type, FAIL the test and go to the step 10.
      - 9.3.4. If *m* does not contain Message.Message.Data.SimpleItem.Failed, FAIL the test and go to the step 10.
      - 9.3.5. If *m* Message.Message.Data.SimpleItem.Failed has value type different from xs:boolean type, FAIL the test and go to the step 10.
      - 9.3.6. Go to the step 10.
    - 9.4. If *timeout1* timeout expires for step 9 without Notification with PropertyOperation = Initialized, FAIL the test and go to the step 10.
  10. ONVIF Client invokes **Unsubscribe** to the subscription endpoint *s*
  11. The DUT responds with **UnsubscribeResponse** message.

**Test Result:**



**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetEventPropertiesResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message(s).
- The DUT did not send **UnsubscribeResponse** message.

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

## 7.9.10 Power Supply Failure event

**Test Case ID:** DEVICE-9-1-10

**Specification Coverage:** Monitoring Event Power Supply Failure

**Feature Under Test:** GetServices, GetEventProperties, CreatePullPointSubscription, PullMessages, Device/HardwareFailure/PowerSupplyFailure event

**WSDL Reference:** devicemgmt.wsdl and event.wsdl

**Test Purpose:** To verify tns1:Device/HardwareFailure/PowerSupplyFailure event generation and to verify tns1:Device/HardwareFailure/PowerSupplyFailure event format.

**Pre-Requisite:** Event Service was received from the DUT. tns1:Device/HardwareFailure/PowerSupplyFailure event is supported by the DUT as indicated by the GetEventPropertiesResponse.

**Test Configuration:** ONVIF Client and DUT

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
  - TopicNamespaceLocation list
  - FixedTopicSet
  - TopicSet =: *topicSet*

- TopicExpressionDialect list
  - MessageContentFilterDialect list
  - MessageContentSchemaLocation list
5. If *topicSet* does not contain **tns1:Device/HardwareFailure/PowerSupplyFailure** topic, FAIL the test and skip other steps.
  6. ONVIF Client verifies **tns1:Device/HardwareFailure/PowerSupplyFailure** topic (*powerSupplyFailureTopic*) from *topicSet*:
    - 6.1. If *powerSupplyFailureTopic.MessageDescription.IsProperty* is skipped or equals to false, FAIL the test and skip other steps.
    - 6.2. If *powerSupplyFailureTopic* does not contain *MessageDescription.Source.SimpleItemDescription* item with Name = "Token", FAIL the test and skip other steps.
    - 6.3. If *powerSupplyFailureTopic.MessageDescription.Source.SimpleItemDescription* item with Name = "Token" does not have Type = "tt:ReferenceToken", FAIL the test and skip other steps.
    - 6.4. If *powerSupplyFailedTopic* does not contain *MessageDescription.Data.SimpleItemDescription* item with Name = "Failed", FAIL the test and skip other steps.
    - 6.5. If *powerSupplyFailureTopic.MessageDescription.Data.SimpleItemDescription* item with Name = "Failed" does not have Type = "xs:boolean", FAIL the test and skip other steps.
  7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
    - Filter.TopicExpression := "tns1:Device/HardwareFailure/PowerSupplyFailure"
    - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
  8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
    - SubscriptionReference =: s
    - CurrentTime
    - TerminationTime

9. Until *timeout1* timeout expires, repeat the following steps:
  - 9.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
    - Timeout := PT60S
    - MessageLimit := 1
  - 9.2. The DUT responds with **PullMessagesResponse** message with parameters
    - CurrentTime
    - TerminationTime
    - NotificationMessage =: *m*
  - 9.3. If *m* is not null and *m* Message.Message.PropertyOperation = Initialized ONVIF Client verifies *m*:
    - 9.3.1. If *m* Topic does not equal to **tns1:Device/HardwareFailure/PowerSupplyFailure**, FAIL the test and go to the step 10.
    - 9.3.2. If *m* does not contain Message.Message.Source.SimpleItem.Token, FAIL the test and go to the step 10.
    - 9.3.3. If *m* Message.Message.Source.SimpleItem.Token has value type different from tt:ReferenceToken type, FAIL the test and go to the step 10.
    - 9.3.4. If *m* does not contain Message.Message.Data.SimpleItem.Failed, FAIL the test and go to the step 10.
    - 9.3.5. If *m* Message.Message.Data.SimpleItem.Failed has value type different from xs:boolean type, FAIL the test and go to the step 10.
    - 9.3.6. Go to the step 10.
  - 9.4. If *timeout1* timeout expires for step 9 without Notification with PropertyOperation = Initialized, FAIL the test and go to the step 10.
10. ONVIF Client invokes **Unsubscribe** to the subscription endpoint *s*
11. The DUT responds with **UnsubscribeResponse** message.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetEventPropertiesResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message(s).
- The DUT did not send **UnsubscribeResponse** message.

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

## 7.9.11 Storage Failure event

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

**Specification Coverage:** Monitoring Event Storage Failure

**Feature Under Test:** GetServices, GetEventProperties, CreatePullPointSubscription, PullMessages, Device/HardwareFailure/StorageFailure event

**WSDL Reference:** devicemgmt.wsdl and event.wsdl

**Test Purpose:** To verify tns1:Device/HardwareFailure/StorageFailure event generation and to verify tns1:Device/HardwareFailure/StorageFailure event format.

**Pre-Requisite:** Event Service was received from the DUT. tns1:Device/HardwareFailure/StorageFailure event is supported by the DUT as indicated by the GetEventPropertiesResponse.

**Test Configuration:** ONVIF Client and DUT

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
  - TopicNamespaceLocation list
  - FixedTopicSet
  - TopicSet =: *topicSet*
  - TopicExpressionDialect list
  - MessageContentFilterDialect list
  - MessageContentSchemaLocation list

5. If *topicSet* does not contain **tns1:Device/HardwareFailure/StorageFailure** topic, FAIL the test and skip other steps.
6. ONVIF Client verifies **tns1:Device/HardwareFailure/StorageFailure** topic (*storageFailureTopic*) from *topicSet*:
  - 6.1. If *storageFailureTopic.MessageDescription.IsProperty* is skipped or equals to false, FAIL the test and skip other steps.
  - 6.2. If *storageFailureTopic* does not contain *MessageDescription.Source.SimpleItemDescription* item with Name = "Token", FAIL the test and skip other steps.
  - 6.3. If *storageFailureTopic.MessageDescription.Source.SimpleItemDescription* item with Name = "Token" does not have Type = "tt:ReferenceToken", FAIL the test and skip other steps.
  - 6.4. If *storageFailureTopic* does not contain *MessageDescription.Data.SimpleItemDescription* item with Name = "Failed", FAIL the test and skip other steps.
  - 6.5. If *storageFailureTopic.MessageDescription.Data.SimpleItemDescription* item with Name = "Failed" does not have Type = "xs:boolean", FAIL the test and skip other steps.
7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
  - Filter.TopicExpression := "tns1:Device/HardwareFailure/StorageFailure"
  - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
  - SubscriptionReference =: s
  - CurrentTime
  - TerminationTime
9. Until *timeout1* timeout expires, repeat the following steps:
  - 9.1. ONVIF Client invokes **PullMessages** to the subscription endpoint s with parameters
    - Timeout := PT60S
    - MessageLimit := 1

