Driving IP-based physical security through global standardization

ONVIF^{TM}

Schedule Test Specification

Version 17.01

January 2017

© 2017 by 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

Ver.	Date	Description
15.06		First issue of Schedule Test Specification
16.09	Sept, 2016	Annex A.7 was updated with changing of DTEND value
17.01	Nov, 2016	CREATE SCHEDULE - INVALID TIME RANGE INTERVAL: fault code was changed

Table of Contents

1 Introduction

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

1.1 Scope

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

The principal intended purposes are:

1. To provide self-assessment tool for implementations.
2. To provide comprehensive test suite coverage for [ONVIF Network Interface Specs].

This specification does not address the following.

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

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

This ONVIF Schedule Test Specification covers the ONVIF Schedule Service, which is a functional block of [ONVIF Network Interface Specs]. The following section gives a brief overview of each functional block and its scope.

1.1.1 Capabilities

The Capabilities section covers the test cases needed for getting capabilities from an ONVIF device.

The scope of this specification section is to cover the following functions:

- Getting Schedule service address with GetServices command via Device service
- Getting capabilities with GetServiceCapabilities command
- Getting capabilities with GetServices command via Device service

1.1.2 Schedule Info

The Schedule Info section covers the test cases needed for getting schedule list and information from an ONVIF device.

The scope of this specification section is to cover the following functions:

- Getting schedule information with GetScheduleInfo command
- Getting schedule information list with GetScheduleInfoList command

1.1.3 Schedule

The Schedule section covers the test cases needed for getting schedule from an ONVIF device.

The scope of this specification section is to cover the following functions:

- Getting schedule with GetSchedules command
- Getting schedule list with GetScheduleList command
- Creating schedule with CreateSchedule command
- Modifying schedule with ModifySchedule command
- Deleting schedule with DeleteSchedule command

1.1.4 Special Day Group Info

The Special Day Group Info section covers the test cases needed for getting Special Day Group list and information from an ONVIF device.

The scope of this specification section is to cover the following functions:

- Getting SpecialDayGroup information with GetSpecialDayGroupInfo command
- Getting SpecialDayGroup information list with GetSpecialDayGroupInfoList command

1.1.5 Special Day Group

The Special Day Group section covers the test cases needed for getting Special Day Group from an ONVIF device.

The scope of this specification section is to cover the following functions:

- Getting special day group with GetSpecialDayGroups command
- Getting special day group list with GetSpecialDayGroupList command
- Creating special day group with CreateSpecialDayGroup command
- Modifying special day group with ModifySpecialDayGroup command
- Deleting special day group with DeleteSpecialDayGroup command

1.1.6 Schedule States

The Schedule States section covers the test cases needed for getting schedule states from an ONVIF device.

The scope of this specification section is to cover the following functions:

- Getting schedule states with GetScheduleStates command
- Changing schedule states

1.1.7 Schedule Events

The Schedule Events section covers the test cases needed for for checking specified events format.

The scope of this specification section is to cover the following functions:

- Getting event properties with GetEventProperties command

1.1.8 Consistency

Consistency test cases cover verification of consistency between different entities and commands.

Consistency between the following entities is covered by the following test case:

- Schedule and Special Day Info

2 Terms and Definitions

2.1 Definitions

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

iCalendar An industry standard format for exchanging scheduling and activity-recording information electronically.

Schedule A set of time periods, e.g. working hours (weekdays from 8 AM to 6 PM). It may also include one or more Special Days Schedules.

Special Days A set of dates that require the regular Schedule to be overridden, e.g. holidays, half-days or working Sundays.

Special Days Schedule A schedule that defines time periods for a Special Day List.

Time Period A time period has a start time and an end time, e.g. 8 AM to 6 PM.

vEvent A component in iCalendar, specifying the properties of an event.

2.2 Abbreviations

This section describes abbreviations used in this document.

DUT	Device Under Test
HTTP	Hypertext Transfer Protocol
PACS	Physical Access Control System

3 Test Overview

This section provides information the test setup procedure and required prerequisites, and the test policies that should be followed for test case execution.

3.1 Test Setup

3.1.1 Network Configuration for DUT

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

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

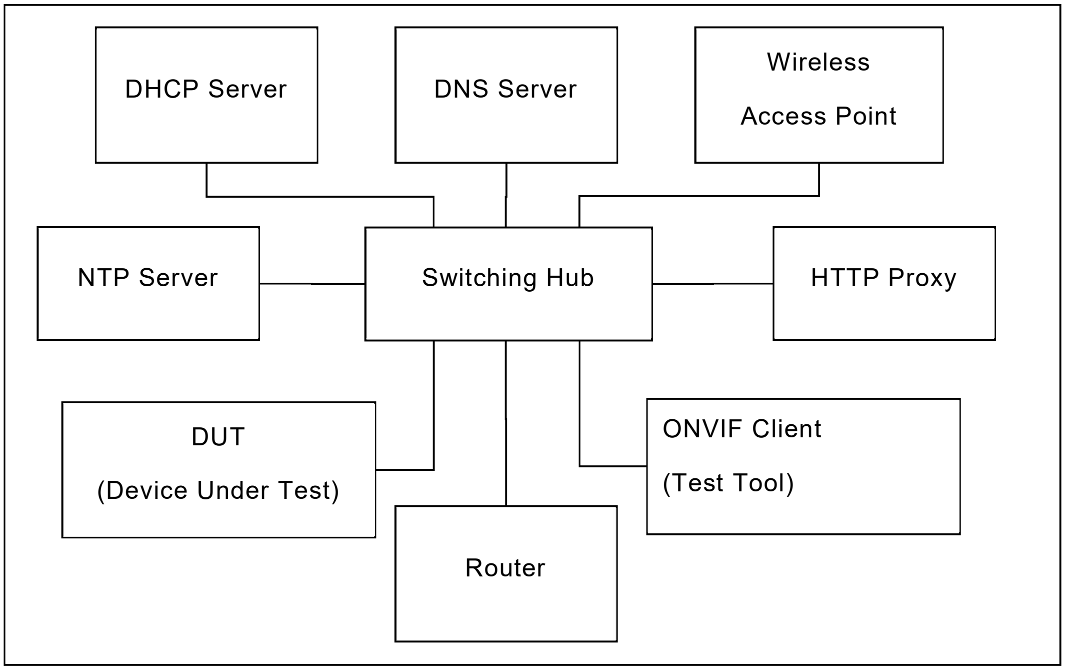


Figure 1: Test Configuration for DUT

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

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

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

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

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

DHCP Server: provides IPv4 Address to the connected devices.

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

3.2 Prerequisites

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

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

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

3.3 Test Policy

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

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

3.3.1 Capabilities

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

- DUT shall give the Schedule Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.
- DUT shall support the following commands:
  - GetServices
  - GetServiceCapabilities
- The following tests are performed
  - Getting capabilities with GetServiceCapabilities command
  - Getting capabilities with GetServices command

Please, refer to Section 4.1 for Capabilities Test Cases.

3.3.2 Schedule Info

The test policies specific to the test case execution of Schedule Info functional block:

- DUT shall give the Schedule Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.
- DUT shall support the following commands:
  - GetServices
  - GetServiceCapabilities
  - GetScheduleInfo

  - GetScheduleInfoList

- DUT shall not return more items in GetScheduleInfo and GetScheduleInfoList responses than specified in service capabilities by MaxLimit.

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

- DUT shall not return items with the same tokens in GetScheduleInfoList responses for one schedule info list resieving.

- DUT shall not return more ScheduleInfo items in GetScheduleInfoList responses than specified in service capabilities by MaxSchedules.

- DUT shall not return any fault if GetScheduleInfo was invoked for non-exciting Schedule token. Such tokens shall be ignored.

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

- The following tests are performed
  - Getting schedule info with GetScheduleInfo command
  - Getting schedule info list with GetScheduleInfoList command with using different Limit and NextReference values
  - Getting schedule info with invalid schedule token
  - Getting schedule info with number of requested items is greater than MaxLimit

Please refer to Section 4.2 for Schedule Info Test Cases.

3.3.3 Schedule

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

- DUT shall give the Schedule Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.

- DUT shall support the following commands:
  - GetSchedules
  - GetScheduleList
  - GetScheduleInfoList
  - CreateSchedule
  - ModifySchedule
  - DeleteSchedule

- DUT shall return only requested items in GetSchedules response that specified in GetSchedules request.

- DUT shall return all requested items in GetSchedules response that specified in

GetSchedules request.

- DUT shall not return more items in GetSchedules responses than specified in service capabilities by MaxLimit.
- DUT shall return the same information in GetSchedules responses and in GetScheduleInfoList responses for the items with the same token.
- DUT shall not return more items in GetScheduleList response than specified by Limit parameter in a request.
- DUT shall not return items with the same tokens in GetScheduleList responses for one schedule list resieving.
- DUT shall return the same information in GetSchedules responses and in GetScheduleList responses for the items with the same token.
- DUT shall return the same information in GetScheduleList responses and in GetScheduleInfoList responses for the items with the same token.
- DUT shall return the schedules in GetScheduleList responses and in GetScheduleInfoList responses.
- DUT shall return SOAP 1.2 fault message (InvalidArgs/TooManyItems) if more items than MaxLimit was requested by GetSchedules command.
- The DUT shall support creating of schedule.
- The DUT shall support modifying of schedule.
- The DUT shall support deleting of schedule.
- DUT shall return SOAP 1.2 fault message (InvalidArgs) if schedule token is specified in CreateSchedule request.
- DUT should return SOAP 1.2 fault message (InvalidArgVal/NotFound) if ModifySchedule or DeleteSchedule command was invoked for non-exciting schedule token.
- The following tests are performed
  - Getting schedule with GetSchedule command and test that it includes the same information with GetScheduleInfoList command
  - Getting schedule info list with GetScheduleList command with using different Limit and NextReference values and test that it includes the same information with GetScheduleInfoList command
  - Creating schedule with CreateSchedule command with empty token and test that corresponding notification message is received
  - Modifying schedule with ModifySchedule command and test that corresponding notification message is received
  - Deleting schedule with DeleteSchedule command and test that corresponding notification message is received
  - Getting schedules with invalid schedule token

  - Getting schedules with number of requested items is greater than MaxLimit
  - Creating schedule with CreateSchedule command with specified token
  - Modifying schedule with ModifySchedule command with invalid token
  - Deleting schedule with DeleteSchedule command with invalid token

Please refer to Section 4.3 for Schedule Test Cases.

3.3.4 Special Day Group Info

The test policies specific to the test case execution of SpecialDayGroup Info Info functional block:

- DUT shall give the Schedule Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.
- DUT shall support the following commands:
  - GetServices
  - GetServiceCapabilities
- If DUT supports Special Days as indicated by the Capabilities.SpecialDaysSupported, then DUT shall support the following commands:
  - GetSpecialDayGroupInfo
  - GetSpecialDayGroupInfoList
- DUT shall not return more items in GetSpecialDayGroupInfo and GetSpecialDayGroupInfoList responses than specified in service capabilities by MaxLimit.
- DUT shall not return more items in GetSpecialDayGroupInfoList response than specified by Limit parameter in a request.
- DUT shall not return items with the same tokens in GetSpecialDayGroupInfoList responses for one special day group info list resieving.
- DUT shall not return more SpecialDayGroupInfo items in GetSpecialDayGroupInfoList responses than specified in service capabilities by MaxSpecialDayGroups.
- DUT shall not return any fault if GetSpecialDayGroupInfo was invoked for non-exciting SpecialDayGroup token. Such tokens shall be ignored.
- DUT shall return SOAP 1.2 fault message (InvalidArgs/TooManyItems) if more items than MaxLimit was requested by GetSpecialDayGroupInfo command.
- The following tests are performed if DUT supports Special Days:
  - Getting special day group info with GetSpecialDayGroupInfo command
  - Getting special day group info list with GetSpecialDayGroupInfoList command with using different Limit and NextReference values
  - Getting special day group info with invalid special day group token

  - Getting special day group info with number of requested items is greater than MaxLimit

Please refer to Section 0 for Special Day Group Info Test Cases.

3.3.5 Special Day Group

The test policies specific to the test case execution of Special Day Group functional block:

- DUT shall give the Schedule Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.
- DUT shall support the following commands:
  - GetServices
  - GetServiceCapabilities
- If DUT supports Special Days as indicated by the Capabilities.SpecialDaysSupported, then DUT shall support the following commands:
  - GetSpecialDayGroups
  - GetSpecialDayGroupList
  - GetSpecialDayGroupInfoList
  - CreateSpecialDayGroup
  - ModifySpecialDayGroup
  - DeleteSpecialDayGroup
- DUT shall return only requested items in GetSpecialDayGroups response that specified in GetSpecialDayGroups request.
- DUT shall return all requested items in GetSpecialDayGroups response that specified in GetSpecialDayGroups request.
- DUT shall not return more items in GetSpecialDayGroups responses than specified in service capabilities by MaxLimit.
- DUT shall return the same information in GetSpecialDayGroups responses and in GetSpecialDayGroupInfoList responses for the items with the same token.
- DUT shall not return more items in GetSpecialDayGroupList response than specified by Limit parameter in a request.
- DUT shall not return items with the same tokens in GetSpecialDayGroupList responses for one special day group list resieving.
- DUT shall return the same information in GetSpecialDayGroups responses and in GetSpecialDayGroupList responses for the items with the same token.
- DUT shall return the same information in GetSpecialDayGroupList responses and in GetSpecialDayGroupInfoList responses for the items with the same token.
- DUT shall return the special day groups in GetSpecialDayGroupList responses and in

GetSpecialDayGroupInfoList responses.

- The DUT shall support creating of special day group.
- The DUT shall support modifying of special day group.
- The DUT shall support deleting of special day group.
- DUT shall return SOAP 1.2 fault message (InvalidArgs) if special day group token is specified in CreateSpecialDayGroup request.
- DUT should return SOAP 1.2 fault message (InvalidArgVal/NotFound) if ModifySpecialDayGroup or DeleteSpecialDayGroup command was invoked for non-exciting special day group token.
- The following tests are performed if DUT supports Special Days:
    - Getting special day group with GetSpecialDayGroups command and test that it includes the same information with GetSpecialDayGroupInfoList command
    - Getting special day group info list with GetSpecialDayGroupList command with using different Limit and NextReference values and test that it includes the same information with GetSpecialDayGroupInfoList command
    - Creating special day group with CreateSpecialDayGroup command with empty token and test that corresponding notification message is received
    - Modifying special day group with ModifySpecialDayGroup command and test that corresponding notification message is received
    - Deleting special day group with DeleteSpecialDayGroup command and test that corresponding notification message is received
    - Getting special day groups with invalid special day group token
    - Getting special day groups with number of requested items is greater than MaxLimit
    - Creating special day group with CreateSpecialDayGroup command with specified token
    - Modifying special day group with ModifySpecialDayGroup command with invalid token
    - Deleting special day group with DeleteSpecialDayGroup command with invalid token

Please refer to Section 4.5 for Special Day Group Test Cases.

3.3.6 Schedule State

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

- DUT shall give the Schedule Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.
- DUT shall support the following command:

  - GetServices

- If DUT supports ReportingSupported as indicated by the Capabilities.StateReportingSupported, then DUT shall support the following command:
  - GetScheduleState
- The following tests are performed if DUT supports State Reporting:
  - Getting schedule state with GetScheduleState command
  - Changing schedule states with changing if iCalendar values in schedule
  - Changing schedule states with changing if iCalendar values in special dey group

Please refer to Section 0 for Schedule State Test Cases.

3.3.7 Schedule Events

The test policies specific to the test case execution of Schedule Events functional block:

- DUT shall give the Schedule Service and Event Service entry points by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.
- DUT shall support the following commands:
  - GetServices
  - GetEventProperties
- 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.
- If DUT supports ReportingSupported as indicated by the Capabilities.StateReportingSupported, then DUT shall support the following event:
  - tns1:Schedule/State/Active
- DUT shall supports the following events:
  - tns1:Configuration/Schedule/Changed
  - tns1:Configuration/Schedule/Removed
- If DUT supports Special Days as indicated by the Capabilities.SpecialDaysSupported, then DUT shall support the following events:
  - tns1:Configuration/SpecialDays/Changed
  - tns1:Configuration/SpecialDays/Removed
- The following tests are performed
  - Getting event properties with GetEventProperties command

Please refer to Section 0 for Schedule Events Test Cases.

3.3.8 Consistency

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

- DUT shall give the Schedule Service by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.
- DUT shall support the following commands:
  - GetServices
  - GetSpecialDayGroupInfo
  - GetSchedules
- The following tests are performed
  - Schedule and Special Day Group Info Consistency

Please refer to Section **Error! Reference source not found.** for Consistency Test Cases.

4 Schedule Test Cases

4.1 Capabilities

4.1.1 SCHEDULE SERVICE CAPABILITIES

Test Label: Schedule Service Capabilities Verification

Test Case ID: SCHEDULE-1-1-1

ONVIF Core Specification Coverage: ServiceCapabilities (ONVIF Schedule Service Specification), GetServiceCapabilities command (ONVIF Schedule Service Specification)

Command Under Test: GetServiceCapabilities (for Schedule Service)

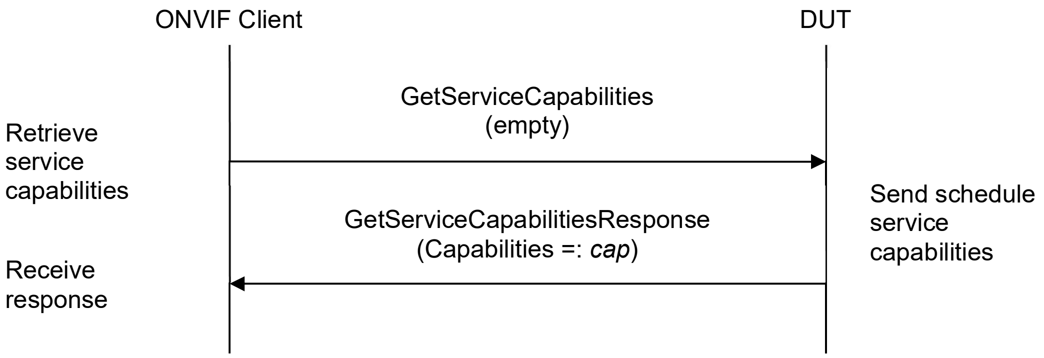
WSDL Reference: schedule.wsdl

Test Purpose: To verify DUT Schedule Service Capabilities.