- 9.2. The DUT responds with **PullMessagesResponse** message with parameters
- CurrentTime
  - TerminationTime
  - NotificationMessage =: *m*
- 9.3. If *m* is not null and *m* Message.Message.PropertyOperation = Initialized ONVIF Client verifies *m*:
- 9.3.1. If *m* Topic does not equal to **tns1:Device/HardwareFailure/StorageFailure**, FAIL the test and go to the step 10.
- 9.3.2. If *m* does not contain Message.Message.Source.SimpleItem.Token, FAIL the test and go to the step 10.
- 9.3.3. If *m* Message.Message.Source.SimpleItem.Token has value type different from tt:ReferenceToken type, FAIL the test and go to the step 10.
- 9.3.4. If *m* does not contain Message.Message.Data.SimpleItem.Failed, FAIL the test and go to the step 10.
- 9.3.5. If *m* Message.Message.Data.SimpleItem.Failed has value type different from xs:boolean type, FAIL the test and go to the step 10.
- 9.3.6. Go to the step 10.
- 9.4. If *timeout1* timeout expires for step 9 without Notification with PropertyOperation = Initialized, FAIL the test and go to the step 10.
10. ONVIF Client invokes **Unsubscribe** to the subscription endpoint *s*.
11. The DUT responds with **UnsubscribeResponse** message.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetEventPropertiesResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message(s).

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

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

## 7.9.12 Critical Temperature event

**Test Case ID:** DEVICE-9-1-12

**Specification Coverage:** Monitoring Event Critical Temperature

**Feature Under Test:** GetServices, GetEventProperties, CreatePullPointSubscription, PullMessages, Device/HardwareFailure/TemperatureCritical event

**WSDL Reference:** devicemgmt.wsdl and event.wsdl

**Test Purpose:** To verify tns1:Device/HardwareFailure/TemperatureCritical event generation and to verify tns1:Device/HardwareFailure/TemperatureCritical event format.

**Pre-Requisite:** Event Service was received from the DUT. tns1:Device/HardwareFailure/TemperatureCritical event is supported by the DUT as indicated by the GetEventPropertiesResponse.

**Test Configuration:** ONVIF Client and DUT

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
  - TopicNamespaceLocation list
  - FixedTopicSet
  - TopicSet =: *topicSet*
  - TopicExpressionDialect list
  - MessageContentFilterDialect list
  - MessageContentSchemaLocation list
5. If *topicSet* does not contain **tns1:Device/HardwareFailure/TemperatureCritical** topic, FAIL the test and skip other steps.
6. ONVIF Client verifies **tns1:Device/HardwareFailure/TemperatureCritical** topic (*criticalTemperatureTopic*) from *topicSet*:

- 6.1. If *criticalTemperatureTopic*.MessageDescription.IsProperty is skipped or equals to false, FAIL the test and skip other steps.
- 6.2. If *criticalTemperatureTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "Critical", FAIL the test and skip other steps.
- 6.3. If *criticalTemperatureTopic*.MessageDescription.Data.SimpleItemDescription item with Name = "Critical" does not have Type = "xs:boolean", FAIL the test and skip other steps.
7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
  - Filter.TopicExpression := "tns1:Device/HardwareFailure/TemperatureCritical"
  - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
  - SubscriptionReference =: *s*
  - CurrentTime
  - TerminationTime
9. Until *timeout1* timeout expires, repeat the following steps:
  - 9.1. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
    - Timeout := PT60S
    - MessageLimit := 1
  - 9.2. The DUT responds with **PullMessagesResponse** message with parameters
    - CurrentTime
    - TerminationTime
    - NotificationMessage =: *m*
  - 9.3. If *m* is not null and *m* Message.Message.PropertyOperation = Initialized ONVIF Client verifies *m*:



9.3.1. If *m* Topic does not equal to **tns1:Device/HardwareFailure/TemperatureCritical**, FAIL the test and go to the step 10.

9.3.2. If *m* does not contain Message.Message.Data.SimpleItem.Critical, FAIL the test and go to the step 10.

9.3.3. If *m* Message.Message.Data.SimpleItem.Critical has value type different from xs:boolean type, FAIL the test and go to the step 10.

9.3.4. Go to the step 10.

9.4. If *timeout1* timeout expires for step 9 without Notification with PropertyOperation = Initialized, FAIL the test and go to the step 10.

10. ONVIF Client invokes **Unsubscribe** to the subscription endpoint *s*.

11. The DUT responds with **UnsubscribeResponse** message.

#### Test Result:

##### PASS –

- DUT passes all assertions.

##### FAIL –

- The DUT did not send **GetEventPropertiesResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message(s).
- The DUT did not send **UnsubscribeResponse** message.

**Note:** *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

## 8 Security Test Cases

### 8.1 USER TOKEN PROFILE

**Test Case ID:** SECURITY-1-1-1

**Specification Coverage:** Message level security

**Feature Under Test:** WS-Security Username token authentication

**WSDL Reference:** None.

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

**Pre-Requisite:**

- A user with Administrator rights (and password set) is needed in this test case. If such a user does not exist, then one shall be created before the test is executed. Similarly, if there exists such a user but the password has not been set, then the password shall be set before the test is executed. If any such changes to user settings are needed for this test case, then they should be done before starting the test sequence and the DUT should be reset to its original settings when the test sequence is finished.
- At least one operation that requires authentication is needed in this test case. If the example given in this test case (GetUsers) does not require authentication, then the test operator shall choose another operation that does require authentication.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client sends a request that requires authentication (e.g. GetUsers) to the DUT with incorrectly implemented UsernameToken. Each of the following shall be tested:
  - Missing nonce.
  - Missing timestamp.
  - Incorrect password type.
4. Verify that the DUT rejects all incorrect requests.

5. ONVIF Client sends a request (e.g. GetUsers) to the DUT with correctly formatted UsernameToken.
6. 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 Network Interface Specs] and [WS-Security].

```
<SOAP:Envelope xmlns:SOAP="..." xmlns:wsse="..." xmlns:wsu="...">
  <SOAP:Header>
    ...
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>...</wsse:Username>
        <wsse:Password Type="...#PasswordDigest">...</wsse:Password>
        <wsse:Nonce>...</wsse:Nonce>
        <wsu:Created>...</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
    ...
  </SOAP:Header>
  ...
</SOAP:Envelope>
```

**Note:** Other types of authentication shall not be used during this test.

## 8.2 DIGEST AUTHENTICATION

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

**Specification Coverage:** Security

**Feature Under Test:** HTTP Digest authentication

**WSDL Reference:** None.

**Test Purpose:** To verify that the DUT supports the HTTP Digest Authentication for HTTP level security.

**Pre-Requisite:**

- A user with Administrator rights (and password set) is needed in this test case. If such a user does not exist, then one shall be created before the test is executed. Similarly, if there exists such a user but the password has not been set, then the password shall be set before the test is executed. If any such changes to user settings are needed for this test case, then they should be done before starting the test sequence and the DUT should be reset to its original settings when the test sequence is finished.
- At least one operation that requires authentication is needed in this test case. If the example given in this test case (GetUsers) does not require authentication, then the test operator shall choose another operation that does require authentication.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client sends a request that requires authentication (e.g. GetUsers) to the DUT without any authentication.
4. Verify that the DUT rejects the request with HTTP error code 401.
5. Verify that DUT returns the required information in HTTP Response Header.
6. Send a valid request with HTTP Digest Authentication.
7. Verify that the DUT accepts the correct request.

**Test Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send HTTP error code 401 at step 3.
- The DUT did not send valid WWW-Authenticate Header at step 3 (there are no Digest indications, no realm or no nonce).
- DUT rejects request with a valid HTTP digest authentication.

**Note:** No WS-UsernameToken authentication will be used in requests for this test.

## Annex A Helper Procedures and Additional Notes

### A.1 Invalid Device Type and Scope Type

Device Type in the <d:Types> declaration:

- <d:Types> list shall include "dn:NetworkVideoTransmitter" or "tds:Device" depending on supported types, otherwise it is considered as invalid <d:Types> value.

Device Type namespaces shall also be declared in a message:

- <http://www.onvif.org/ver10/network/wsd/> for "dn:NetworkVideoTransmitter"
- <http://www.onvif.org/ver10/device/wsd/> for "tds:Device"

Invalid Scope Type:

- Scope URI is not formed according to the rules of [RFC 3986].

### A.2 Invalid Hostname, DNSname

A string which is not formed according to the rules of [RFC 952] and [RFC 1123] is considered as an invalid string.

### A.3 Invalid TimeZone

The Time Zone format is specified by POSIX, refer to POSIX 1003.1 section 8.3.

**Example:** Europe, Paris TZ=CET-1CEST,M3.5.0/2,M10.5.0/3

- CET - designation for standard time when daylight saving is not in force
- -1 - offset in hours = negative so 1 hour east of Greenwich meridian
- CEST - designation when daylight saving is in force ("Central European Summer Time")
- , - no offset number between code and comma, so default to one hour ahead for daylight saving
- M3.5.0 - when daylight saving starts = the last Sunday in March (the "5th" week means the last in the month)
- /2, - the local time when the switch occurs = 2 a.m. in this case
- M10.5.0 - when daylight saving ends = the last Sunday in October
- /3, - the local time when the switch occurs = 3 a.m. in this case

A TimeZone token which is not formed according to the rules of POSIX 1003.1 section 8.3 is considered as an invalid time zone.

## A.4 Invalid SOAP 1.2 Fault Message

A SOAP 1.2 fault message which is not formed according to the rules defined in SOAP 1.2, Part 1 Section 5.4 is considered as invalid.

## A.5 Invalid WSDL URL

An URL which is not formed according to the rules of [RFC 3986] is considered as an invalid WSDL URL.

## A.6 Valid/Invalid IPv4 Address

IPv4 Address token is represented in dotted decimal notation (32 bit internet address is divided into four 8-bit fields and each field is represented in decimal number separated by a dot).

Valid IPv4 addresses are in the range 0.0.0.0 to 255.255.255.255 excluding 0/8, 255/8, and 127/8, as defined in [RFC 758], and 169.254/16 as defined in [RFC 3927].

Valid IPv4 addresses for a device shall be valid according to the defined network mask and gateway (the gateway shall be reachable and shall not be identical to the assigned IPv4 address).

Reserved addresses such as 240.0.0.0 through 255.255.255.254, as defined in [RFC 2780] are prohibited for IPv4 devices.

## A.7 WS-Discovery Timeout Value

The ONVIF Client will use a hard coded timeout value (DISCOVERY\_TIMEOUT) when waiting for discovery responses. The value for this timeout in the respective Test result steps is:

- DISCOVERY\_TIMEOUT = 5 sec.

## A.8 Restore Network Settings

**Name:** HelperRestoreNetworkSettings

**Procedure Purpose:** Helper procedure to restore the original default network settings.

**Pre-requisite:** None.

**Input:** Original default network settings (*defaultNetworkSettings*).

**Returns:** None.

**Procedure:**

1. ONVIF Client invokes **SetNetworkInterfaces** request with parameters
  - InterfaceToken := *defaultNetworkSettings.@token*
  - NetworkInterface.Enabled := *defaultNetworkSettings.Enabled*
  - NetworkInterface.Link := *defaultNetworkSettings.Link*
  - If *defaultNetworkSettings.Info.MTU* is specified:
    - NetworkInterface.MTU := *defaultNetworkSettings.Info.MTU*otherwise NetworkInterface.MTU - skipped.
  - NetworkInterface.IPv4 := *defaultNetworkSettings.IPv4*
  - NetworkInterface.IPv6 := *defaultNetworkSettings*
  - NetworkInterface.Extension - skipped
2. The DUT responds with **SetNetworkInterfacesResponse** message with parameters
  - RebootNeeded =: *rebootNeeded*
3. If *rebootNeeded* = true:
  - 3.1. ONVIF Client invokes **SystemReboot** request.
  - 3.2. The DUT responds with **SystemRebootResponse** message with parameters
    - Message
4. The DUT sends **Hello** message from the default network interface.

**Procedure Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **SetNetworkInterfacesResponse** message.
- DUT did not send **SystemRebootResponse** message.
- DUT did not send **Hello** message during *rebootTimeout*.

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



## A.9 Subscribe and CreatePullPointSubscription for Receiving All Events

When subscribing for events an ONVIF Client might be interested in receiving all or some of the Events produced by the DUT.

If an ONVIF Client is interested in receiving some events it includes a filter tag in the CreatePullPointSubscription or Subscribe requests describing the events which the ONVIF Client is interested in (see examples in the [ONVIF Core Specs]).

If an ONVIF Client is interested in receiving all events from a device it does not include the Filter sub tag in the Subscribe or CreatePullPointSubscription request.

### Example:

The following Subscribe and CreatePullPointSubscription requests can be used if an ONVIF Client is interested in receiving all events.