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

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Retrieve service capabilities
GetServiceCapabilities (empty)
GetServiceCapabilitiesResponse (Capabilities =: *cap*)
Send schedule service capabilities
Receive response

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServiceCapabilities**.
4. The DUT responds with a **GetServiceCapabilitiesResponse** message with parameters
    - Capabilities =: *cap*

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

4.1.2 GET SERVICES AND GET SCHEDULE SERVICE CAPABILITIES CONSISTENCY

Test Label: Get Services and Schedule Service Capabilities Consistency Verification

Test Case ID: SCHEDULE-1-1-2

ONVIF Core Specification Coverage: Capability exchange (ONVIF Core Specification), ServiceCapabilities (ONVIF Schedule Service Specification), GetServiceCapabilities command (ONVIF Schedule Service Specification)

Command Under Test: GetServices, GetServiceCapabilities (for Schedule Service)

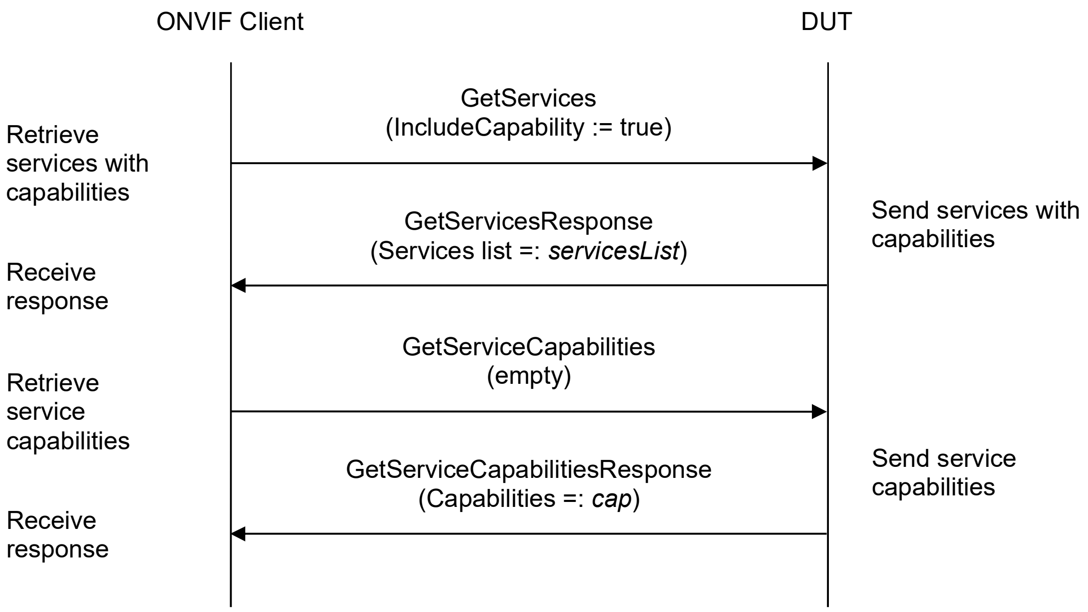
WSDL Reference: devicemgmt.wsdl, schedule.wsdl

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

Pre-requisite: None.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Retrieve services with capabilities
GetServices (IncludeCapability := true)
GetServicesResponse (Services list =: *servicesList*)
Send services with capabilities
Receive response
Retrieve service capabilities
GetServiceCapabilities (empty)
GetServiceCapabilitiesResponse (Capabilities =: *cap*)
Send service capabilities
Receive response

Test Procedure:

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

3. ONVIF Client invokes **GetServices** with parameters
    - IncludeCapability := true
4. The DUT responds with a **GetServicesResponse** message with parameters
    - Services list =: *servicesList*
5. ONVIF Client selects Service with Service.Namespace = “http://www.onvif.org/ver10/schedule/wsdl”:
    - Services list [Namespace = “http://www.onvif.org/ver10/schedule/wsdl”] =: *scheduleService*
6. ONVIF Client invokes **GetServiceCapabilities**.
7. The DUT responds with a **GetServiceCapabilitiesResponse** message with parameters
    - Capabilities =: *cap*
8. If *cap* differs from *scheduleService*.Capabilities.Capabilities, FAIL the test.

Test Result:

PASS –

The DUT passed 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:

- MaxLimit
- MaxSchedules
- MaxTimePeriodsPerDay
- MaxSpecialDayGroups
- MaxSpecialDaysInSpecialDayGroup
- MaxSpecialDaysSchedules
- ExtendedRecurrenceSupported
- SpecialDaysSupported
- StateReportingSupported

4.2 Schedule Info

4.2.1 GET SCHEDULE INFO

Test Label: Get Schedule Info Verification

Test Case ID: SCHEDULE-2-1-1

ONVIF Core Specification Coverage: ScheduleInfo (ONVIF Schedule Service Specification), Get Schedule Info command (ONVIF Schedule Service Specification)

Command Under Test: GetScheduleInfo

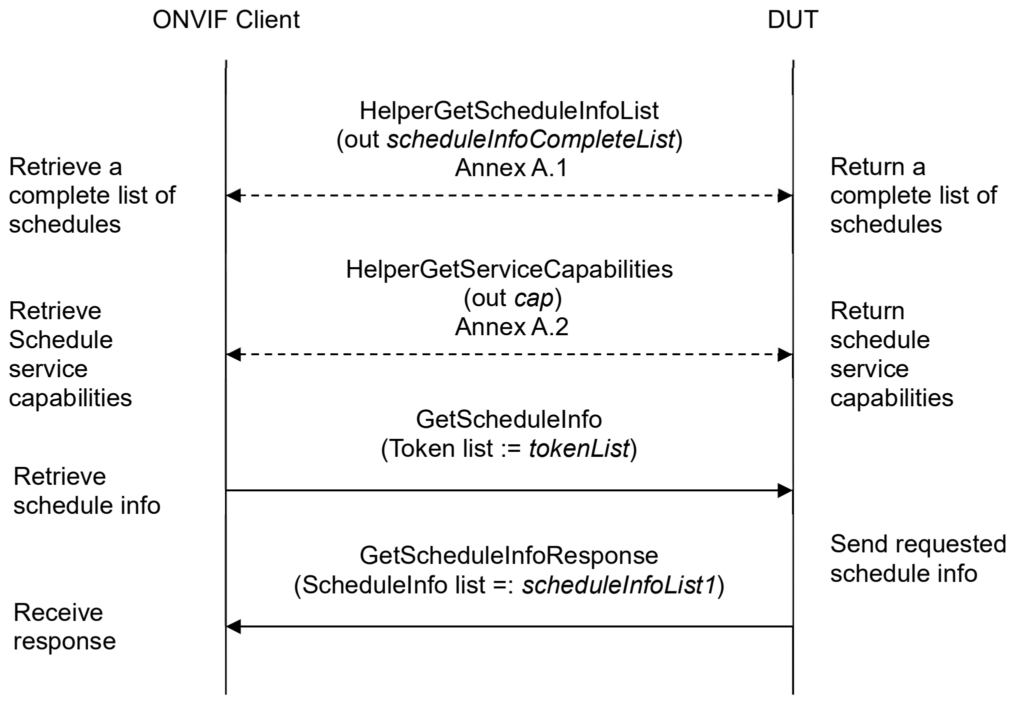
WSDL Reference: schedule.wsdl

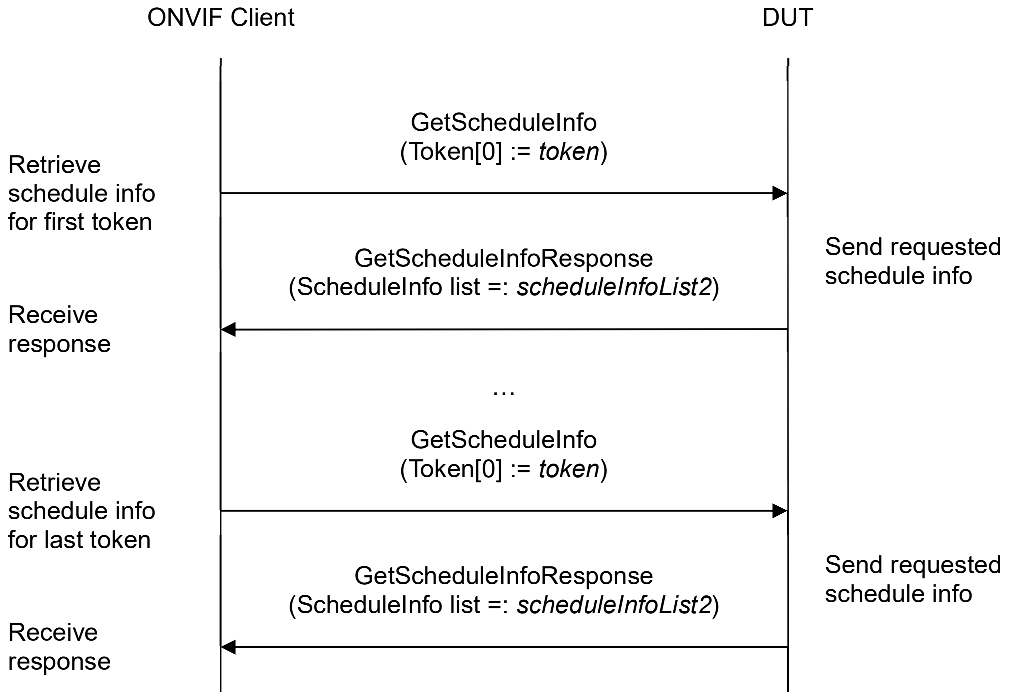
Test Purpose: To verify Get Schedule Info.

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

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetScheduleInfoList
(out *scheduleInfoCompleteList*)
Annex A.1
Retrieve a complete list of schedules
Return a complete list of schedules
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieve Schedule service capabilities
Return schedule service capabilities
GetScheduleInfo
(Token list := *tokenList*)
Retrieve schedule info
GetScheduleInfoResponse
(ScheduleInfo list =: *scheduleInfoList1*)
Send requested schedule info
Receive response


ONVIF Client
DUT
GetScheduleInfo
(Token[0] := *token*)
Retrieve schedule info for first token
GetScheduleInfoResponse
(ScheduleInfo list =: *scheduleInfoList2*)
Send requested schedule info
Receive response
…
GetScheduleInfo
(Token[0] := *token*)
Retrieve schedule info for last token
GetScheduleInfoResponse
(ScheduleInfo list =: *scheduleInfoList2*)
Send requested schedule info
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of schedule info (out *scheduleInfoCompleteList*) by following the procedure mentioned in Annex A.1.

4. If *scheduleInfoCompleteList* is empty, skip other steps.

5. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

6. Set the following:

   - *tokenList* := [subset of *scheduleInfoCompleteList*.token values with items number equal to *cap*.MaxLimit]

7. ONVIF client invokes **GetScheduleInfo** with parameters

   - Token list := *tokenList*

8. The DUT responds with **GetScheduleInfoResponse** message with parameters

   - ScheduleInfo list =: *scheduleInfoList1*

9. If *scheduleInfoList1* does not contain ScheduleInfo item for each token from *tokenList*, FAIL the test and skip other steps.

10. If *scheduleInfoList1* contains at least two ScheduleInfo items with equal token, FAIL the test and skip other steps.

11. If *scheduleInfoList1* contains other ScheduleInfo items than listed in *tokenList*, FAIL the test and skip other steps.

12. For each ScheduleInfo.token *token* from *scheduleInfoCompleteList* repeat the following steps:

    12.1. ONVIF client invokes **GetScheduleInfo** with parameters

    - Token[0] := *token*

    12.2. The DUT responds with **GetScheduleInfoResponse** message with parameters

    - ScheduleInfo list =: *scheduleInfoList2*

    12.3. If *scheduleInfoList2* does not contain only one ScheduleInfo item with token equal to *token*, FAIL the test and skip other steps.

    12.4. If *scheduleInfoList2*[0] item is not equal to *scheduleInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

Note: If number of items in *scheduleInfoCompleteList* is less than *cap*.MaxLimit, then all *scheduleInfoCompleteList*.Token items shall be used for the step 6.

Note: The following fields are compared at step 12.4:

- ScheduleInfo:
    - token
    - Name
    - Description

4.2.2 GET SCHEDULE INFO LIST - LIMIT

Test Label: Get Schedule Info List Verification with Limit

Test Case ID: SCHEDULE-2-1-2

ONVIF Core Specification Coverage: ScheduleInfo (ONVIF Schedule Service Specification), GetScheduleInfoList command (ONVIF Schedule Service Specification)

Command Under Test: GetScheduleInfoList

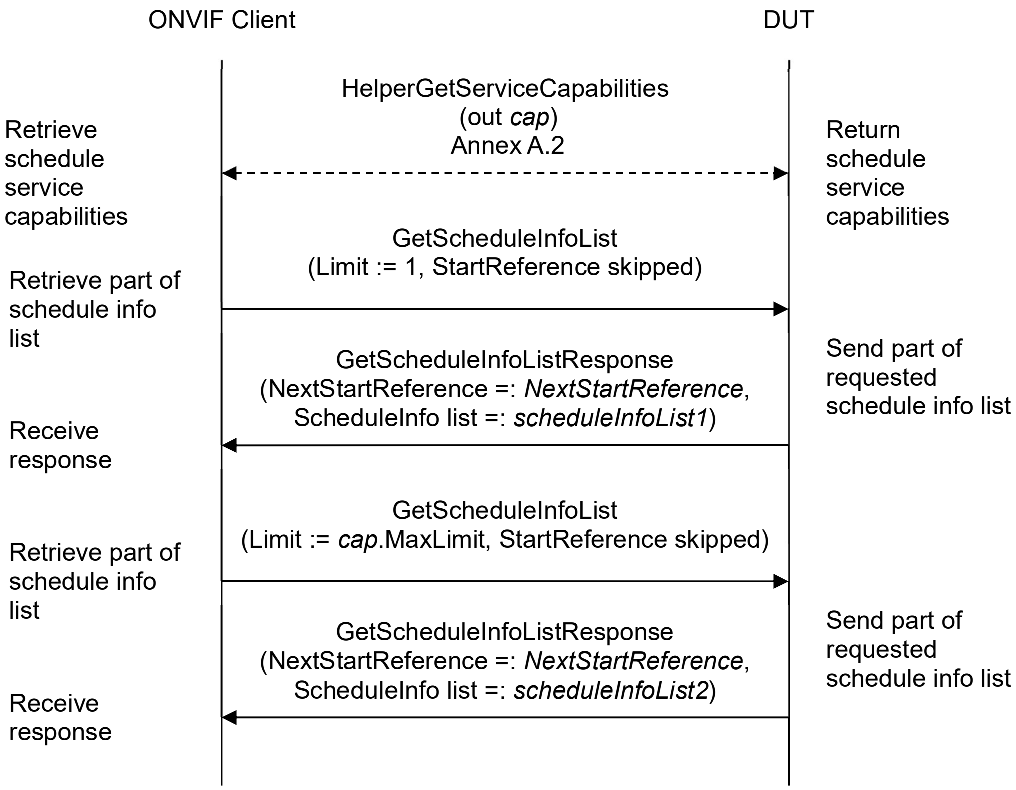
WSDL Reference: schedule.wsdl

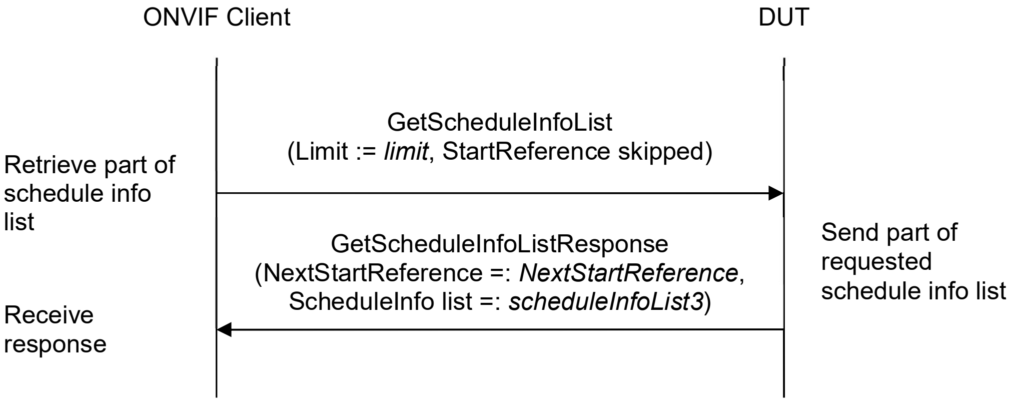
Test Purpose: To verify Get Schedule Info List using Limit.

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

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieve schedule service capabilities
Return schedule service capabilities
GetScheduleInfoList
(Limit := 1, StartReference skipped)
Retrieve part of schedule info list
GetScheduleInfoListResponse
(NextStartReference =: *NextStartReference*,
ScheduleInfo list =: *scheduleInfoList1*)
Send part of requested schedule info list
Receive response
GetScheduleInfoList
(Limit := *cap*.MaxLimit, StartReference skipped)
Retrieve part of schedule info list
GetScheduleInfoListResponse
(NextStartReference =: *NextStartReference*,
ScheduleInfo list =: *scheduleInfoList2*)
Send part of requested schedule info list
Receive response


ONVIF Client
DUT
Retrieve part of schedule info list
GetScheduleInfoList (Limit := *limit*, StartReference skipped)
GetScheduleInfoListResponse (NextStartReference =: *NextStartReference*, ScheduleInfo list =: *scheduleInfoList3*)
Send part of requested schedule info list
Receive response

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.
4. ONVIF client invokes **GetScheduleInfoList** with parameters
   - Limit := 1
   - StartReference skipped
5. The DUT responds with **GetScheduleInfoListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - ScheduleInfo list =: *scheduleInfoList1*
6. If *scheduleInfoList1* contains more ScheduleInfo items than 1, FAIL the test and skip other steps.
7. If *cap*.MaxLimit is equal to 1, skip other steps.
8. ONVIF client invokes **GetScheduleInfoList** with parameters
   - Limit := *cap*.MaxLimit
   - StartReference skipped
9. The DUT responds with **GetScheduleInfoListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - ScheduleInfo list =: *scheduleInfoList2*
10. If *scheduleInfoList2* contains more ScheduleInfo items than *cap*.MaxLimit, FAIL the test and skip other steps.