1. Subscribe request:

```
<m:Subscribe xmlns:m="http://docs.oasis-open.org/wsn/b-2"
  xmlns:m0="http://www.w3.org/2005/08/addressing">
  <m:ConsumerReference>
    <m0:Address>
      http://192.168.0.1/events
    </m0:Address>
  </m:ConsumerReference>
</m:Subscribe>
```

2. CreatePullPointSubscription request

```
<m:CreatePullPointSubscription
  xmlns:m="http://www.onvif.org/ver10/events/wsd1"/>
```

## 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>
```

```

    </m:DNSInformation>
  </m:GetDNSResponse>

```

### 3. Omit tag

```

<m:GetDNSResponse xmlns:m="http://www.onvif.org/ver10/device/wsdl"
  xmlns:m0="http://www.onvif.org/ver10/schema">
  <m:DNSInformation>
    <m0:FromDHCP>>false</m0:FromDHCP>
    <m0:DNSManual>
      <m0:Type>IPv4</m0:Type>
    </m0:DNSManual>
  </m:DNSInformation>
</m:GetDNSResponse>

```

## A.11 Example of Requests for Namespaces Test Cases

For the execution of namespaces test cases, ONVIF Client shall send a request with specific namespaces definition. Examples of how this request shall look like are the following.

### 1. Defaults Namespaces Definition in Each Tag Examples

GetDNS request example:

```

<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
  <Header xmlns="http://www.w3.org/2003/05/soap-envelope">
    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <UsernameToken xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
        <Username xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
          user</Username>
        <Password xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordDigest">

```

```

5zjIbmVWxVevGlpqg6Qnt9h8Fmo=</Password>
<Nonce xmlns="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-secext-1.0.xsd">
ikcoiK+AmJvA5UpfxTzG8Q==</Nonce>
<Created xmlns="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-utility-1.0.xsd">
2011-04-25T09:27:48Z</Created>
</UsernameToken>
</Security>
</Header>
<Body xmlns="http://www.w3.org/2003/05/soap-envelope">
<GetDNS xmlns="http://www.onvif.org/ver10/device/wsdl" />
</Body>
</Envelope>

```

#### SetDNS request example:

```

<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
<Header xmlns="http://www.w3.org/2003/05/soap-envelope">
<Security xmlns="http://docs.oasis-open.org/wss/2004/
01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
<UsernameToken xmlns="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-secext-1.0.xsd">
<Username xmlns="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-secext-1.0.xsd">
service</Username>
<Password xmlns="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-secext-1.0.xsd" Type=
"http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordDigest">
znYLkZuoGqC5RrFD4KDs529JvHI=</Password>
<Nonce xmlns="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-secext-1.0.xsd">
7L0dq2/ZF3zWYEpQpFhcHA==</Nonce>
<Created xmlns="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-utility-1.0.xsd">
2011-04-25T09:27:49Z</Created>
</UsernameToken>

```

```

    </Security>
  </Header>
  <Body xmlns="http://www.w3.org/2003/05/soap-envelope">
    <SetDNS xmlns="http://www.onvif.org/ver10/device/wsdl">
      <FromDHCP xmlns="http://www.onvif.org/ver10/device/wsdl">
        false</FromDHCP>
      <DNSManual xmlns="http://www.onvif.org/ver10/device/wsdl">
        <Type xmlns="http://www.onvif.org/ver10/schema">IPv4</Type>
        <IPv4Address xmlns="http://www.onvif.org/ver10/schema">
          10.1.1.1</IPv4Address>
        </DNSManual>
      </SetDNS>
    </Body>
  </Envelope>

```

#### GetCapabilities request example:

```

<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
  <Header xmlns="http://www.w3.org/2003/05/soap-envelope">
    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <UsernameToken xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
        <Username xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
          service</Username>
        <Password xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordDigest">
          tg6EnHtyMWW8eUfntHO6XpPjOsg</Password>
        <Nonce xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
          iBIGpSuHtNPbdSWGzG48ng==</Nonce>
        <Created xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
          2011-04-25T10:54:34Z</Created>
      </UsernameToken>
    </Security>
  </Header>
  <Body xmlns="http://www.w3.org/2003/05/soap-envelope">
    <GetCapabilities xmlns="http://www.onvif.org/ver10/device/wsdl">
      <Capabilities xmlns="http://www.onvif.org/ver10/schema">
        <SupportedProtocols xmlns="http://www.onvif.org/ver10/schema">
          <Protocol xmlns="http://www.onvif.org/ver10/schema">
            <Name>ONVIF</Name>
            <Version>1.0</Version>
            <SupportedFeatures xmlns="http://www.onvif.org/ver10/schema">
              <Feature xmlns="http://www.onvif.org/ver10/schema">
                <Name>GetCapabilities</Name>
                <Version>1.0</Version>
                <SupportedVersions xmlns="http://www.onvif.org/ver10/schema">
                  <Version xmlns="http://www.onvif.org/ver10/schema">1.0</Version>
                </SupportedVersions>
              </Feature>
            </SupportedFeatures>
          </Protocol>
        </SupportedProtocols>
      </Capabilities>
    </GetCapabilities>
  </Body>
</Envelope>

```

```

    </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>

```

Probe request example:

```

<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
  <Header xmlns="http://www.w3.org/2003/05/soap-envelope">
    <MessageID xmlns="http://schemas.xmlsoap.org/ws/2004/08/
      addressing">uuid:9c35aca0-d1cc-41bf-a932-2dd0fe45cc87
    </MessageID>
    <To xmlns="http://schemas.xmlsoap.org/ws/2004/08/addressing">
      urn:schemas-xmlsoap-org:ws:2005:04:discovery</To>
    <Action xmlns="http://schemas.xmlsoap.org/ws/2004/08/
      addressing">http://schemas.xmlsoap.org/ws/2005/04/
      discovery/Probe</Action>
  </Header>
  <Body xmlns="http://www.w3.org/2003/05/soap-envelope">
    <Probe xmlns="http://schemas.xmlsoap.org/ws/2005/04/discovery">
      <Types xmlns="http://schemas.xmlsoap.org/ws/2005/04/
        discovery" xmlns:dn="http://www.onvif.org/ver10/
        network/wsdl">dn:NetworkVideoTransmitter</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

GetDNS request example:

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
  <Header>
    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <UsernameToken>
        <Username>user</Username>
        <Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordDigest">5zjIbmVWxVevGlpqg6Qnt9h8Fmo=</Password>
        <Nonce>ikcoiK+AmJvA5UpfxTzG8Q==</Nonce>
        <Created xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">2011-04-25T09:27:48Z</Created>
      </UsernameToken>
    </Security>
  </Header>
  <Body>
    <GetDNS xmlns="http://www.onvif.org/ver10/device/wsdl" />
  </Body>
</Envelope>
```

SetDNS request example:

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
  <Header>
    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <UsernameToken>
        <Username>service</Username>
        <Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0
```

```

    #PasswordDigest">znYLkZuoGqC5RrFD4KDs529JvHI=</Password>
  <Nonce>7L0dq2/ZF3zWYEpQpFhcHA==</Nonce>
  <Created xmlns="http://docs.oasis-open.org/wss/2004/01/
    oasis-200401-wss-wssecurity-utility-1.0.xsd">
    2011-04-25T09:27:49Z</Created>
  </UsernameToken>
</Security>
</Header>
<Body>
  <SetDNS xmlns="http://www.onvif.org/ver10/device/wsd1">
    <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>

```

GetCapabilities request 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/wsd1">
    <Category>Events</Category>
  </GetCapabilities>
</Body>
</Envelope>

```

Probe request example:

```

<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
  <Header>
    <MessageID xmlns="http://schemas.xmlsoap.org/ws/2004/08/
      addressing">uuid:9c35aca0-d1cc-41bf-a932-2dd0fe45cc87
    </MessageID>
    <To xmlns="http://schemas.xmlsoap.org/ws/2004/08/addressing">
      urn:schemas-xmlsoap-org:ws:2005:04:discovery</To>
    <Action xmlns="http://schemas.xmlsoap.org/ws/2004/08/
      addressing">http://schemas.xmlsoap.org/ws/2005/04/
      discovery/Probe</Action>
  </Header>
  <Body>
    <Probe xmlns="http://schemas.xmlsoap.org/ws/2005/04/discovery">
      <Types xmlns:dn="http://www.onvif.org/ver10/network/wsd1">
        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 non-Standard Prefixes Examples

GetDNS request 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/wsd1">
  <prefix1:Header>
    <prefix2:Security>
      <prefix2:UsernameToken>
        <prefix2:Username>user</prefix2:Username>
        <prefix2:Password Type="http://docs.oasis-open.org/wss/
          2004/01/oasis-200401-wss-username-token-profile-1.0
            #PasswordDigest">5zjIbmVWxVevGlpqg6Qnt9h8Fmo=
          </prefix2:Password>
        <prefix2:Nonce>ikcoiK+AmJvA5UpfxTzG8Q==</prefix2:Nonce>
        <prefix3:Created>2011-04-25T09:27:48Z</prefix3:Created>
      </prefix2:UsernameToken>
    </prefix2:Security>
  </prefix1:Header>
  <prefix1:Body>
    <prefix4:GetDNS/>
  </prefix1:Body>
</prefix1:Envelope>

```

SetDNS request 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/wsd1"
  xmlns:wsu="http://www.onvif.org/ver10/schema">
  <prefix2:Header>
    <prefix1:Security>

```

```

<prefix1:UsernameToken>
  <prefix1:Username>service</prefix1:Username>
  <prefix1:Password Type="http://docs.oasis-open.org/wss/
    2004/01/oasis-200401-wss-username-token-profile-1.0
    #PasswordDigest">znYLkZuoGqC5RrFD4KDs529JvHI=
  </prefix1:Password>
  <prefix1:Nonce>7L0dq2/ZF3zWYEpQpFhcHA==</prefix1:Nonce>
  <prefix4:Created>2011-04-25T09:27:49Z</prefix4:Created>
</prefix1:UsernameToken>
</prefix1:Security>
</prefix2:Header>
<prefix2:Body>
  <prefix3:SetDNS>
    <prefix3:FromDHCP>>false</prefix3:FromDHCP>
    <prefix3:DNSManual>
      <wsu:Type>IPv4</wsu:Type>
      <wsu:IPv4Address>10.1.1.1</wsu:IPv4Address>
    </prefix3:DNSManual>
  </prefix3:SetDNS>
</prefix2:Body>
</prefix2:Envelope>

```

GetCapabilities request example:

```

<prefix1:Envelope
  xmlns:prefix4="http://www.onvif.org/ver10/device/wsd1"
  xmlns:prefix3="http://docs.oasis-open.org/wss/2004/01/
    oasis-200401-wss-wssecurity-utility-1.0.xsd"
  xmlns:prefix1="http://www.w3.org/2003/05/soap-envelope"
  xmlns:prefix2="http://docs.oasis-open.org/wss/2004/01/
    oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <prefix1:Header>
    <prefix2:Security>
      <prefix2:UsernameToken>
        <prefix2:Username>service</prefix2:Username>
        <prefix2:Password Type="http://docs.oasis-open.org/wss/
          2004/01/oasis-200401-wss-username-token-profile-1.0
          #PasswordDigest">tg6EnHtyMWW8eUfntH06XpPjOsg=

```

```

    </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>

```

Probe request 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

GetDNS request example:

```
<p1:Envelope xmlns:p1="http://www.w3.org/2003/05/soap-envelope">
  <p2:Header xmlns:p2="http://www.w3.org/2003/05/soap-envelope">
    <p3:Security xmlns:p3="http://docs.oasis-open.org/wss/2004/
      01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <p4:UsernameToken xmlns:p4="http://docs.oasis-open.org/wss/
        2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
        <p5:Username xmlns:p5="http://docs.oasis-open.org/wss/
          2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">user
        </p5:Username>
        <p6:Password xmlns:p6="http://docs.oasis-open.org/wss/
          2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" Type
          ="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
          wss-username-token-profile-1.0#PasswordDigest">
          5zjIbmVWxVevGlpqg6Qnt9h8Fmo=</p6:Password>
        <p7:Nonce xmlns:p7="http://docs.oasis-open.org/wss/2004/01/
          oasis-200401-wss-wssecurity-secext-1.0.xsd">
          ikcoiK+AmJvA5UpfxTzG8Q==</p7:Nonce>
        <p8:Created xmlns:p8="http://docs.oasis-open.org/wss/
          2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
          2011-04-25T09:27:48Z</p8:Created>
      </p4:UsernameToken>
    </p3:Security>
  </p2:Header>
  <p9:Body xmlns:p9="http://www.w3.org/2003/05/soap-envelope">
    <p10:GetDNS xmlns:p10=
      "http://www.onvif.org/ver10/device/wsdl" />
  </p9:Body>
</p1:Envelope>
```

SetDNS request example:

```

<q1:Envelope xmlns:q1="http://www.w3.org/2003/05/soap-envelope">
  <q2:Header xmlns:q2="http://www.w3.org/2003/05/soap-envelope">
    <q3:Security xmlns:q3="http://docs.oasis-open.org/wss/2004/
      01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <q4:UsernameToken xmlns:q4="http://docs.oasis-open.org/wss/
        2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
        <q3:Username>service</q3:Username>
        <q4:Password Type="http://docs.oasis-open.org/wss/
          2004/01/oasis-200401-wss-username-token-profile-1.0
          #PasswordDigest">znYLkZuoGqC5RrFD4KDs529JvHI=
          </q4:Password>
        <q3:Nonce>7L0dq2/ZF3zWYEpQpFhcHA==</q3:Nonce>
        <q5:Created xmlns:q5="http://docs.oasis-open.org/wss/
          2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
          2011-04-25T09:27:49Z</q5:Created>
        </q4:UsernameToken>
      </q3:Security>
    </q2:Header>
    <q1:Body>
      <q6:SetDNS xmlns:q6="http://www.onvif.org/ver10/device/wsd1">
        <q7:FromDHCP xmlns:q7="http://www.onvif.org/ver10/
          device/wsd1">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>
  