11. If *cap*.MaxLimit is equal to 2, skip other steps.

12. Set the following:

    - *limit* := [number between 1 and *cap*.MaxLimit]

13. ONVIF client invokes **GetScheduleInfoList** with parameters

    - Limit := *limit*
    - StartReference skipped

14. The DUT responds with **GetScheduleInfoListResponse** message with parameters

    - NextStartReference =: *nextStartReference*
    - ScheduleInfo list =: *scheduleInfoList3*

15. If *scheduleInfoList3* contains more ScheduleInfo items than *limit*, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

4.2.3 GET SCHEDULE INFO LIST - START REFERENCE AND LIMIT

Test Label: Get Schedule Info List Verification with Start Reference and Limit

Test Case ID: SCHEDULE-2-1-3

ONVIF Core Specification Coverage: ScheduleInfo (ONVIF Schedule Service Specification), GetScheduleInfoList command (ONVIF Schedule Service Specification)

Command Under Test: GetScheduleInfoList

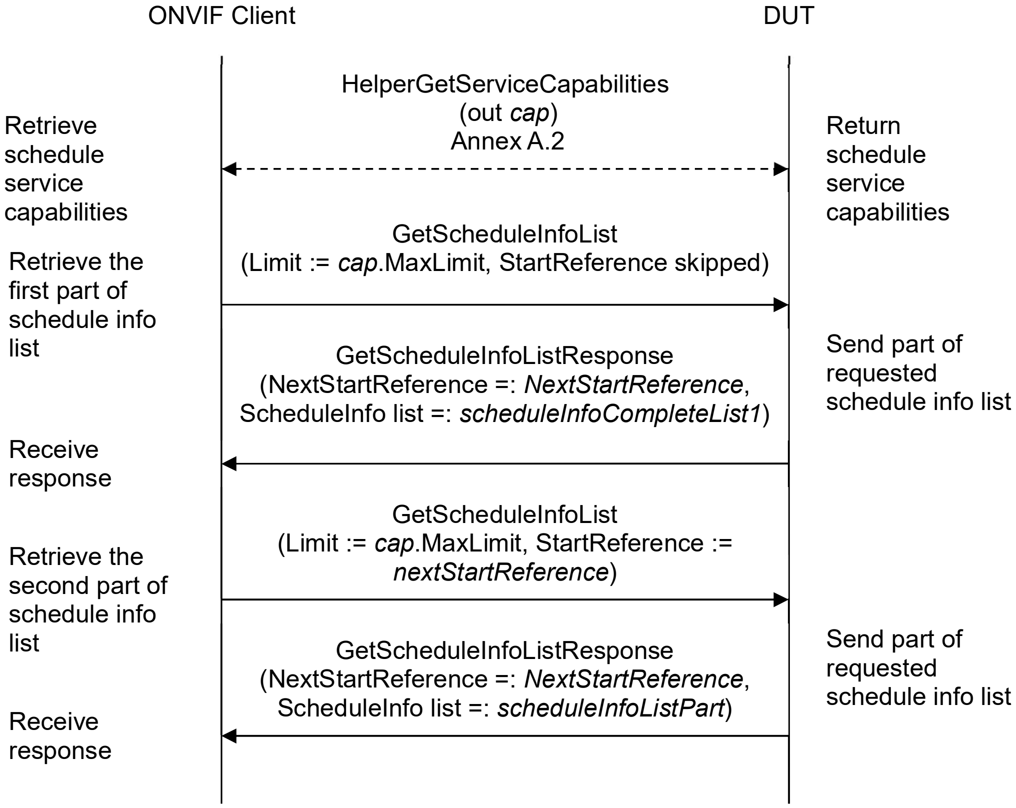
WSDL Reference: schedule.wsdl

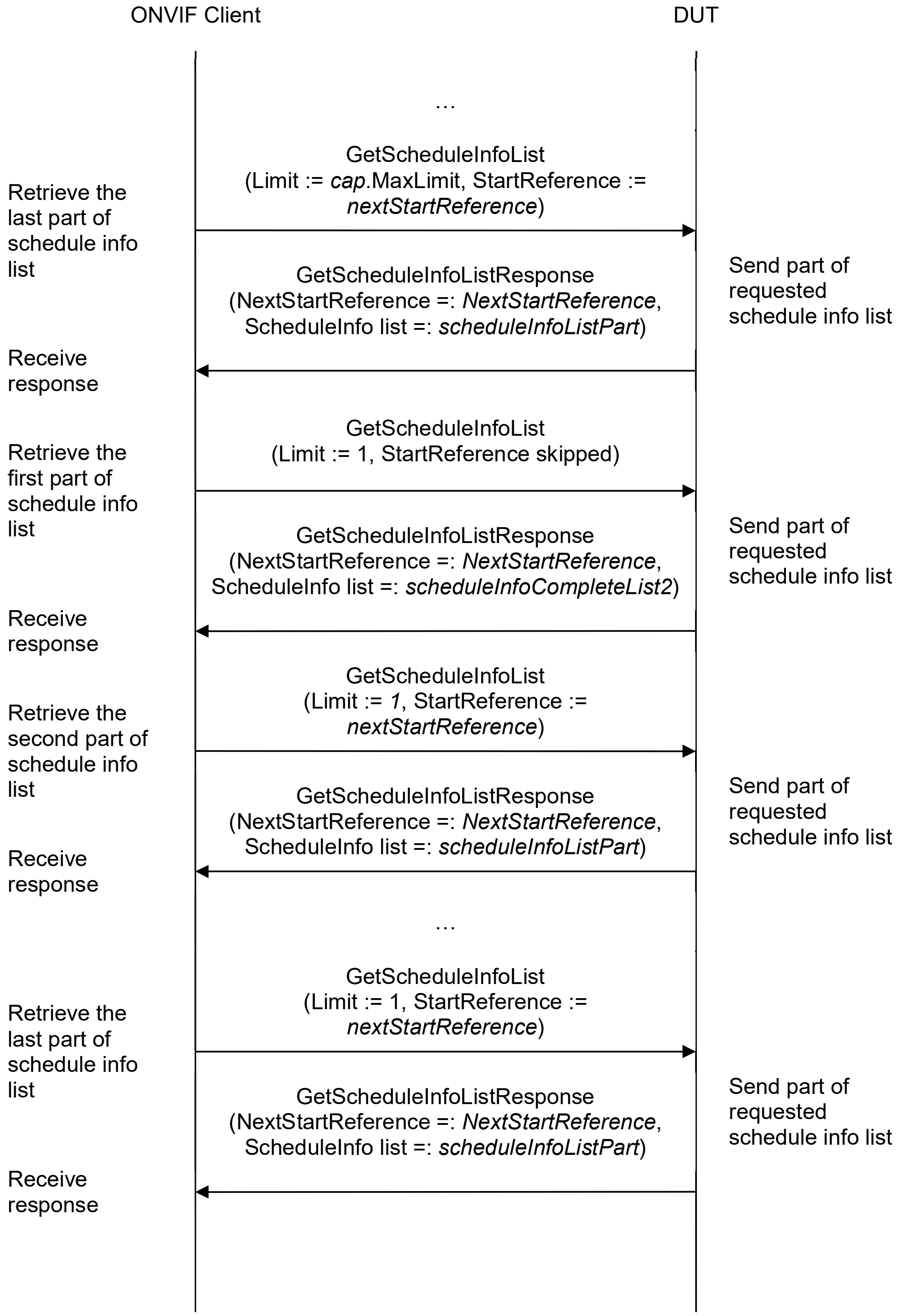
Test Purpose: To verify Get Schedule Info List using StartReference and Limit.

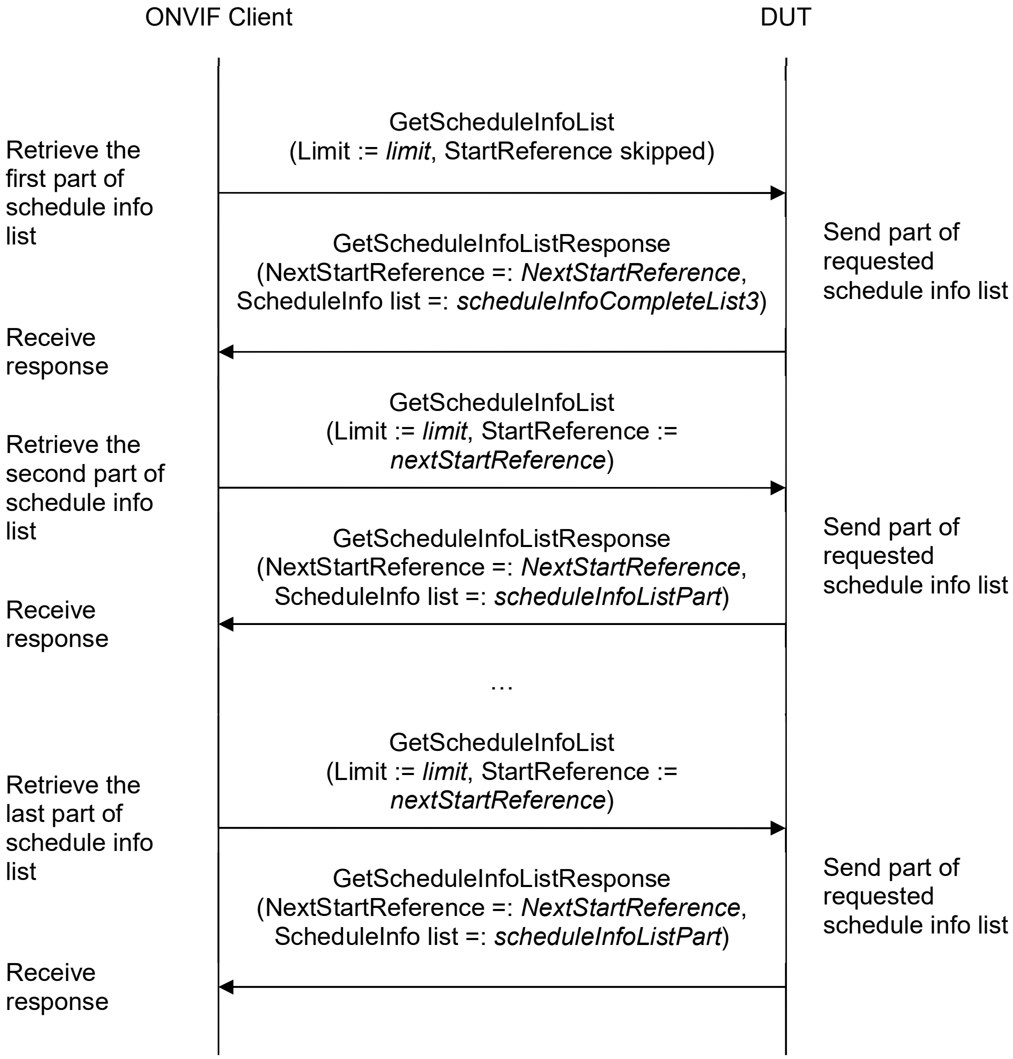
Pre-requisite: Schedule Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Retrieve schedule service capabilities
HelperGetServiceCapabilities (out *cap*) Annex A.2
Return schedule service capabilities
Retrieve the first part of schedule info list
GetScheduleInfoList (Limit := *cap*.MaxLimit, StartReference skipped)
GetScheduleInfoListResponse (NextStartReference =: *NextStartReference*, ScheduleInfo list =: *scheduleInfoCompleteList1*)
Send part of requested schedule info list
Receive response
Retrieve the second part of schedule info list
GetScheduleInfoList (Limit := *cap*.MaxLimit, StartReference := *nextStartReference*)
GetScheduleInfoListResponse (NextStartReference =: *NextStartReference*, ScheduleInfo list =: *scheduleInfoListPart*)
Send part of requested schedule info list
Receive response


ONVIF Client
DUT
…
GetScheduleInfoList (Limit := *cap*.MaxLimit, StartReference := *nextStartReference*)
Retrieve the last part of schedule info list
GetScheduleInfoListResponse (NextStartReference =: *NextStartReference*, ScheduleInfo list =: *scheduleInfoListPart*)
Send part of requested schedule info list
Receive response
GetScheduleInfoList (Limit := 1, StartReference skipped)
Retrieve the first part of schedule info list
GetScheduleInfoListResponse (NextStartReference =: *NextStartReference*, ScheduleInfo list =: *scheduleInfoCompleteList2*)
Send part of requested schedule info list
Receive response
GetScheduleInfoList (Limit := *1*, StartReference := *nextStartReference*)
Retrieve the second part of schedule info list
GetScheduleInfoListResponse (NextStartReference =: *NextStartReference*, ScheduleInfo list =: *scheduleInfoListPart*)
Send part of requested schedule info list
Receive response
…
GetScheduleInfoList (Limit := 1, StartReference := *nextStartReference*)
Retrieve the last part of schedule info list
GetScheduleInfoListResponse (NextStartReference =: *NextStartReference*, ScheduleInfo list =: *scheduleInfoListPart*)
Send part of requested schedule info list
Receive response


ONVIF Client
DUT
Retrieve the first part of schedule info list
GetScheduleInfoList
(Limit := *limit*, StartReference skipped)
GetScheduleInfoListResponse
(NextStartReference =: *NextStartReference*,
ScheduleInfo list =: *scheduleInfoCompleteList3*)
Send part of requested schedule info list
Receive response
Retrieve the second part of schedule info list
GetScheduleInfoList
(Limit := *limit*, StartReference :=
nextStartReference)
GetScheduleInfoListResponse
(NextStartReference =: *NextStartReference*,
ScheduleInfo list =: *scheduleInfoListPart*)
Send part of requested schedule info list
Receive response
…
Retrieve the last part of schedule info list
GetScheduleInfoList
(Limit := *limit*, StartReference :=
nextStartReference)
GetScheduleInfoListResponse
(NextStartReference =: *NextStartReference*,
ScheduleInfo list =: *scheduleInfoListPart*)
Send part of requested schedule info list
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

4. ONVIF client invokes **GetScheduleInfoList** with parameters

   - Limit := *cap*.MaxLimit

   - StartReference skipped

5. The DUT responds with **GetScheduleInfoListResponse** message with parameters

- NextStartReference =: *nextStartReference*
- ScheduleInfo list =: *scheduleInfoCompleteList1*

6. If *scheduleInfoCompleteList1* contains more ScheduleInfo items than *cap*.MaxLimit, FAIL the test and skip other steps.

7. Until *nextStartReference* is not null, repeat the following steps:

   7.1. ONVIF client invokes **GetScheduleInfoList** with parameters

   - Limit := *cap*.MaxLimit
   - StartReference := *nextStartReference*

   7.2. The DUT responds with **GetScheduleInfoListResponse** message with parameters

   - NextStartReference =: *nextStartReference*
   - ScheduleInfo list =: *scheduleInfoListPart*

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

   7.4. Set the following:

   - *scheduleInfoCompleteList1* := *scheduleInfoCompleteList1* + *scheduleInfoListPart*

8. If *scheduleInfoCompleteList1* contains at least two ScheduleInfo item with equal token, FAIL the test and skip other steps.

9. If *cap*.MaxLimit is equal to 1, skip other steps.

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

- Limit := 1
- StartReference skipped

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

- NextStartReference =: *nextStartReference*
- ScheduleInfo list =: *scheduleInfoCompleteList2*

12. If *scheduleInfoCompleteList2* contains more ScheduleInfo items than 1, FAIL the test and skip other steps.

13. Until *nextStartReference* is not null, repeat the following steps:

   13.1. ONVIF client invokes **GetScheduleInfoList** with parameters

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

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

- NextStartReference =: *nextStartReference*
- ScheduleInfo list =: *scheduleInfoListPart*

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

13.4. Set the following:

- *scheduleInfoCompleteList2* := *scheduleInfoCompleteList2* + *scheduleInfoListPart*

14. If *scheduleInfoCompleteList2* contains at least two ScheduleInfo item with equal token, FAIL the test and skip other steps.

15. If *scheduleInfoCompleteList2* does not contain all schedules from *scheduleInfoCompleteList1*, FAIL the test and skip other steps.

16. If *scheduleInfoCompleteList2* contains schedules other than schedules from *scheduleInfoCompleteList1*, FAIL the test and skip other steps.

17. If *cap*.MaxLimit is equal to 2, skip other steps.

18. Set the following:

- *limit* := [number between 1 and *cap*.MaxLimit]

19. ONVIF client invokes **GetScheduleInfoList** with parameters

- Limit := *limit*
- StartReference skipped

20. The DUT responds with **GetScheduleInfoListResponse** message with parameters

- NextStartReference =: *nextStartReference*
- ScheduleInfo list =: *scheduleInfoCompleteList3*

21. If *scheduleInfoCompleteList3* contains more ScheduleInfo items than *limit*, FAIL the test and skip other steps.

22. Until *nextStartReference* is not null, repeat the following steps:

22.1. ONVIF client invokes **GetScheduleInfoList** with parameters

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

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

- NextStartReference =: *nextStartReference*
- ScheduleInfo list =: *scheduleInfoListPart*

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

22.4. Set the following:

- *scheduleInfoCompleteList3* := *scheduleInfoCompleteList3* + *scheduleInfoListPart*

23. If *scheduleInfoCompleteList3* contains at least two ScheduleInfo item with equal token, FAIL the test and skip other steps.

24. If *scheduleInfoCompleteList3* does not contain all schedules from *scheduleInfoCompleteList1*, FAIL the test and skip other steps.

25. If *scheduleInfoCompleteList3* contains schedules other than schedules from *scheduleInfoCompleteList1*, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

4.2.4 GET SCHEDULE INFO LIST - NO LIMIT

Test Label: Get Schedule Info List Verification without Limit

Test Case ID: SCHEDULE-2-1-4

ONVIF Core Specification Coverage: ScheduleInfo (ONVIF Schedule Service Specification), GetScheduleInfoList command (ONVIF Schedule Service Specification)

Command Under Test: GetScheduleInfoList

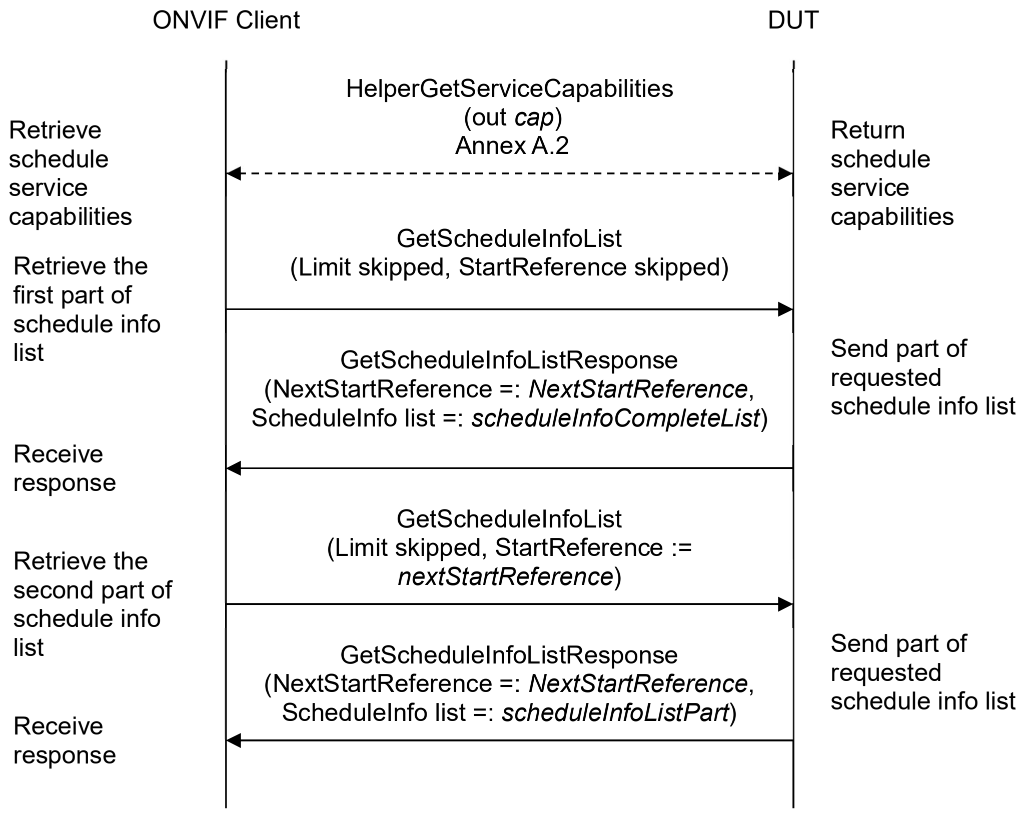
WSDL Reference: schedule.wsdl

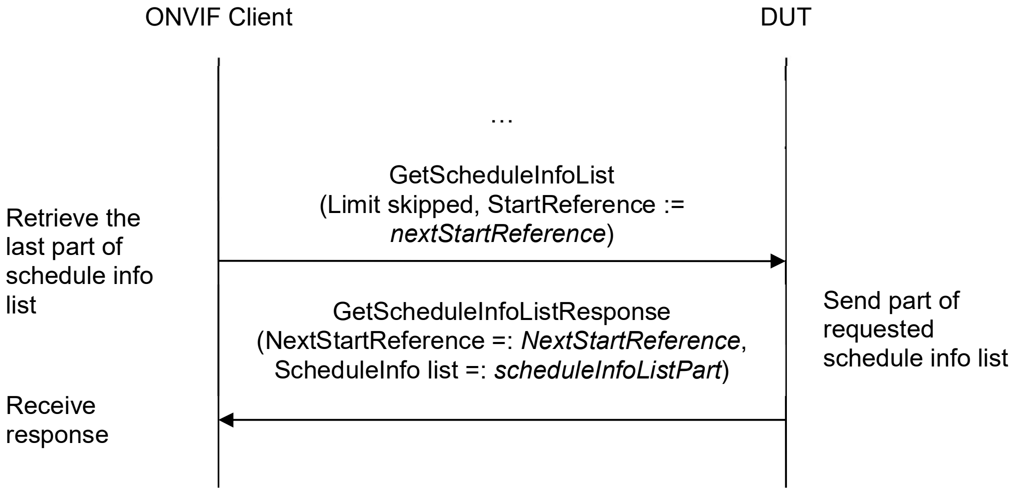
Test Purpose: To verify Get Schedule Info List without using Limit.

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

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieve schedule service capabilities
Return schedule service capabilities
GetScheduleInfoList
(Limit skipped, StartReference skipped)
Retrieve the first part of schedule info list
GetScheduleInfoListResponse
(NextStartReference =: *NextStartReference*,
ScheduleInfo list =: *scheduleInfoCompleteList*)
Send part of requested schedule info list
Receive response
GetScheduleInfoList
(Limit skipped, StartReference :=
nextStartReference)
Retrieve the second part of schedule info list
GetScheduleInfoListResponse
(NextStartReference =: *NextStartReference*,
ScheduleInfo list =: *scheduleInfoListPart*)
Send part of requested schedule info list
Receive response


ONVIF Client
DUT
…
GetScheduleInfoList
(Limit skipped, StartReference :=
nextStartReference)
Retrieve the last part of schedule info list
GetScheduleInfoListResponse
(NextStartReference =: *NextStartReference*,
ScheduleInfo list =: *scheduleInfoListPart*)
Send part of requested schedule info list
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

4. ONVIF client invokes **GetScheduleInfoList** with parameters

   - Limit skipped

   - StartReference skipped

5. The DUT responds with **GetScheduleInfoListResponse** message with parameters

   - NextStartReference =: *nextStartReference*

   - ScheduleInfo list =: *scheduleInfoCompleteList*

6. If *scheduleInfoCompleteList* contains more ScheduleInfo items than *cap*.MaxLimit, FAIL the test and skip other steps.

7. Until *nextStartReference* is not null, repeat the following steps:

   7.1. ONVIF client invokes **GetScheduleInfoList** with parameters

   - Limit skipped

   - StartReference := *nextStartReference*

   7.2. The DUT responds with **GetScheduleInfoListResponse** message with parameters

   - NextStartReference =: *nextStartReference*

- ScheduleInfo list =: *scheduleInfoListPart*

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

7.4. Set the following:

- *scheduleInfoCompleteList* := *scheduleInfoCompleteList* + *scheduleInfoListPart*

8. If *scheduleInfoCompleteList* contains at least two ScheduleInfo items with equal token, FAIL the test.

9. If *scheduleInfoCompleteList* contains more ScheduleInfo items than *cap*.MaxSchedules, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

4.2.5 GET SCHEDULE INFO WITH INVALID TOKEN

Test Label: Get Schedule Info with Invalid Token Verification

Test Case ID: SCHEDULE-2-1-5

ONVIF Core Specification Coverage: ScheduleInfo (ONVIF Schedule Service Specification), GetScheduleInfo command (ONVIF Schedule Service Specification)

Command Under Test: GetScheduleInfo

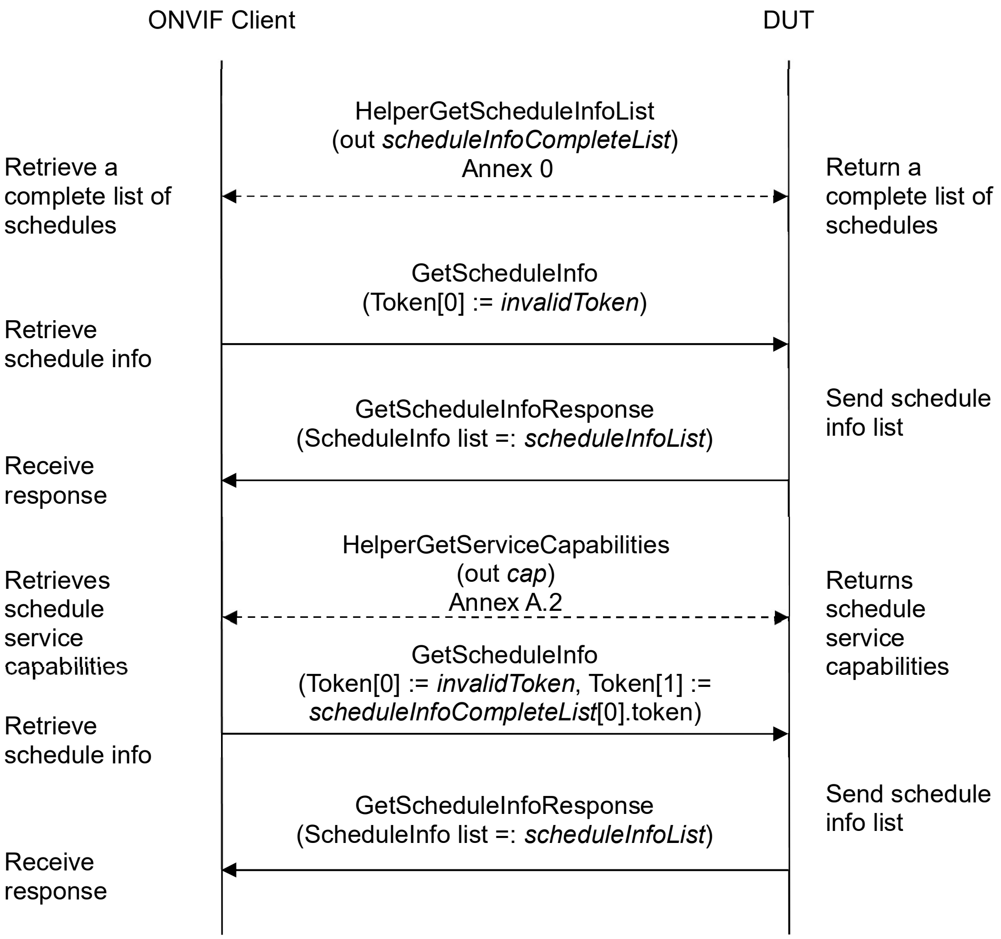
WSDL Reference: schedule.wsdl

Test Purpose: To verify Get Schedule Info with invalid token.

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

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetScheduleInfoList
(out *scheduleInfoCompleteList*)
Annex 0
Retrieve a complete list of schedules
Return a complete list of schedules
GetScheduleInfo
(Token[0] := *invalidToken*)
Retrieve schedule info
GetScheduleInfoResponse
(ScheduleInfo list =: *scheduleInfoList*)
Send schedule info list
Receive response
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieves schedule service capabilities
Returns schedule service capabilities
GetScheduleInfo
(Token[0] := *invalidToken*, Token[1] := *scheduleInfoCompleteList*[0].token)
Retrieve schedule info
GetScheduleInfoResponse
(ScheduleInfo list =: *scheduleInfoList*)
Send schedule info list
Receive response

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves a complete list of schedule info (out *scheduleInfoCompleteList*) by following the procedure mentioned in Annex 0.
4. Set the following:
   - *invalidToken* := value not equal to any *scheduleInfoCompleteList*.token
5. ONVIF client invokes **GetScheduleInfo** with parameters
   - Token list := *invalidToken*
6. The DUT responds with **GetScheduleInfoResponse** message with parameters
   - ScheduleInfo list =: *scheduleInfoList*
7. If *scheduleInfoList* is not empty, FAIL the test.
8. If *scheduleInfoCompleteList* is empty, skip other steps.
9. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.
10. If *cap*.MaxLimit is less than 2, skip other steps.
11. ONVIF client invokes **GetScheduleInfo** with parameters
    - Token[0]:= *invalidToken*
    - Token[1]:= *scheduleInfoCompleteList*[0].token
12. The DUT responds with **GetScheduleInfoResponse** message with parameters
    - ScheduleInfo list =: *scheduleInfoList*
13. If *scheduleInfoList* is empty, FAIL the test.
14. If *scheduleInfoList* contains more than one item, FAIL the test.
15. If *scheduleInfoList*[0].token is not equal to *scheduleInfoCompleteList*[0].token, FAIL the test.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

4.2.6 GET SCHEDULE INFO - TOO MANY ITEMS

Test Label: Get Schedule Info - number of requested items is greater than MaxLimit

Test Case ID: SCHEDULE-2-1-6

ONVIF Core Specification Coverage: ScheduleInfo (ONVIF Schedule Service Specification), GetScheduleInfo command (ONVIF Schedule Service Specification)

Command Under Test: GetScheduleInfo

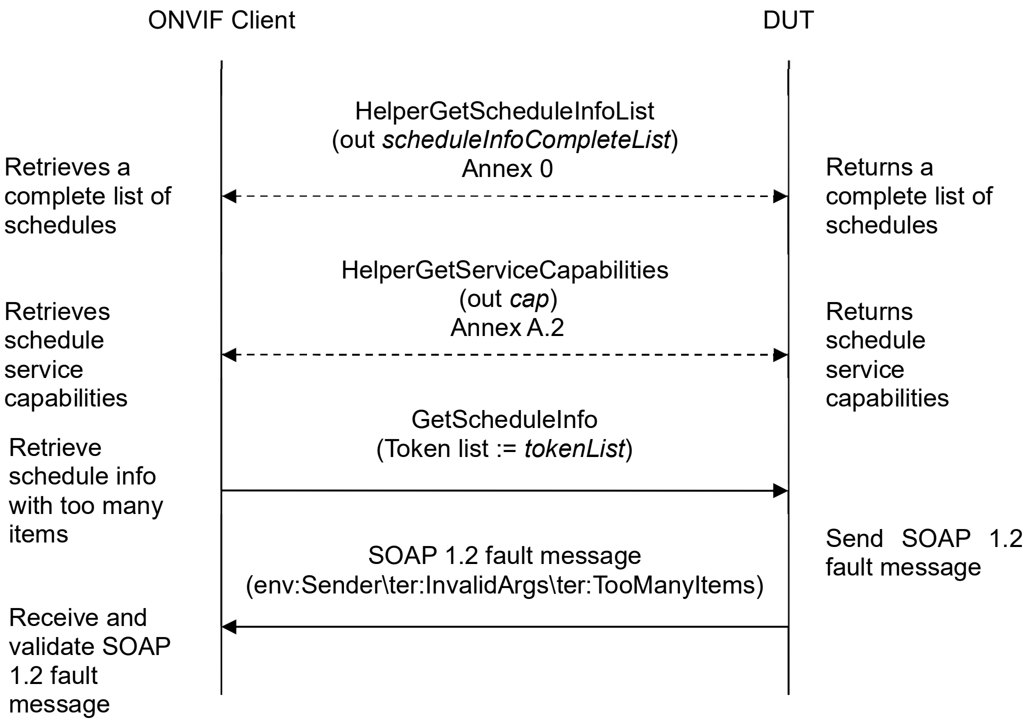
WSDL Reference: schedule.wsdl

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

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

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetScheduleInfoList (out *scheduleInfoCompleteList*) Annex 0
Retrieves a complete list of schedules
Returns a complete list of schedules
HelperGetServiceCapabilities (out *cap*) Annex A.2
Retrieves schedule service capabilities
Returns schedule service capabilities
GetScheduleInfo (Token list := *tokenList*)
Retrieve schedule info with too many items
SOAP 1.2 fault message (env:Sender\ter:InvalidArgs\ter:TooManyItems)
Send SOAP 1.2 fault message
Receive and validate SOAP 1.2 fault message

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of schedule info (out *scheduleInfoCompleteList*) by following the procedure mentioned in Annex 0.

4. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

5. If *scheduleInfoCompleteList*.token items number is less than *cap*.MaxLimit or equal to *cap*.MaxLimit, skip other steps.

6. Set the following:

   - *tokenList* := [subset of *scheduleInfoCompleteList*.token values with items number equal to *cap*.MaxLimit + 1]

7. ONVIF client invokes **GetScheduleInfo** with parameters

   - Token list := *tokenList*

8. The DUT returns **env:Sender\ter:InvalidArgs\ter:TooManyItems** SOAP 1.2 fault.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send env:Sender\ter:InvalidArgs\ter:TooManyItems SOAP 1.2 fault.

4.3 Schedule

4.3.1 GET SCHEDULES

Test Label: Get Schedules Verification

Test Case ID: SCHEDULE-3-1-1

ONVIF Core Specification Coverage: Schedule (ONVIF Schedule Service Specification), GetSchedules command (ONVIF Schedule Service Specification)

Command Under Test: GetSchedules

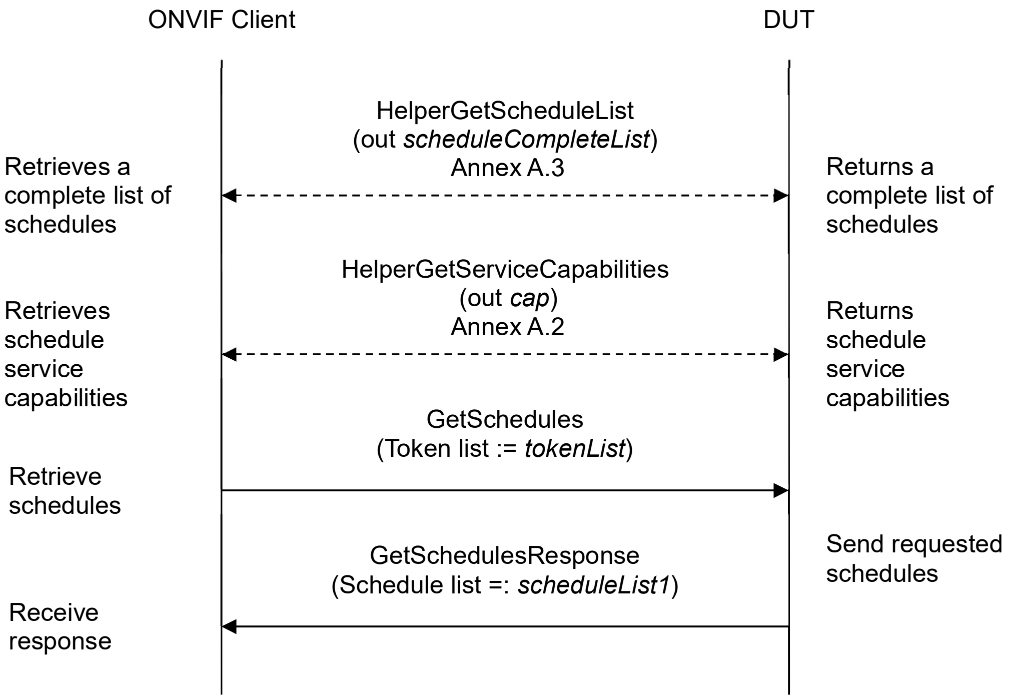
WSDL Reference: schedule.wsdl

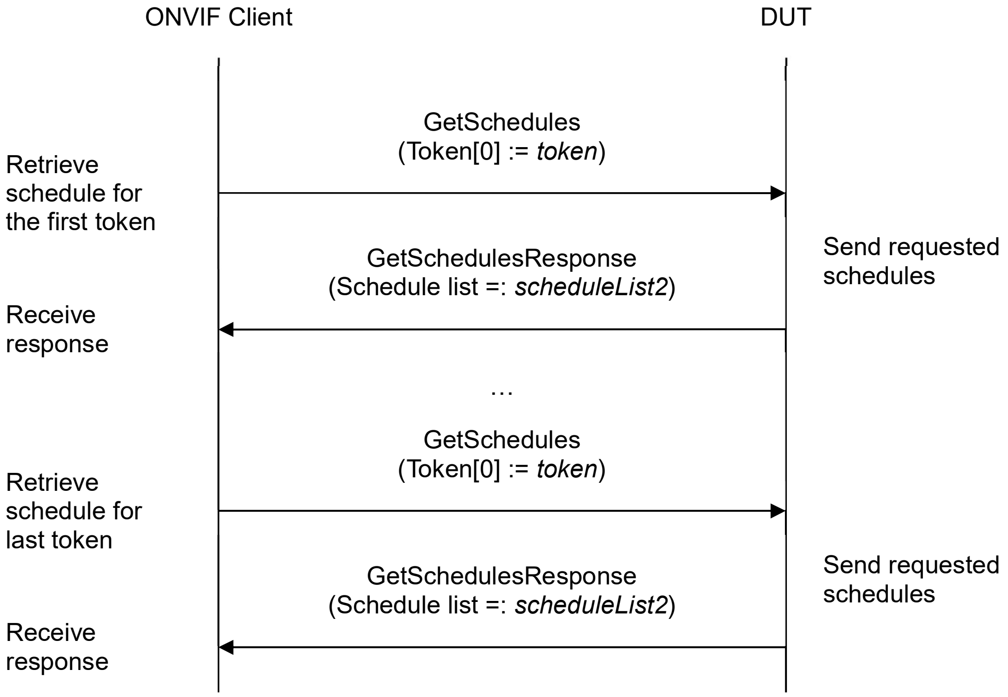
Test Purpose: To verify Get Schedule.

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

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetScheduleList
(out *scheduleCompleteList*)
Annex A.3
Retrieves a complete list of schedules
Returns a complete list of schedules
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieves schedule service capabilities
Returns schedule service capabilities
GetSchedules
(Token list := *tokenList*)
Retrieve schedules
GetSchedulesResponse
(Schedule list =: *scheduleList1*)
Send requested schedules
Receive response



Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of schedules (out *scheduleCompleteList*) by following the procedure mentioned in Annex A.3.

4. If *scheduleCompleteList* is empty, skip other steps.

5. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

6. Set the following:

   - *tokenList* := [subset of *scheduleCompleteList*.token values with items number equal to *cap*.MaxLimit]

7. ONVIF client invokes **GetSchedules** with parameters

   - Token list := *tokenList*

8. The DUT responds with **GetSchedulesResponse** message with parameters

   - Schedule list =: *scheduleList1*

9. If *scheduleList1* does not contain Schedule item for each token from *tokenList*, FAIL the test and skip other steps.

10. If *scheduleList1* contains at least two Schedule items with equal token, FAIL the test and skip other steps.

11. If *scheduleList1* contains other Schedule items than listed in *tokenList*, FAIL the test and skip other steps.

12. For each Schedule.token *token* from *scheduleCompleteList* repeat the following steps:

    12.1. ONVIF client invokes **GetSchedules** with parameters

    - Token[0] := *token*

    12.2. The DUT responds with **GetSchedulesResponse** message with parameters

    - Schedule list =: *scheduleList2*

    12.3. If *scheduleList2* does not contain only one Schedule item with token equal to *token*, FAIL the test and skip other steps.

    12.4. If *scheduleList2*[0] item does not have equal field values to *scheduleCompleteList*[token = *token*] item, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

Note: If number of items in *scheduleCompleteList* is less than *cap*.MaxLimit, then all *scheduleCompleteList*.Token items shall be used for the step 6.