```

GetCapabilities request example:

```

<p1:Envelope xmlns:p1="http://www.w3.org/2003/05/soap-envelope">
  <p2:Header xmlns:p2="http://www.w3.org/2003/05/soap-envelope">
  
```

```

    <p4:Security xmlns:p4="http://docs.oasis-open.org/wss/2004/
01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <p5:UsernameToken xmlns:p5="http://docs.oasis-open.org/wss/
2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
        <p6:Username xmlns:p6="http://docs.oasis-open.org/wss/
2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
          service</p6:Username>
        <p7:Password xmlns:p7="http://docs.oasis-open.org/wss/
2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" Type
="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordDigest">
          tg6EnHtyMWW8eUfntHO6XpPjOsg=</p7:Password>
        <p8:Nonce xmlns:p8="http://docs.oasis-open.org/wss/
2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
          iBIGpSuHtNPbdSWGzG48ng==</p8:Nonce>
        <p9:Created xmlns:p9="http://docs.oasis-open.org/wss/
2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
          2011-04-25T10:54:34Z</p9:Created>
      </p5:UsernameToken>
    </p4:Security>
  </p2:Header>
  <p3:Body xmlns:p3="http://www.w3.org/2003/05/soap-envelope">
    <p10:GetCapabilities
      xmlns:p10="http://www.onvif.org/ver10/device/wsd">
      <p11:Category xmlns:p11="http://www.onvif.org/ver10/
device/wsd">Events</p11:Category>
    </p10:GetCapabilities>
  </p3:Body>
</p1:Envelope>

```

Probe request example:

```

<p1:Envelope xmlns:p1="http://www.w3.org/2003/05/soap-envelope">
  <p2:Header xmlns:p2="http://www.w3.org/2003/05/soap-envelope">
    <p3:MessageID xmlns:p3="http://schemas.xmlsoap.org/ws/2004/08/
addressing">uuid:9c35aca0-d1cc-41bf-a932-2dd0fe45cc87
    </p3:MessageID>
  <p4:To xmlns:p4="http://schemas.xmlsoap.org/ws/2004/08/

```

```

    addressing">urn:schemas-xmlsoap-org:ws:2005:04:discovery
  </p4:To>
  <p5:Action xmlns:p5="http://schemas.xmlsoap.org/ws/2004/08/
    addressing">http://schemas.xmlsoap.org/ws/2005/04/
    discovery/Probe</p5:Action>
</p2:Header>
<p6:Body xmlns:p6="http://www.w3.org/2003/05/soap-envelope">
  <p7:Probe xmlns:p7="http://schemas.xmlsoap.org/ws/2005/04/
    discovery">
    <p8:Types xmlns:p8="http://schemas.xmlsoap.org/ws/2005/04/
      discovery" xmlns:dn="http://www.onvif.org/ver10/
      network/wsdl">dn:NetworkVideoTransmitter</p8: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

GetDNS request example:

```

<p1:Envelope xmlns:p1="http://www.w3.org/2003/05/soap-envelope">
  <p1:Header>
    <p1:Security xmlns:p1="http://docs.oasis-open.org/wss/2004/
    01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <p1:UsernameToken>
        <p1:Username>user</p1:Username>
        <p1:Password Type="http://docs.oasis-open.org/wss/
          2004/01/oasis-200401-wss-username-token-profile-1.0
          #PasswordDigest">5zjIbmVxVevGlpqg6Qnt9h8Fmo=
        </p1:Password>
        <p1:Nonce>ikcoiK+AmJvA5UpfxTzG8Q==</p1:Nonce>
        <p1:Created xmlns:p1="http://docs.oasis-open.org/wss/
          2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
          2011-04-25T09:27:48Z</p1:Created>

```



```

    </p1:UsernameToken>
  </p1:Security>
</p1:Header>
<p1:Body>
  <p1:GetDNS xmlns:p1="http://www.onvif.org/ver10/device/wsdl" />
</p1:Body>
</p1:Envelope>

```

### SetDNS request example:

```

<q1:Envelope xmlns:q1="http://www.w3.org/2003/05/soap-envelope">
  <q1:Header>
    <q1:Security xmlns:q1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <q1:UsernameToken>
        <q1:Username>service</q1:Username>
        <q1:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordDigest">znYLkZuoGqC5RrFD4KDs529JvHI=
        </q1:Password>
        <q1:Nonce>7L0dq2/ZF3zWYEpQpFhcHA==</q1:Nonce>
        <q1:Created xmlns:q1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">2011-04-25T09:27:49Z</q1:Created>
      </q1:UsernameToken>
    </q1:Security>
  </q1:Header>
  <q1:Body>
    <q1:SetDNS xmlns:q1="http://www.onvif.org/ver10/device/wsdl">
      <q1:FromDHCP>>false</q1:FromDHCP>
      <q1:DNSManual>
        <q1:Type xmlns:q1="http://www.onvif.org/ver10/schema">IPv4</q1:Type>
        <q1:IPv4Address xmlns:q1="http://www.onvif.org/ver10/schema">10.1.1.1</q1:IPv4Address>
      </q1:DNSManual>
    </q1:SetDNS>
  </q1:Body>

```

```
</q1:Envelope>
```

#### GetCapabilities request example:

```
<p1:Envelope xmlns:p1="http://www.w3.org/2003/05/soap-envelope">
  <p1:Header>
    <p1:Security xmlns:p1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <p1:UsernameToken>
        <p1:Username>service</p1:Username>
        <p1:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordDigest">tg6EnHtyMWW8eUfntH06XpPjOsg=
          </p1:Password>
        <p1:Nonce>iBIGpSuHtNPbdSWGzG48ng==</p1:Nonce>
        <p1:Created xmlns:p1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
          2011-04-25T10:54:34Z</p1:Created>
      </p1:UsernameToken>
    </p1:Security>
  </p1:Header>
  <p1:Body>
    <p1:GetCapabilities
      xmlns:p1="http://www.onvif.org/ver10/device/wsdl">
      <p1:Category>Events</p1:Category>
    </p1:GetCapabilities>
  </p1:Body>
</p1:Envelope>
```

#### Probe request example:

```
<p1:Envelope xmlns:p1="http://www.w3.org/2003/05/soap-envelope">
  <p1:Header xmlns:p1="http://www.w3.org/2003/05/soap-envelope">
    <p1:MessageID xmlns:p1="http://schemas.xmlsoap.org/ws/2004/08/addressing">uuid:9c35aca0-d1cc-41bf-a932-2dd0fe45cc87
    </p1:MessageID>
```