Note: The following fields are compared at step 12.4:

- Schedule:
  - token
  - Name
  - Description
  - Standard
  - SpecialDays
    - GroupToken
    - TimeRange
      - From
      - Until

4.3.2 GET SCHEDULE LIST - LIMIT

Test Label: Get Schedule List Verification with Limit

Test Case ID: SCHEDULE-3-1-2

ONVIF Core Specification Coverage: Schedule (ONVIF Schedule Service Specification), GetScheduleList command (ONVIF Schedule Service Specification)

Command Under Test: GetScheduleList

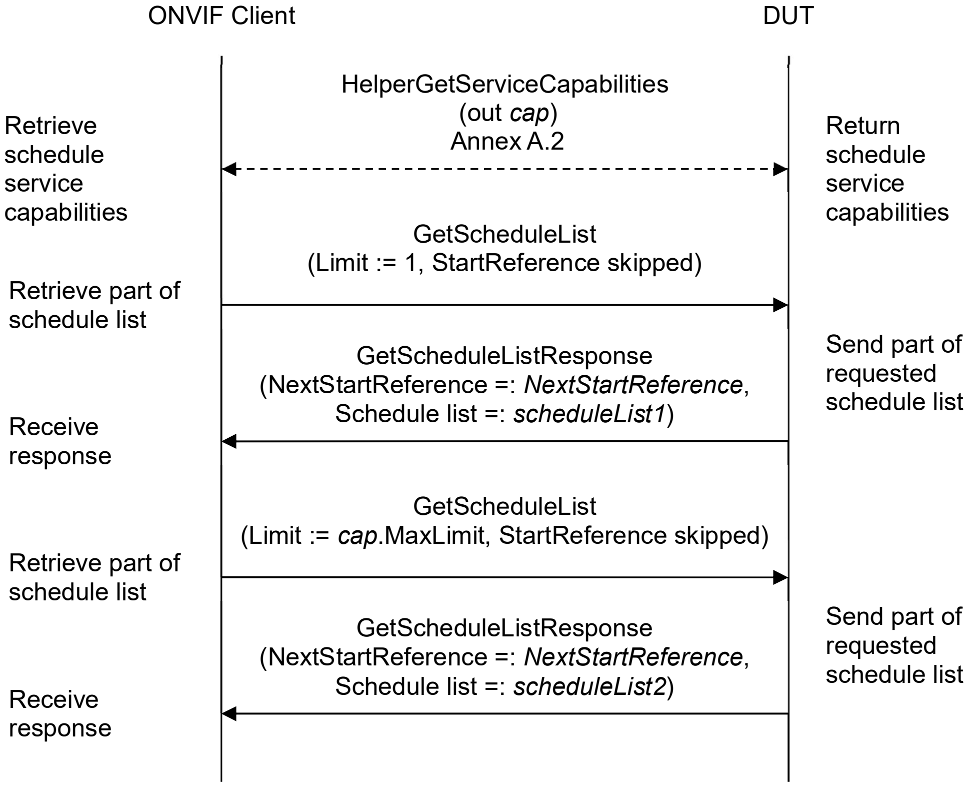
WSDL Reference: schedule.wsdl

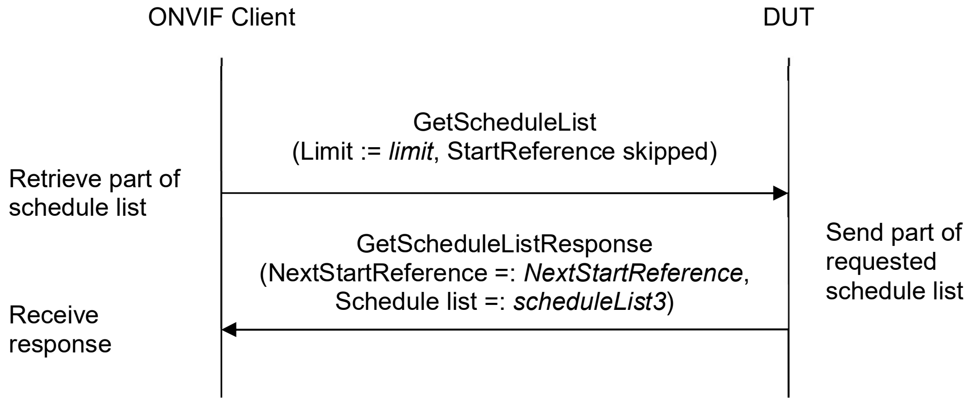
Test Purpose: To verify Get Schedule List using Limit.

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

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieve schedule service capabilities
Return schedule service capabilities
GetScheduleList
(Limit := 1, StartReference skipped)
Retrieve part of schedule list
GetScheduleListResponse
(NextStartReference =: *NextStartReference*,
Schedule list =: *scheduleList1*)
Send part of requested schedule list
Receive response
GetScheduleList
(Limit := *cap*.MaxLimit, StartReference skipped)
Retrieve part of schedule list
GetScheduleListResponse
(NextStartReference =: *NextStartReference*,
Schedule list =: *scheduleList2*)
Send part of requested schedule list
Receive response


ONVIF Client
DUT
GetScheduleList
(Limit := *limit*, StartReference skipped)
Retrieve part of schedule list
GetScheduleListResponse
(NextStartReference =: *NextStartReference*,
Schedule list =: *scheduleList3*)
Send part of requested schedule list
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

4. ONVIF client invokes **GetScheduleList** with parameters

   - Limit := 1

   - StartReference skipped

5. The DUT responds with **GetScheduleListResponse** message with parameters

   - NextStartReference =: *nextStartReference*

   - Schedule list =: *scheduleList1*

6. If *scheduleList1* contains more Schedule items than 1, FAIL the test and skip other steps.

7. If *cap*.MaxLimit is equal to 1, skip other steps.

8. ONVIF client invokes **GetScheduleList** with parameters

   - Limit := *cap*.MaxLimit

   - StartReference skipped

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

   - NextStartReference =: *nextStartReference*

   - Schedule list =: *scheduleList2*

10. If *scheduleList2* contains more Schedule items than *cap*.MaxLimit, FAIL the test and skip other steps.

11. If *cap*.MaxLimit is equal to 2, skip other steps.

12. Set the following:

    - *limit* := [number between 1 and *cap*.MaxLimit]

13. ONVIF client invokes **GetScheduleList** with parameters

    - Limit := *limit*
    - StartReference skipped

14. The DUT responds with **GetScheduleListResponse** message with parameters

    - NextStartReference =: *nextStartReference*
    - Schedule list =: *scheduleList3*

15. If *scheduleList3* contains more Schedule items than *limit*, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

4.3.3 GET SCHEDULE LIST - START REFERENCE AND LIMIT

Test Label: Get Schedule List Verification with Start Reference and Limit

Test Case ID: SCHEDULE-3-1-3

ONVIF Core Specification Coverage: ScheduleInfo (ONVIF Schedule Service Specification), Schedule (ONVIF Schedule Service Specification), GetScheduleList command (ONVIF Schedule Service Specification)

Command Under Test: GetScheduleList

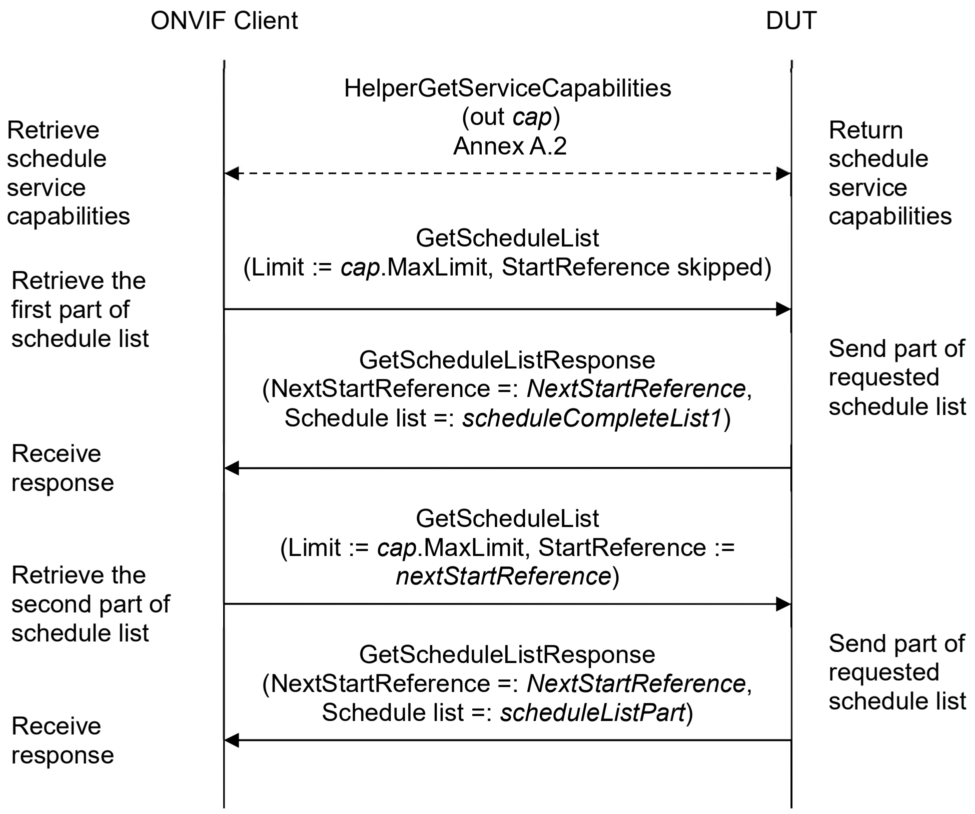
WSDL Reference: schedule.wsdl

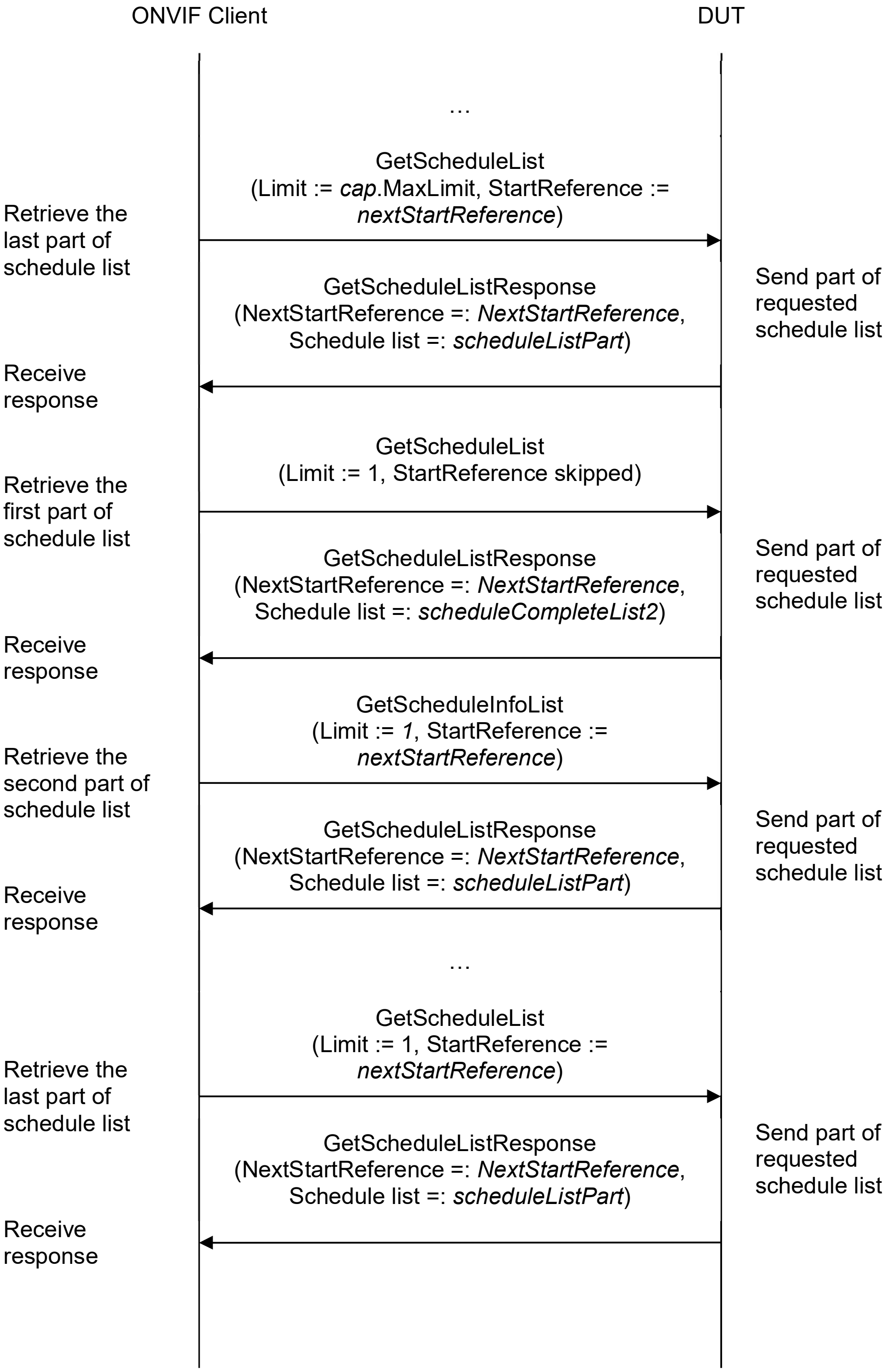
Test Purpose: To verify Get Schedule List using StartReference and Limit.

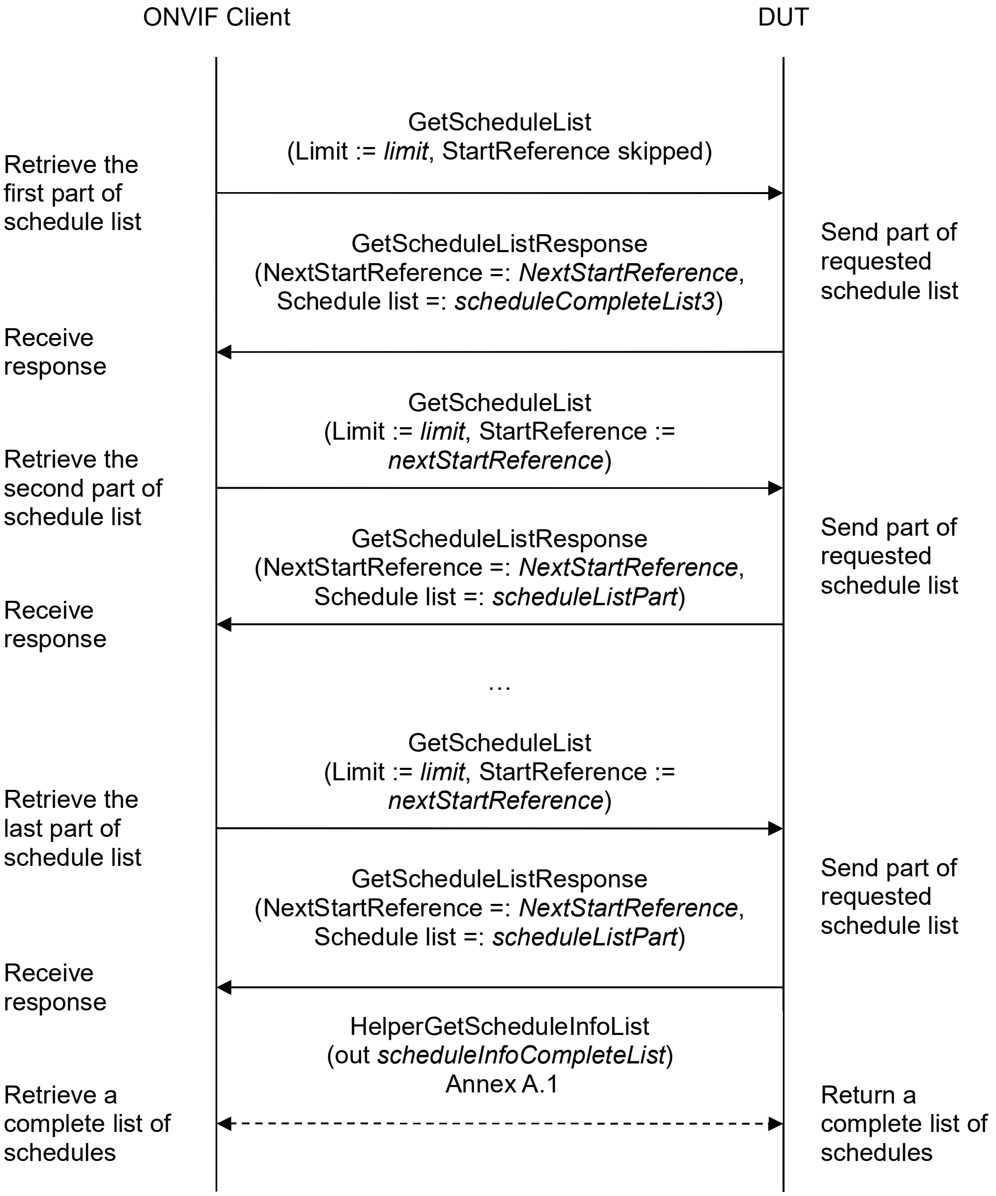
Pre-requisite: Schedule Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieve schedule service capabilities
Return schedule service capabilities
GetScheduleList
(Limit := *cap*.MaxLimit, StartReference skipped)
Retrieve the first part of schedule list
GetScheduleListResponse
(NextStartReference =: *NextStartReference*,
Schedule list =: *scheduleCompleteList1*)
Send part of requested schedule list
Receive response
GetScheduleList
(Limit := *cap*.MaxLimit, StartReference :=
nextStartReference)
Retrieve the second part of schedule list
GetScheduleListResponse
(NextStartReference =: *NextStartReference*,
Schedule list =: *scheduleListPart*)
Send part of requested schedule list
Receive response


ONVIF Client
DUT
…
GetScheduleList
(Limit := *cap*.MaxLimit, StartReference :=
nextStartReference)
Retrieve the last part of schedule list
GetScheduleListResponse
(NextStartReference =: *NextStartReference*,
Schedule list =: *scheduleListPart*)
Send part of requested schedule list
Receive response
GetScheduleList
(Limit := 1, StartReference skipped)
Retrieve the first part of schedule list
GetScheduleListResponse
(NextStartReference =: *NextStartReference*,
Schedule list =: *scheduleCompleteList2*)
Send part of requested schedule list
Receive response
GetScheduleInfoList
(Limit := *1*, StartReference :=
nextStartReference)
Retrieve the second part of schedule list
GetScheduleListResponse
(NextStartReference =: *NextStartReference*,
Schedule list =: *scheduleListPart*)
Send part of requested schedule list
Receive response
…
GetScheduleList
(Limit := 1, StartReference :=
nextStartReference)
Retrieve the last part of schedule list
GetScheduleListResponse
(NextStartReference =: *NextStartReference*,
Schedule list =: *scheduleListPart*)
Send part of requested schedule list
Receive response


ONVIF Client
DUT
GetScheduleList
(Limit := *limit*, StartReference skipped)
Retrieve the first part of schedule list
GetScheduleListResponse
(NextStartReference =: *NextStartReference*, Schedule list =: *scheduleCompleteList3*)
Send part of requested schedule list
Receive response
GetScheduleList
(Limit := *limit*, StartReference := *nextStartReference*)
Retrieve the second part of schedule list
GetScheduleListResponse
(NextStartReference =: *NextStartReference*, Schedule list =: *scheduleListPart*)
Send part of requested schedule list
Receive response
…
GetScheduleList
(Limit := *limit*, StartReference := *nextStartReference*)
Retrieve the last part of schedule list
GetScheduleListResponse
(NextStartReference =: *NextStartReference*, Schedule list =: *scheduleListPart*)
Send part of requested schedule list
Receive response
HelperGetScheduleInfoList
(out *scheduleInfoCompleteList*)
Annex A.1
Retrieve a complete list of schedules
Return a complete list of schedules

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

4. ONVIF client invokes **GetScheduleList** with parameters

- Limit := *cap*.MaxLimit
- StartReference skipped

5. The DUT responds with **GetScheduleListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - Schedule list =: *scheduleCompleteList1*
6. If *scheduleCompleteList1* contains more Schedule items than *cap*.MaxLimit, FAIL the test and skip other steps.
7. Until *nextStartReference* is not null, repeat the following steps:

   7.1. ONVIF client invokes **GetScheduleList** with parameters

   - Limit := *cap*.MaxLimit
   - StartReference := *nextStartReference*

   7.2. The DUT responds with **GetScheduleListResponse** message with parameters

   - NextStartReference =: *nextStartReference*
   - Schedule list =: *scheduleListPart*

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

   7.4. Set the following:

   - *scheduleCompleteList1* := *scheduleCompleteList1* + *scheduleListPart*
8. If *scheduleCompleteList1* contains at least two Schedule items with equal token, FAIL the test and skip other steps.
9. If *cap*.MaxLimit is equal to 1, do the following steps:

   9.1. ONVIF Client retrieves a complete list of ScheduleInfo (out *scheduleInfoCompleteList*) by following the procedure mentioned in Annex A.1.

   9.2. If *scheduleCompleteList1* does not contain all tokens from *scheduleInfoCompleteList*, FAIL the test and skip other steps.

   9.3. If *scheduleCompleteList1* contains tokens other than tokens from schedule*InfoCompleteList*, FAIL the test and skip other steps.

   9.4. For each ScheduleInfo.token *token* from *scheduleInfoCompleteList* repeat the following steps:

   9.4.1. If *scheduleCompleteList1*[token = *token*] item does not have equal field values to *scheduleInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.

   9.5. Skip other steps.
10. ONVIF client invokes **GetScheduleList** with parameters

- Limit := 1
- StartReference skipped

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

    - NextStartReference =: *nextStartReference*
    - Schedule list =: *scheduleCompleteList2*

12. If *scheduleCompleteList2* contains more Schedule items than 1, FAIL the test and skip other steps.

13. Until *nextStartReference* is not null, repeat the following steps:

    13.1. ONVIF client invokes **GetScheduleList** with parameters

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

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

    - NextStartReference =: *nextStartReference*
    - Schedule list =: *scheduleListPart*

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

    13.4. Set the following:

    - *scheduleCompleteList2* := *scheduleCompleteList2* + *scheduleListPart*

14. If *scheduleCompleteList2* contains at least two Schedule items with equal token, FAIL the test and skip other steps.

15. If *scheduleCompleteList2* does not contain all schedules from *scheduleCompleteList1*, FAIL the test and skip other steps.

16. If *scheduleCompleteList2* contains schedules other than schedules from *scheduleCompleteList1*, FAIL the test and skip other steps.

17. If *cap*.MaxLimit is equal to 2 do the following steps:

    17.1. ONVIF Client retrieves a complete list of ScheduleInfo (out *scheduleInfoCompleteList*) by following the procedure mentioned in Annex A.1.

    17.2. If *scheduleCompleteList2* does not contain all tokens from *scheduleInfoCompleteList*, FAIL the test and skip other steps.

    17.3. If *scheduleCompleteList2* contains tokens other than tokens from *scheduleInfoCompleteList*, FAIL the test and skip other steps.

    17.4. For each ScheduleInfo.token *token* from *scheduleInfoCompleteList* repeat the following steps:

    17.4.1. If *scheduleCompleteList2*[token = *token*] item does not have equal field values to *scheduleInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.

17.5. Skip other steps.

18. Set the following:
    - *limit* := [number between 1 and *cap*.MaxLimit]
19. ONVIF client invokes **GetScheduleList** with parameters
    - Limit := *limit*
    - StartReference skipped
20. The DUT responds with **GetScheduleListResponse** message with parameters
    - NextStartReference =: *nextStartReference*
    - Schedule list =: *scheduleCompleteList3*
21. If *scheduleCompleteList3* contains more Schedule items than *limit*, FAIL the test and skip other steps.
22. Until *nextStartReference* is not null, repeat the following steps:

    22.1. ONVIF client invokes **GetScheduleList** with parameters

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

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

    - NextStartReference =: *nextStartReference*
    - Schedule list =: *scheduleListPart*

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

    22.4. Set the following:

    - *scheduleCompleteList3* := *scheduleCompleteList3* + *scheduleListPart*
23. If *scheduleCompleteList3* contains at least two Schedule items with equal token, FAIL the test and skip other steps.
24. If *scheduleCompleteList3* does not contain all schedules from *scheduleCompleteList1*, FAIL the test and skip other steps.
25. If *scheduleCompleteList3* contains schedules other than schedules from *scheduleCompleteList1*, FAIL the test and skip other steps.
26. ONVIF Client retrieves a complete list of ScheduleInfo (out *scheduleInfoCompleteList*) by following the procedure mentioned in Annex A.1.
27. If *scheduleCompleteList3* does not contain all tokens from *scheduleInfoCompleteList*, FAIL the test and skip other steps.
28. If *scheduleCompleteList3* contains tokens other than tokens from *scheduleInfoCompleteList*,

FAIL the test and skip other steps.

29. For each ScheduleInfo.token *token* from *scheduleInfoCompleteList* repeat the following steps:

    29.1. If *scheduleCompleteList3*[token = *token*] item does not have equal field values to *scheduleInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

Note: The following fields are compared at steps 9.4.1, 17.4.1, 29.1:

- ScheduleInfo:
    - token
    - Name
    - Description

4.3.4 GET SCHEDULE LIST - NO LIMIT

Test Label: Get Schedule List Verification without Limit

Test Case ID: SCHEDULE-3-1-4

ONVIF Core Specification Coverage: ScheduleInfo (ONVIF Schedule Service Specification), Schedule (ONVIF Schedule Service Specification), GetScheduleList command (ONVIF Schedule Service Specification)

Command Under Test: GetScheduleList

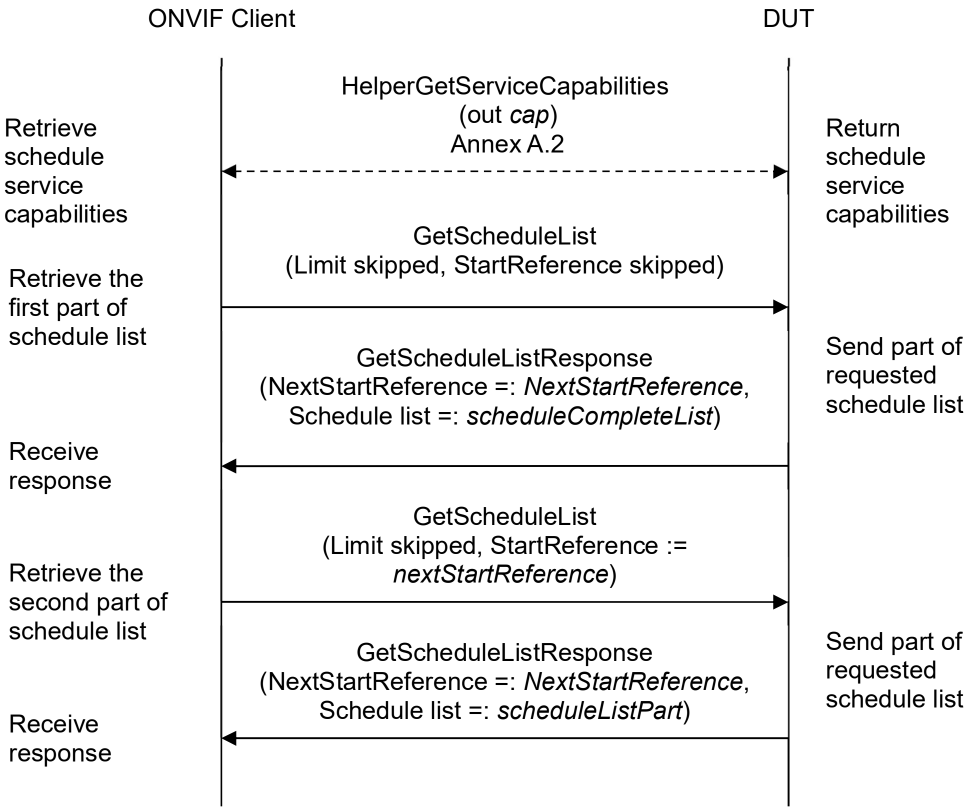
WSDL Reference: schedule.wsdl

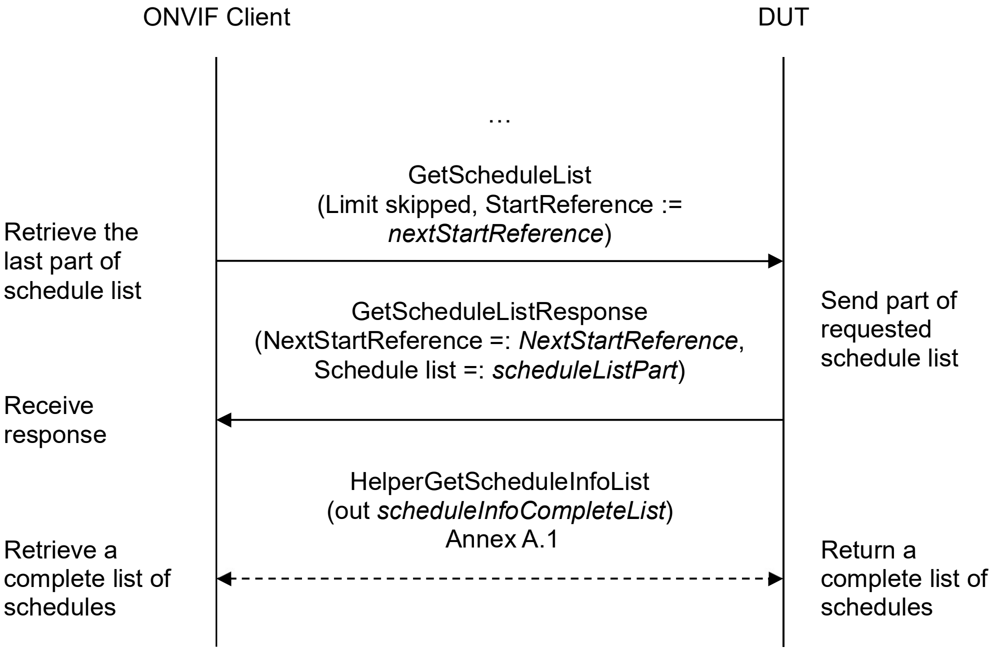
Test Purpose: To verify Get Schedule List without using Limit.

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

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieve schedule service capabilities
Return schedule service capabilities
GetScheduleList
(Limit skipped, StartReference skipped)
Retrieve the first part of schedule list
GetScheduleListResponse
(NextStartReference =: *NextStartReference*,
Schedule list =: *scheduleCompleteList*)
Send part of requested schedule list
Receive response
GetScheduleList
(Limit skipped, StartReference :=
nextStartReference)
Retrieve the second part of schedule list
GetScheduleListResponse
(NextStartReference =: *NextStartReference*,
Schedule list =: *scheduleListPart*)
Send part of requested schedule list
Receive response


ONVIF Client
DUT
…
GetScheduleList
(Limit skipped, StartReference := *nextStartReference*)
Retrieve the last part of schedule list
GetScheduleListResponse
(NextStartReference =: *NextStartReference*, Schedule list =: *scheduleListPart*)
Send part of requested schedule list
Receive response
HelperGetScheduleInfoList
(out *scheduleInfoCompleteList*)
Annex A.1
Retrieve a complete list of schedules
Return a complete list of schedules

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

4. ONVIF client invokes **GetScheduleList** with parameters

   - Limit skipped

   - StartReference skipped

5. The DUT responds with **GetScheduleListResponse** message with parameters

   - NextStartReference =: *nextStartReference*

   - Schedule list =: *scheduleCompleteList*

6. If *scheduleCompleteList* contains more Schedule items than *cap*.MaxLimit, FAIL the test and skip other steps.

7. Until *nextStartReference* is not null, repeat the following steps:

   7.1. ONVIF client invokes **GetScheduleList** with parameters

      - Limit skipped

      - StartReference := *nextStartReference*

7.2. The DUT responds with **GetScheduleListResponse** message with parameters

- NextStartReference =: *nextStartReference*
- Schedule list =: *scheduleListPart*

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

7.4. Set the following:

- *scheduleCompleteList* := *scheduleCompleteList* + *scheduleListPart*

8. If *scheduleCompleteList* contains at least two Schedule item with equal token, FAIL the test.

9. ONVIF Client retrieves a complete list of schedules (out *scheduleInfoCompleteList*) by following the procedure mentioned in Annex A.1.

10. If *scheduleCompleteList* does not contain all schedules from *scheduleInfoCompleteList*, FAIL the test and skip other steps.

11. If *scheduleCompleteList* contains schedules other than schedules from *scheduleInfoCompleteList*, FAIL the test and skip other steps.

12. For each ScheduleInfo.token *token* from *scheduleInfoCompleteList* repeat the following steps:

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

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

Note: The following fields are compared at step 12.1:

- ScheduleInfo:
  - token
  - Name
  - Description

4.3.5 CREATE SCHEDULE

Test Label: Create Schedule Verification

Test Case ID: SCHEDULE-3-1-5

ONVIF Core Specification Coverage: ScheduleInfo (ONVIF Schedule Service Specification), Schedule (ONVIF Schedule Service Specification), CreateSchedule command (ONVIF Schedule Service Specification)

Command Under Test: CreateSchedule

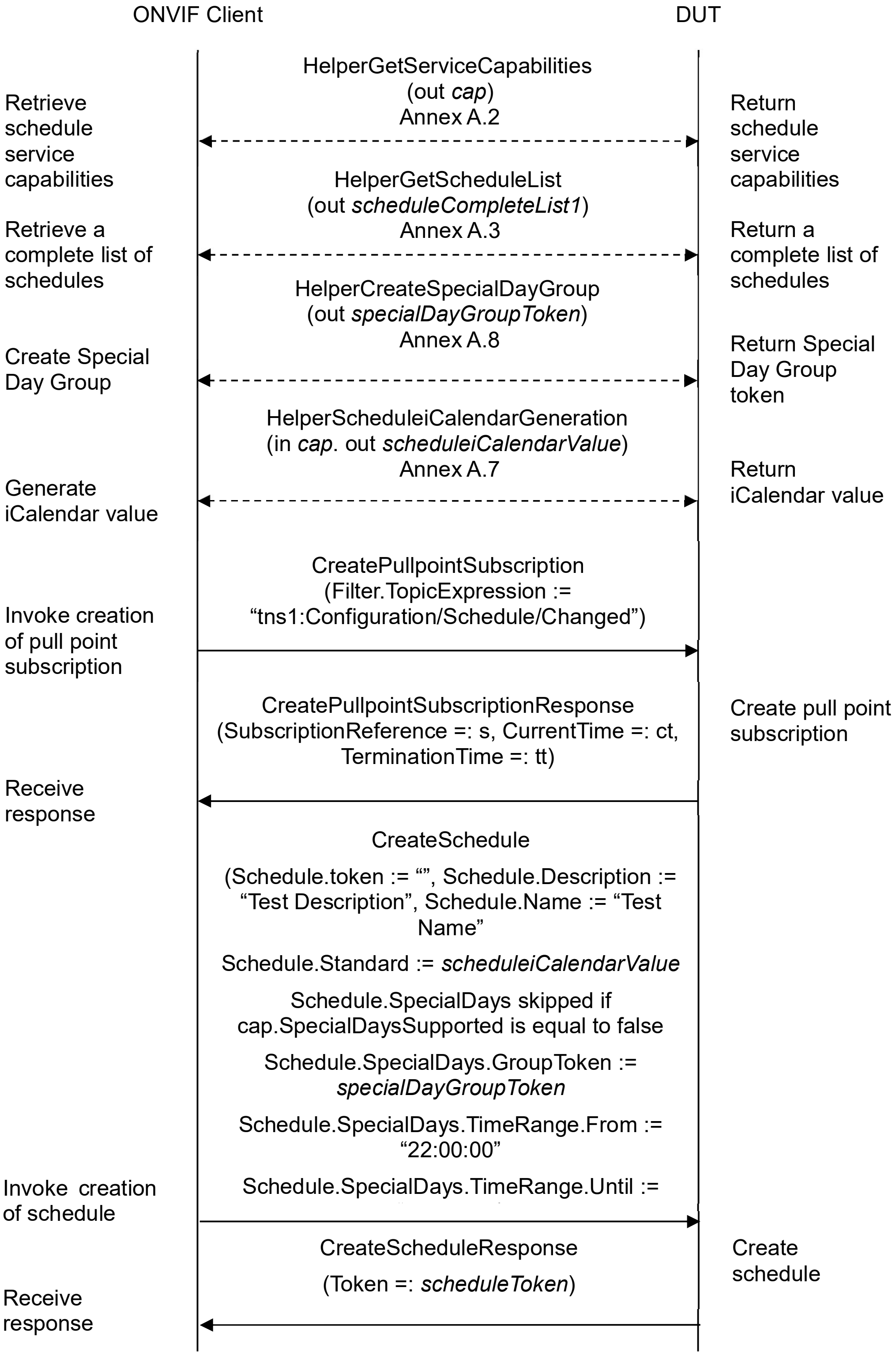
WSDL Reference: schedule.wsdl and event.wsdl

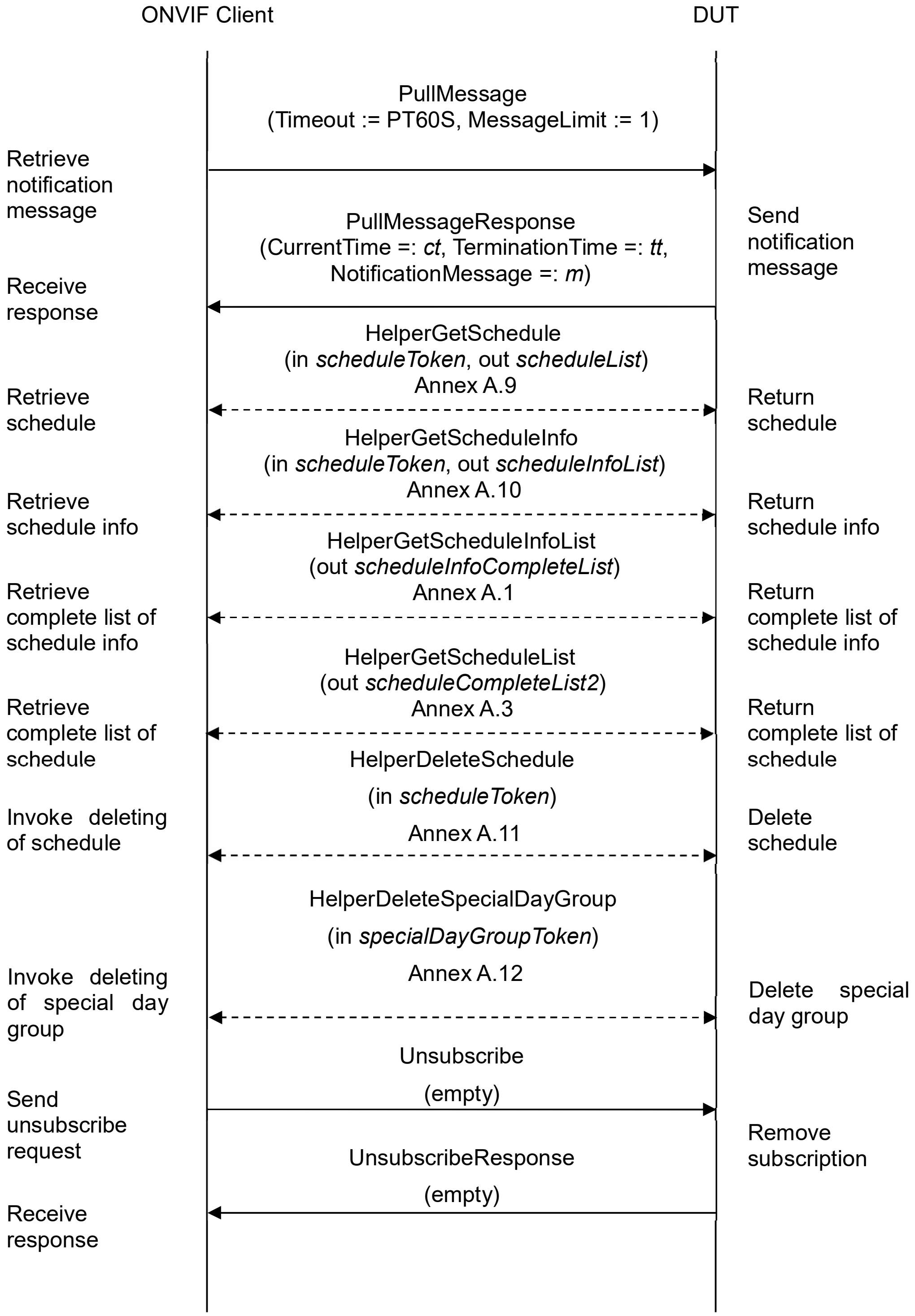
Test Purpose: To verify creation of schedule and generating of appropriate notifications.