```
<p1:To xmlns:p1="http://schemas.xmlsoap.org/ws/2004/08/
  addressing">urn:schemas-xmlsoap-org:ws:2005:04:discovery
</p1:To>
<p1:Action xmlns:p1="http://schemas.xmlsoap.org/ws/2004/08/
  addressing">http://schemas.xmlsoap.org/ws/2005/04/
  discovery/Probe</p1:Action>
</p1:Header>
<p1:Body xmlns:p1="http://www.w3.org/2003/05/soap-envelope">
  <p1:Probe
    xmlns:p1="http://schemas.xmlsoap.org/ws/2005/04/discovery">
    <p1:Types xmlns:p1="http://schemas.xmlsoap.org/ws/2005/04/
      discovery" xmlns:dn="http://www.onvif.org/ver10/
      network/wsdl">dn:NetworkVideoTransmitter</p1: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>
```

## A.12 Procedure to Turn On IPv4 DHCP

**Name:** HelperTurnOnIPv4DHCP

**Procedure Purpose:** Helper procedure to turn on IPv4 DHCP.

**Pre-requisite:** None.

**Input:** None.

**Returns:** Original default network settings (*defaultNetworkSettings*).

**Procedure:**

1. ONVIF Client invokes **GetNetworkInterfaces** request.
2. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
  - NetworkInterfaces list := *networkInterfacesList*
3. Set *defaultNetworkSettings* := current interface from *networkInterfacesList*.

4. If *defaultNetworkSettings.IPv4.Config.DHCP* = false:
  - 4.1. ONVIF Client invokes **SetNetworkInterfaces** request with parameters
    - *InterfaceToken* := *defaultNetworkSettings.@token*
    - *NetworkInterface.Enabled* skipped
    - *NetworkInterface.Link* skipped
    - *NetworkInterface.MTU* skipped
    - *NetworkInterface.IPv4.Enabled* := true
    - *NetworkInterface.IPv4.Manual* skipped
    - *NetworkInterface.IPv4.DHCP* := true
    - *NetworkInterface.IPv6* skipped
    - *NetworkInterface.Extension* - skipped
  - 4.2. The DUT responds with **SetNetworkInterfacesResponse** message with parameters
    - *RebootNeeded* =: *rebootNeeded*
  - 4.3. If *rebootNeeded* = true:
    - 4.3.1. ONVIF Client invokes **SystemReboot** request.
    - 4.3.2. The DUT responds with **SystemRebootResponse** message with parameters
      - *Message*
  - 4.4. The DUT sends **Hello** message from the default network interface.

**Procedure Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetNetworkInterfacesResponse** message.
- DUT did not send **SetNetworkInterfacesResponse** message.
- DUT did not send **SystemRebootResponse** message.

- DUT did not send **Hello** message during *rebootTimeout*.

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

## A.13 Procedure to Turn On IPv4 DHCP

**Name:** HelperTurnOffIPv4DHCP

**Procedure Purpose:** Helper procedure to turn off IPv4 DHCP.

**Pre-requisite:** None.

**Input:** None.

**Returns:** Original default network settings (*defaultNetworkSettings*).

**Procedure:**

1. ONVIF Client invokes **GetNetworkInterfaces** request.
2. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
  - NetworkInterfaces list := *networkInterfacesList*
3. Set *defaultNetworkSettings* := current interface from *networkInterfacesList*.
4. If *defaultNetworkSettings*.IPv4.Config.DHCP = true:
  - 4.1. ONVIF Client invokes **SetNetworkInterfaces** request with parameters
    - InterfaceToken := *defaultNetworkSettings*.@token
    - NetworkInterface.Enabled skipped
    - NetworkInterface.Link skipped
    - NetworkInterface.MTU skipped
    - NetworkInterface.IPv4.Enabled := true
    - NetworkInterface.IPv4.Manual.Address := current IPv4 address of the DUT
    - NetworkInterface.IPv4.Manual.PrefixLength := current IPv4 prefix of the DUT
    - NetworkInterface.IPv4.DHCP := false
    - NetworkInterface.IPv6 skipped

- NetworkInterface.Extension skipped
- 4.2. The DUT responds with **SetNetworkInterfacesResponse** message with parameters
    - RebootNeeded =: *rebootNeeded*
  - 4.3. If *rebootNeeded* = true:
    - 4.3.1. ONVIF Client invokes **SystemReboot** request.
    - 4.3.2. The DUT responds with **SystemRebootResponse** message with parameters
      - Message
  - 4.4. The DUT sends **Hello** message from the default network interface.

**Procedure Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetNetworkInterfacesResponse** message.
- DUT did not send **SetNetworkInterfacesResponse** message.
- DUT did not send **SystemRebootResponse** message.
- DUT did not send **Hello** message during *rebootTimeout*.

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

## A.14 Procedure to Turn On IPv6 DHCP

**Name:** HelperTurnOnIPv6DHCP

**Procedure Purpose:** Helper procedure to turn on IPv6 DHCP.

**Pre-requisite:** IPv6 is supported by DUT.

**Input:** None.

**Returns:** Original default network settings (*defaultNetworkSettings*).

**Procedure:**

1. ONVIF Client invokes **GetNetworkInterfaces** request.
2. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
  - NetworkInterfaces list := *networkInterfacesList*
3. Set *defaultNetworkSettings* := interface from *networkInterfacesList* with IPv6 support.
4. If *defaultNetworkSettings.IPv6.Config.DHCP* = Off:
  - 4.1. ONVIF Client invokes **SetNetworkInterfaces** request with parameters
    - InterfaceToken := *defaultNetworkSettings.@token*
    - NetworkInterface.Enabled = true
    - NetworkInterface.Link skipped
    - NetworkInterface.MTU skipped
    - NetworkInterface.IPv4 skipped
    - NetworkInterface.IPv6.Enabled := true
    - NetworkInterface.IPv6.AcceptRouterAdvert skipped
    - NetworkInterface.IPv6.Manual skipped
    - NetworkInterface.IPv6.DHCP := Auto
    - NetworkInterface.Extension - skipped
  - 4.2. The DUT responds with **SetNetworkInterfacesResponse** message with parameters
    - RebootNeeded := *rebootNeeded*
  - 4.3. If *rebootNeeded* = true:
    - 4.3.1. ONVIF Client invokes **SystemReboot** request.
    - 4.3.2. The DUT responds with **SystemRebootResponse** message with parameters
      - Message
  - 4.4. The DUT sends **Hello** message from the default network interface.

**Procedure Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetNetworkInterfacesResponse** message.
- DUT did not send **SetNetworkInterfacesResponse** message.
- DUT did not send **SystemRebootResponse** message.
- DUT did not send **Hello** message during *rebootTimeout*.

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

## A.15 Procedure to Turn Off IPv6 DHCP

**Name:** HelperTurnOffIPv6DHCP

**Procedure Purpose:** Helper procedure to turn off IPv6 DHCP.

**Pre-requisite:** IPv6 is supported by DUT.

**Input:** None.

**Returns:** Original default network settings (*defaultNetworkSettings*).

**Procedure:**

1. ONVIF Client invokes **GetNetworkInterfaces** request.