Pre-requisite: Schedule Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional Schedule. The DUT shall have enough free storage capacity for one additional SpecialDayGroup if Special Days is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieve schedule service capabilities
Return schedule service capabilities
HelperGetScheduleList
(out *scheduleCompleteList1*)
Annex A.3
Retrieve a complete list of schedules
Return a complete list of schedules
HelperCreateSpecialDayGroup
(out *specialDayGroupToken*)
Annex A.8
Create Special Day Group
Return Special Day Group token
HelperScheduleiCalendarGeneration
(in *cap*. out *scheduleiCalendarValue*)
Annex A.7
Generate iCalendar value
Return iCalendar value
CreatePullpointSubscription
(Filter.TopicExpression :=
“tns1:Configuration/Schedule/Changed”)
Invoke creation of pull point subscription
CreatePullpointSubscriptionResponse
(SubscriptionReference =: s, CurrentTime =: ct,
TerminationTime =: tt)
Create pull point subscription
Receive response
CreateSchedule
(Schedule.token := “”, Schedule.Description :=
“Test Description”, Schedule.Name := “Test
Name”
Schedule.Standard := *scheduleiCalendarValue*
Schedule.SpecialDays skipped if
cap.SpecialDaysSupported is equal to false
Schedule.SpecialDays.GroupToken :=
specialDayGroupToken
Schedule.SpecialDays.TimeRange.From :=
“22:00:00”
Schedule.SpecialDays.TimeRange.Until :=
Invoke creation of schedule
CreateScheduleResponse
(Token =: *scheduleToken*)
Create schedule
Receive response


ONVIF Client
DUT
PullMessage
(Timeout := PT60S, MessageLimit := 1)
Retrieve notification message
PullMessageResponse
(CurrentTime =: *ct*, TerminationTime =: *tt*, NotificationMessage =: *m*)
Send notification message
Receive response
HelperGetSchedule
(in *scheduleToken*, out *scheduleList*)
Annex A.9
Retrieve schedule
Return schedule
HelperGetScheduleInfo
(in *scheduleToken*, out *scheduleInfoList*)
Annex A.10
Retrieve schedule info
Return schedule info
HelperGetScheduleInfoList
(out *scheduleInfoCompleteList*)
Annex A.1
Retrieve complete list of schedule info
Return complete list of schedule info
HelperGetScheduleList
(out *scheduleCompleteList2*)
Annex A.3
Retrieve complete list of schedule
Return complete list of schedule
HelperDeleteSchedule
(in *scheduleToken*)
Annex A.11
Invoke deleting of schedule
Delete schedule
HelperDeleteSpecialDayGroup
(in *specialDayGroupToken*)
Annex A.12
Invoke deleting of special day group
Delete special day group
Unsubscribe
(empty)
Send unsubscribe request
Remove subscription
UnsubscribeResponse
(empty)
Receive response

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.
4. ONVIF Client retrieves an initial complete list of schedules (out *scheduleCompleteList1*) by following the procedure mentioned in Annex A.3.
5. If *cap*.SpecialDaysSupported is equal to true, ONVIF Client creates SpecialDayGroup (out *specialDayGroupToken*) by following the procedure mentioned in Annex A.8.
6. ONVIF Client generates appropriate iCalendar value for the Schedule.Standard field (in *cap*, out *scheduleiCalendarValue*) by following the procedure mentioned in Annex A.7.
7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
    - Filter.TopicExpression := “tns1:Configuration/Schedule/Changed”
8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
    - SubscriptionReference =: *s*
    - CurrentTime =: *ct*
    - TerminationTime =: *tt*
9. ONVIF client invokes **CreateSchedule** with parameters
    - Schedule.token := “”
    - Schedule.Description := “Test Description”
    - Schedule.Name := “Test Name”
    - Schedule.Standard := *scheduleiCalendarValue*
    - Schedule.SpecialDays skipped if *cap*.SpecialDaysSupported is equal to false
    - Schedule.SpecialDays.GroupToken := *specialDayGroupToken*
    - Schedule.SpecialDays.TimeRange.From := “22:00:00”
    - Schedule.SpecialDays.TimeRange.Until := “23:00:00”
10. The DUT responds with **CreateScheduleResponse** message with parameters
    - Token =: *scheduleToken*
11. Until *timeout1* timeout expires, repeat the following steps:

    11.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.

    11.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters

- Timeout := PT60S
- MessageLimit := 1

11.3. The DUT responds with **PullMessagesResponse** message with parameters

- CurrentTime =: *ct*
- TerminationTime =: *tt*
- NotificationMessage =: *m*

11.4. If *m* is not null and the TopicExpression item in *m* is not equal to “tns1:Configuration/Schedule/Changed”, FAIL the test and go to the step 22.

11.5. If *m* is not null and does not contain Source.SimpleItem item with Name =: "ScheduleToken" and Value =: *scheduleToken*, FAIL the test and go to the step 22.

11.6. If *m* is not null and contains Source.SimpleItem item with Name =: "ScheduleToken" and Value =: *scheduleToken*, go to the step 13.

12. If *timeout1* timeout expires for step 11 without Notification with ScheduleToken source simple item equal to *scheduleToken*, FAIL the test and go to the step 22.

13. ONVIF Client retrieves a schedule (in *scheduleToken*, out *scheduleList*) by following the procedure mentioned in Annex A.9.

14. If *scheduleList*[0] item does not have equal field values to values from step 9, FAIL the test and go step 22.

15. ONVIF Client retrieves a schedule info (in *scheduleToken*, out *scheduleInfoList*) by following the procedure mentioned in Annex A.10.

16. If *scheduleInfoList*[0] item does not have equal field values to values from step 9, FAIL the test and go step 22.

17. ONVIF Client retrieves a complete schedule information list (out *scheduleInfoCompleteList*) by following the procedure mentioned in Annex A.1.

18. If *scheduleInfoCompleteList* does not have *scheduleInfo*[token = *scheduleToken*] item with equal field values to values from step 9, FAIL the test and go step 22.

19. ONVIF Client retrieves a complete list of schedules (out *scheduleCompleteList2*) by following the procedure mentioned in Annex A.3.

20. If *scheduleCompleteList2* does not have *schedule*[token = *scheduleToken*] item with equal field values to values from step 9, FAIL the test and go step 22.

21. For each Schedule.token (*token*) from *scheduleCompleteList1* do the following:

21.1. If *scheduleCompleteList2* does not have *schedule*[token = *token*] item, FAIL the test and go step 22.

22. ONVIF Client deletes the Schedule (in *scheduleToken*) by following the procedure mentioned in Annex A.11 to restore DUT configuration.

23. If SpecialDayGroup was created at step 4, ONVIF Client deletes the SpecialDayGroup (in *specialDayGroupToken*) by following the procedure mentioned in Annex A.12 to restore DUT

configuration.

24. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

25. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

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

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

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

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

Note: The following fields are compared at steps 14, 20:

- Schedule:
  - token
  - Name
  - Description
  - Standard
    - GroupToken
    - TimeRange
      - From
      - Until

Note: The following fields are compared at step 16, 18:

- ScheduleInfo:
  - token
  - Name
  - Description

Note: For comparison Schedule.Standard field values the ONVIF Client compares all iCalendar fields each with other excluding UID field.

4.3.6 MODIFY SCHEDULE

Test Label: Modify Schedule Verification

Test Case ID: SCHEDULE-3-1-6

ONVIF Core Specification Coverage: ScheduleInfo (ONVIF Schedule Service Specification), Schedule (ONVIF Schedule Service Specification), ModifySchedule command (ONVIF Schedule Service Specification)

Command Under Test: ModifySchedule

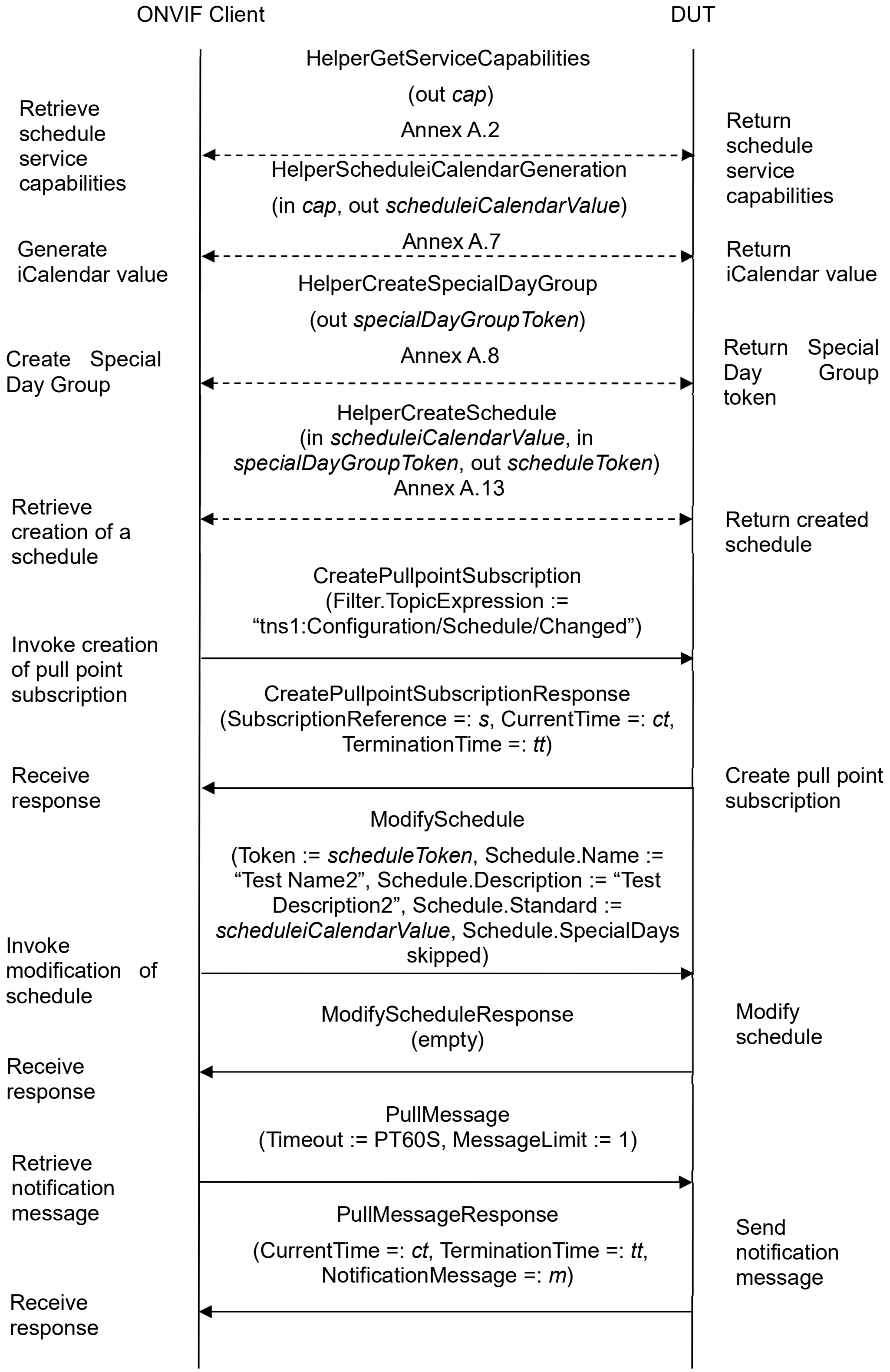
WSDL Reference: schedule.wsdl and event.wsdl

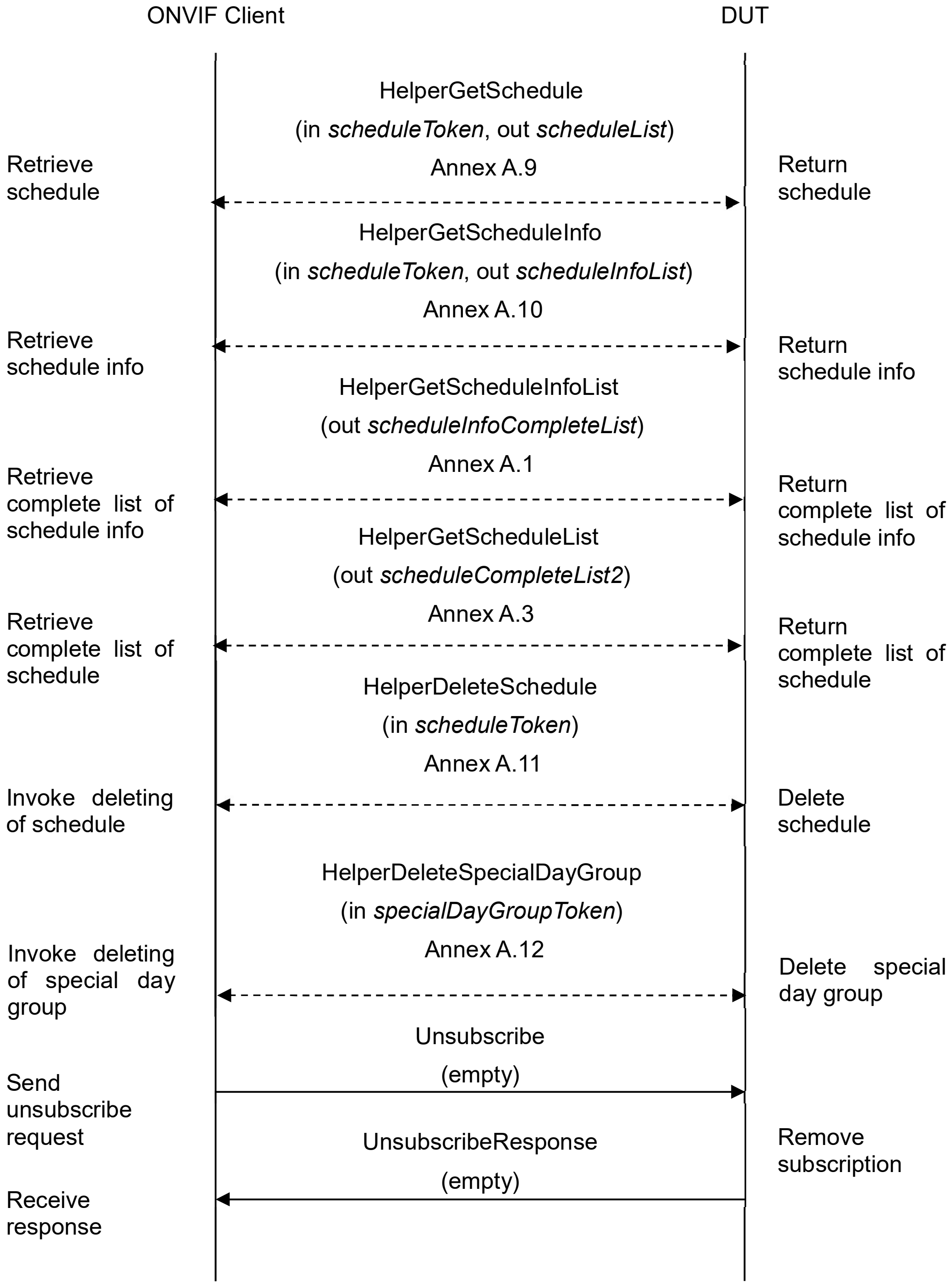
Test Purpose: To verify modifiing of schedule and generating of apropriate notifications.

Pre-requisite: Schedule Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional Schedule. The DUT shall have enough free storage capacity for one additional SpecialDayGroup if Special Days is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieve schedule service capabilities
Return schedule service capabilities
HelperScheduleiCalendarGeneration
(in *cap*, out *scheduleiCalendarValue*)
Annex A.7
Generate iCalendar value
Return iCalendar value
HelperCreateSpecialDayGroup
(out *specialDayGroupToken*)
Annex A.8
Create Special Day Group
Return Special Day Group token
HelperCreateSchedule
(in *scheduleiCalendarValue*, in *specialDayGroupToken*, out *scheduleToken*)
Annex A.13
Retrieve creation of a schedule
Return created schedule
CreatePullpointSubscription
(Filter.TopicExpression := “tns1:Configuration/Schedule/Changed”)
Invoke creation of pull point subscription
CreatePullpointSubscriptionResponse
(SubscriptionReference =: *s*, CurrentTime =: *ct*, TerminationTime =: *tt*)
Receive response
Create pull point subscription
ModifySchedule
(Token := *scheduleToken*, Schedule.Name := “Test Name2”, Schedule.Description := “Test Description2”, Schedule.Standard := *scheduleiCalendarValue*, Schedule.SpecialDays skipped)
Invoke modification of schedule
ModifyScheduleResponse
(empty)
Modify schedule
Receive response
PullMessage
(Timeout := PT60S, MessageLimit := 1)
Retrieve notification message
PullMessageResponse
(CurrentTime =: *ct*, TerminationTime =: *tt*, NotificationMessage =: *m*)
Send notification message
Receive response


ONVIF Client
DUT
HelperGetSchedule
(in *scheduleToken*, out *scheduleList*)
Annex A.9
Retrieve schedule
Return schedule
HelperGetScheduleInfo
(in *scheduleToken*, out *scheduleInfoList*)
Annex A.10
Retrieve schedule info
Return schedule info
HelperGetScheduleInfoList
(out *scheduleInfoCompleteList*)
Annex A.1
Retrieve complete list of schedule info
Return complete list of schedule info
HelperGetScheduleList
(out *scheduleCompleteList2*)
Annex A.3
Retrieve complete list of schedule
Return complete list of schedule
HelperDeleteSchedule
(in *scheduleToken*)
Annex A.11
Invoke deleting of schedule
Delete schedule
HelperDeleteSpecialDayGroup
(in *specialDayGroupToken*)
Annex A.12
Invoke deleting of special day group
Delete special day group
Unsubscribe
(empty)
Send unsubscribe request
Remove subscription
UnsubscribeResponse
(empty)
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

4. ONVIF Client generates appropriate iCalendar value for the Schedule.Standard field (in *cap*, out *scheduleiCalendarValue*) by following the procedure mentioned in Annex A.7.

5. If *cap*.SpecialDaysSupported is equal to true, ONVIF Client creates SpecialDayGroup (out *specialDayGroupToken*) by following the procedure mentioned in Annex A.8.

6. ONVIF Client creates Schedule with Schedule token (out *scheduleToken*), with iCalendar value of the Schedule.Standard field (in *scheduleiCalendarValue*) and with SpecialDayGroupToken (in *specialDayGroupToken*) if Special Days is supported by the DUT by following the procedure mentioned in Annex A.13.