2. The DUT responds with **GetNetworkInterfacesResponse** message with parameters
  - NetworkInterfaces list =: *networkInterfacesList*
3. Set *defaultNetworkSettings* := interface from *networkInterfacesList* with IPv6 support.
4. If *defaultNetworkSettings*.IPv6.Config.DHCP != Off:
  - 4.1. ONVIF Client invokes **SetNetworkInterfaces** request with parameters
    - InterfaceToken := *defaultNetworkSettings*.@token
    - NetworkInterface.Enabled = true
    - NetworkInterface.Link skipped
    - NetworkInterface.MTU skipped



- NetworkInterface.IPv4 skipped
  - NetworkInterface.IPv6.Enabled := true
  - NetworkInterface.IPv6.AcceptRouterAdvert skipped
  - If *defaultNetworkSettings.IPv6.Config.FromDHCP* is specified:
    - NetworkInterface.IPv6.Manual :=  
*defaultNetworkSettings.IPv6.Config.FromDHCP*
    - otherwise NetworkInterface.IPv6.Manual skipped.
  - NetworkInterface.IPv6.DHCP := Auto
  - NetworkInterface.Extension - skipped
- 4.2. The DUT responds with **SetNetworkInterfacesResponse** message with parameters
- RebootNeeded =: *rebootNeeded*
- 4.3. If *rebootNeeded* = true:
- 4.3.1. ONVIF Client invokes **SystemReboot** request.
- 4.3.2. The DUT responds with **SystemRebootResponse** message with parameters
- Message
- 4.4. The DUT sends **Hello** message from the default network interface.

**Procedure Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetNetworkInterfacesResponse** message.
- DUT did not send **SetNetworkInterfacesResponse** message.
- DUT did not send **SystemRebootResponse** message.
- DUT did not send **Hello** message during *rebootTimeout*.

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

## A.16 Name and Token Parameters Maximum Length

There are the following limitations on maximum length of Name and Token parameters that shall be used during tests by ONVIF Device Test Tool to prevent faults from the DUT:

- Name shall be less than or equal to 64 characters (only readable characters are accepted).
- Token shall be less than or equal to 64 characters (only readable characters are accepted).

UTF-8 character set shall be used for Name and Token.

**Note:** these limitations will not be used if ONVIF Device Test Tool re-uses values that were received from the DUT.

## A.17 TooManyUsers Fault Check

**Name:** HelperTooManyUsersFaultCheck

**Procedure Purpose:** Helper procedure to check if the DUT correctly returns TooManyUsers fault to CreateUsers request.

**Pre-requisite:** None

**Input:** Current number of users (*usersNumber*).

**Returns:** None.

**Procedure:**

1. If DUT does not support GetServices, skip other steps.
2. ONVIF Client invokes **GetServiceCapabilities** request.
3. The DUT responds with **GetServiceCapabilitiesResponse** message with parameters
  - Capabilities =: *capabilities*
4. If *capabilities.Security.MaxUsers* is not specified, skip other steps.
5. If *capabilities.Security.MaxUsers* != *usersNumber*, FAIL the test and skip other steps.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

## A.18 Get Service Capabilities (Device Management)

**Name:** HelperGetServiceCapabilities

**Procedure Purpose:** Helper procedure to retrieve Device Management Service Capabilities.

**Pre-requisite:** GetServices command is supported by the DUT.

**Input:** None.

**Returns:** Device Management Service Capabilities (*cap*).

**Procedure:**

1. ONVIF Client invokes **GetServiceCapabilities** request.
2. The DUT responds with **GetServiceCapabilitiesResponse** message with parameters
  - Capabilities =: *cap*

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

## A.19 Get Capabilities (Device Management)

**Name:** HelperGetDeviceCapabilities

**Procedure Purpose:** Helper procedure to retrieve Device Management Capabilities.

**Pre-requisite:** GetCapabilities command is supported by the DUT.

**Input:** None.

**Returns:** Device Management Capabilities (*deviceManagementCap*).

**Procedure:**

1. ONVIF Client invokes **GetCapabilities** request.
2. The DUT responds with **GetCapabilitiesResponse** message with parameters
  - Capabilities =: *cap*
3. Set *deviceManagementCap* := *cap.Device*.

**Procedure Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

## A.20 Restoring System Date and Time

**Name:** HelperRestoreSystemDateAndTime

**Procedure Purpose:** Helper procedure to restoring System Date and Time.

**Pre-requisite:** None.

**Input:** The SystemDateAndTime (*initialSystemDateAndTime*) to restore.

**Returns:** None.

**Procedure:**

1. ONVIF Client invokes **SetSystemDateAndTime** request with parameters
  - DateTimeType := *initialSystemDateAndTime.DateTimeType*
  - DaylightSavings := *initialSystemDateAndTime.DaylightSavings*
  - TimeZone := *initialSystemDateAndTime.TimeZone*
  - If *initialSystemDateAndTime.DateTimeType* = Manual:
    - UTCDateTime := current UTC date and time
  - otherwise *initialSystemDateAndTime.DateTimeType* skipped.
2. The DUT responds with **SetSystemDateAndTimeResponse** message.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

## A.21 Set NTP Settings

**Name:** HelperSetNTP

**Procedure Purpose:** Helper procedure to configure DUT with proper NTP server.

**Pre-requisite:** NTP is supported by the DUT.

**Input:** None.

**Returns:** None.

**Procedure:**

1. ONVIF Client invokes **SetNTP** request with parameters
  - FromDHCP := false
  - NTPManual[0].Type := IPv4
  - NTPManual[0].IPv4Address := *validNTPServerAddress*
  - NTPManual[0].IPv6Address skipped
  - NTPManual[0].DNSname skipped
  - NTPManual[0].Extension skipped
2. The DUT responds with **SetNTPResponse** message.

**Procedure Result:****PASS –**

- DUT passes all assertions.

**FAIL –**

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

**Note:** *validNTPServerAddress* will be taken from NTP IPv4 field of ONVIF Device Test Tool.

## A.22 Restoring NTP Settings

**Name:** HelperRestoreNTP

**Procedure Purpose:** Helper procedure to restoring NTP settings.

**Pre-requisite:** NTP is supported by the DUT.

**Input:** The initial NTP settings (*initialNTPInformation*) to restore.

**Returns:** None.

**Procedure:**

1. ONVIF Client invokes **SetNTP** request with parameters
  - FromDHCP := *initialNTPInformation.FromDHCP*
  - NTPManual := *initialNTPInformation.NTPManual*
2. The DUT responds with **SetNTPResponse** message.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

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

**Note:** *validNTPServerAddress* will be taken from NTP IPv4 field of ONVIF Device Test Tool.