7. Set the following:

    - *scheduleiCalendarValue* := *scheduleiCalendarValue* with changed <hour> in DTSTART and DTEND (increase hour values in 1)

8. ONVIF Client invokes **CreatePullPointSubscription** with parameters

    - Filter.TopicExpression := “tns1:Configuration/Schedule/Changed”

9. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters

    - SubscriptionReference =: *s*
    - CurrentTime =: *ct*
    - TerminationTime =: *tt*

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

    - Schedule.token := “*scheduleToken*”
    - Schedule.Name := “Test Name2”
    - Schedule.Description := “Test Description2”
    - Schedule.Standard := *scheduleiCalendarValue*
    - Schedule.SpecialDays skipped

11. The DUT responds with empty **ModifyScheduleResponse** message.

12. Until *timeout1* timeout expires, repeat the following steps:

    12.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.

    12.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters

    - Timeout := PT60S
    - MessageLimit := 1

    12.3. The DUT responds with **PullMessagesResponse** message with parameters

- CurrentTime =: *ct*
- TerminationTime =: *tt*
- NotificationMessage =: *m*

12.4. If *m* is not null and the TopicExpression item in *m* is not equal to “tns1:Configuration/Schedule/Changed”, FAIL the test and go to the step 23.

12.5. If *m* is not null and does not contain Source.SimpleItem item with Name =: "ScheduleToken" and Value =: *scheduleToken*, FAIL the test and go to the step 23.

12.6. If *m* is not null and contains Source.SimpleItem item with Name =: "ScheduleToken" and Value =: *scheduleToken*, go to the step 14.

13. If *timeout1* timeout expires for step 12 without Notification with ScheduleToken source simple item equal to *scheduleToken*, FAIL the test and go to the step 23.
14. ONVIF Client retrieves a schedule (in *scheduleToken*, out *scheduleList*) by following the procedure mentioned in Annex A.9.
15. If *scheduleList*[0] item does not have equal field values to values from step 10, FAIL the test and go step 23.
16. ONVIF Client retrieves a schedule info (in *scheduleToken*, out *scheduleInfoList*) by following the procedure mentioned in Annex A.10.
17. If *scheduleInfoList*[0] item does not have equal field values to values from step 10, FAIL the test and go step 23.
18. ONVIF Client retrieves a complete schedule information list (out *scheduleInfoCompleteList*) by following the procedure mentioned in Annex A.1.
19. If *scheduleInfoCompleteList* does not have *scheduleInfo.*[token = *scheduleToken*] item with equal field values to values from step 10, FAIL the test and go step 23.
20. ONVIF Client retrieves a complete list of schedules (out *scheduleCompleteList*) by following the procedure mentioned in Annex A.3.
21. If *scheduleCompleteList* does not have *schedule.*[token = *scheduleToken*] item with equal field values to values from step 10, FAIL the test and go step 23.
22. ONVIF Client deletes the Schedule (in *scheduleToken*) by following the procedure mentioned in Annex A.11 to restore DUT configuration.
23. If SpecialDayGroup was created at step 4, ONVIF Client deletes the SpecialDayGroup (in *specialDayGroupToken*) by following the procedure mentioned in Annex A.12 to restore DUT configuration.
24. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.
25. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

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

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

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

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

Note: The following fields are compared at steps 12, 18:

- Schedule:
  - token
  - Name
  - Description
  - Standard
    - GroupToken
    - TimeRange
      - From
      - Until

Note: The following fields are compared at step 14, 16:

- ScheduleInfo:
  - token
  - Name
  - Description

Note: For comparison Schedule.Standard field values the ONVIF Client compares all iCalendar fields each with other excluding UID field.

4.3.7 DELETE SCHEDULE

Test Label: Delete Schedule Verification

Test Case ID: SCHEDULE-3-1-7

ONVIF Core Specification Coverage: ScheduleInfo (ONVIF Schedule Service Specification), Schedule (ONVIF Schedule Service Specification), DeleteSchedule command (ONVIF Schedule Service Specification)

Command Under Test: DeleteSchedule

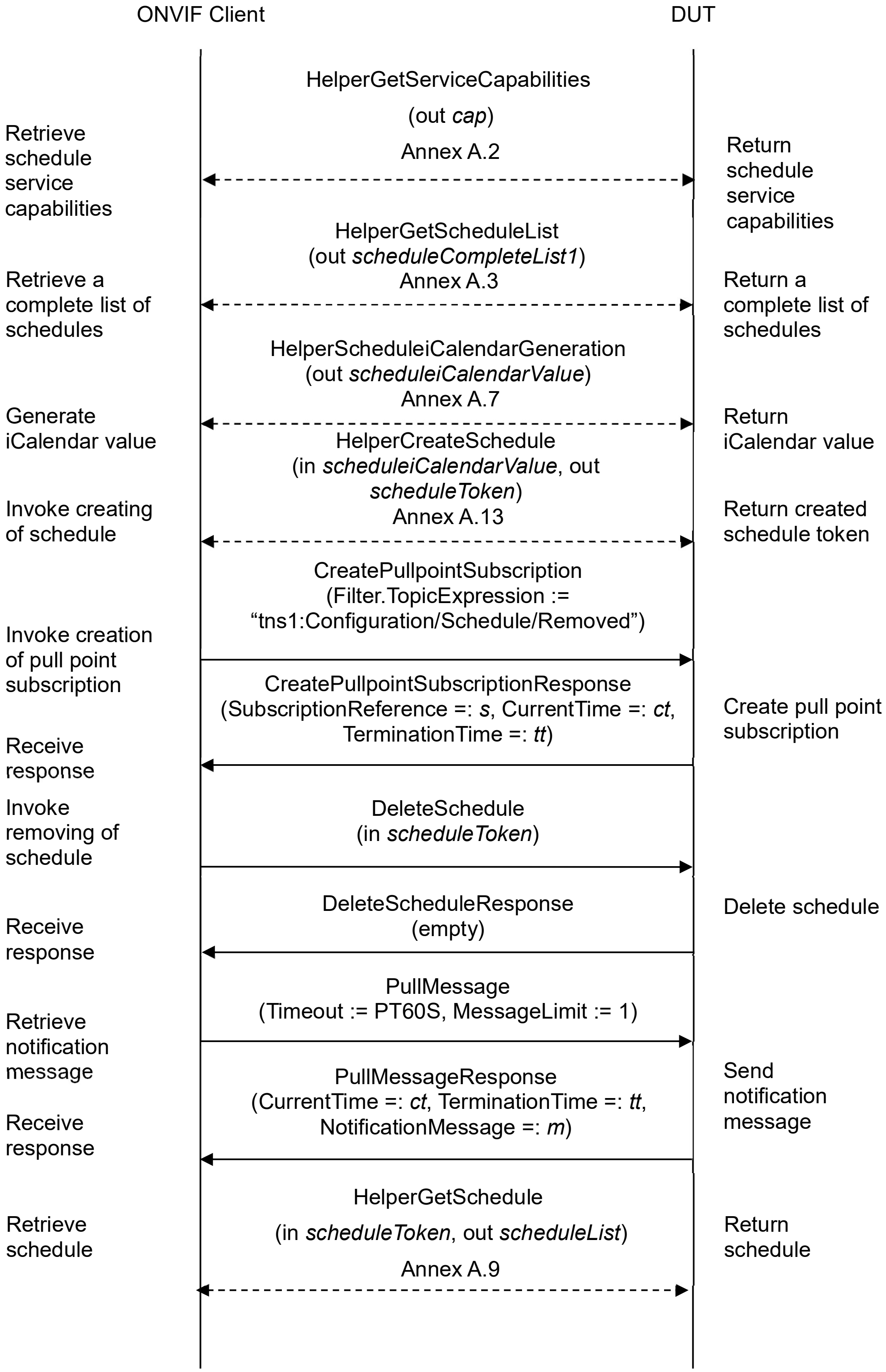
WSDL Reference: schedule.wsdl and event.wsdl

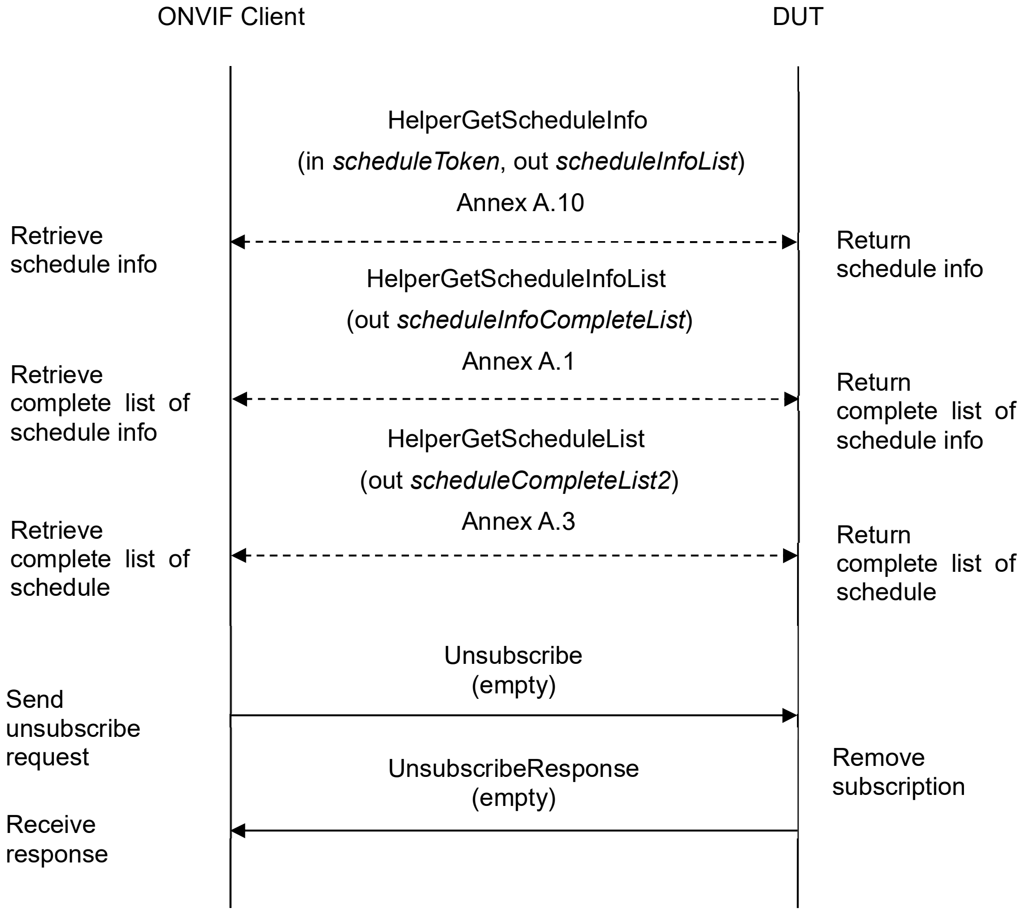
Test Purpose: To verify deleting of schedule and generating of apropriate notifications.

Pre-requisite: Schedule Service is received from the DUT. Event Service was received from the DUT. The DUT shall have enough free storage capacity for one additional Schedule.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieve schedule service capabilities
Return schedule service capabilities
HelperGetScheduleList
(out *scheduleCompleteList1*)
Annex A.3
Retrieve a complete list of schedules
Return a complete list of schedules
HelperScheduleiCalendarGeneration
(out *scheduleiCalendarValue*)
Annex A.7
Generate iCalendar value
Return iCalendar value
HelperCreateSchedule
(in *scheduleiCalendarValue*, out *scheduleToken*)
Annex A.13
Invoke creating of schedule
Return created schedule token
CreatePullpointSubscription
(Filter.TopicExpression :=
“tns1:Configuration/Schedule/Removed”)
Invoke creation of pull point subscription
CreatePullpointSubscriptionResponse
(SubscriptionReference =: *s*, CurrentTime =: *ct*,
TerminationTime =: *tt*)
Create pull point subscription
Receive response
DeleteSchedule
(in *scheduleToken*)
Invoke removing of schedule
DeleteScheduleResponse
(empty)
Delete schedule
Receive response
PullMessage
(Timeout := PT60S, MessageLimit := 1)
Retrieve notification message
PullMessageResponse
(CurrentTime =: *ct*, TerminationTime =: *tt*,
NotificationMessage =: *m*)
Send notification message
Receive response
HelperGetSchedule
(in *scheduleToken*, out *scheduleList*)
Annex A.9
Retrieve schedule
Return schedule


ONVIF Client
DUT
HelperGetScheduleInfo
(in *scheduleToken*, out *scheduleInfoList*)
Annex A.10
Retrieve schedule info
Return schedule info
HelperGetScheduleInfoList
(out *scheduleInfoCompleteList*)
Annex A.1
Retrieve complete list of schedule info
Return complete list of schedule info
HelperGetScheduleList
(out *scheduleCompleteList2*)
Annex A.3
Retrieve complete list of schedule
Return complete list of schedule
Unsubscribe
(empty)
Send unsubscribe request
Remove subscription
UnsubscribeResponse
(empty)
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

4. ONVIF Client retrieves a complete list of schedules (out schedule*CompleteList1*) by following the procedure mentioned in Annex A.3.

5. ONVIF Client generates appropriate iCalendar value for the Schedule.Standard field (in *cap*, out *scheduleiCalendarValue*) by following the procedure mentioned in Annex A.7.

6. ONVIF Client creates Schedule with Schedule token (out *scheduleToken*), with iCalendar value of the Schedule.Standard field (in *scheduleiCalendarValue*) and with skipped SpecialDayGroup by following the procedure mentioned in Annex A.13.

7. ONVIF Client invokes **CreatePullPointSubscription** with parameters
    - Filter.TopicExpression := “tns1:Configuration/Schedule/Removed”
8. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
    - SubscriptionReference =: *s*
    - CurrentTime =: *ct*
    - TerminationTime =: *tt*
9. ONVIF Client invokes **DeleteSchedule** with parameters
    - Token := *scheduleToken*
10. The DUT responds with empty **DeleteScheduleResponse** message.
11. Until *timeout1* timeout expires, repeat the following steps:

    11.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.

    11.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
    - Timeout := PT60S
    - MessageLimit := 1

    11.3. The DUT responds with **PullMessagesResponse** message with parameters
    - CurrentTime =: *ct*
    - TerminationTime =: *tt*
    - NotificationMessage =: *m*

    11.4. If *m* is not null and the TopicExpression item in *m* is not equal to “tns1:Configuration/Schedule/Removed”, FAIL the test and go to the step 21.

    11.5. If *m* is not null and does not contain Source.SimpleItem item with Name =: "ScheduleToken" and Value =: *scheduleToken*, FAIL the test and go to the step 21.

    11.6. If *m* is not null and contains Source.SimpleItem item with Name =: "ScheduleToken" and Value =: *scheduleToken*, go to the step 12.
12. If *timeout1* timeout expires for step 10 without Notification with ScheduleToken source simple item equal to *scheduleToken*, FAIL the test and go to the step 21.
13. ONVIF Client retrieves a schedule (in *scheduleToken*, out *scheduleList*) by following the procedure mentioned in Annex A.9.
14. If *scheduleList* is not empty, FAIL the test and go step 21.
15. ONVIF Client retrieves a schedule info (in *scheduleToken*, out *scheduleInfoList*) by following the procedure mentioned in Annex A.10.
16. If *scheduleInfoList* is not empty, FAIL the test and go step 21.

17. ONVIF Client retrieves a complete schedule information list (out *scheduleInfoCompleteList*) by following the procedure mentioned in Annex A.1.

18. If *scheduleInfoCompleteList* contains *scheduleInfo.*[token = *scheduleToken*] item, FAIL the test and go step 21.

19. ONVIF Client retrieves a complete list of schedules (out *scheduleCompleteList2*) by following the procedure mentioned in Annex A.3.

20. If *scheduleCompleteList2* contains *schedule.*[token = *scheduleToken*] item, FAIL the test and go step 20.

21. For each Schedule.token (*token*) from *scheduleCompleteList1* do the following:

    21.1. If *scheduleCompleteList2* does not have *schedule*[token = *token*] item, FAIL the test and go step 21.

22. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

23. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

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

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

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

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

4.3.8 GET SCHEDULES WITH INVALID TOKEN

Test Label: Get Schedules with Invalid Token Verification

Test Case ID: SCHEDULE-3-1-8

ONVIF Core Specification Coverage: Schedule (ONVIF Schedule Service Specification), GetSchedules command (ONVIF Schedule Service Specification)

Command Under Test: GetSchedules

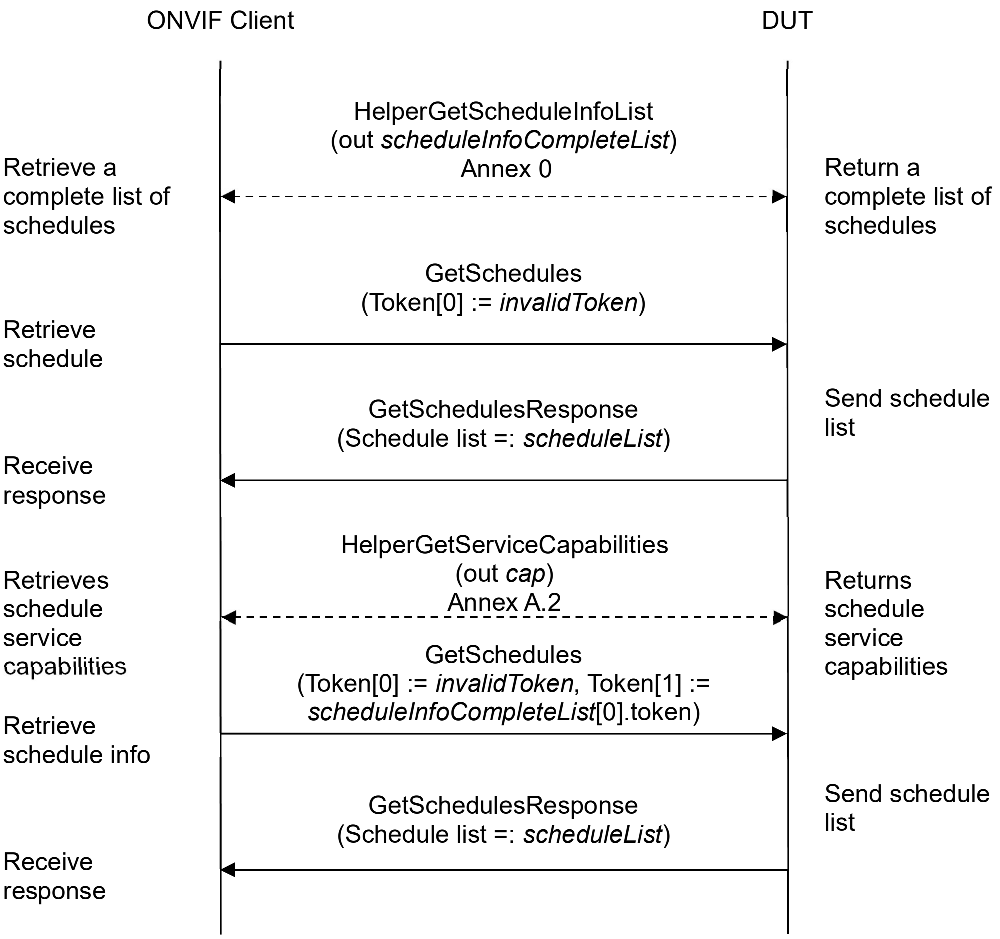
WSDL Reference: schedule.wsdl

Test Purpose: To verify Get Schedule with invalid token.

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

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetScheduleInfoList
(out *scheduleInfoCompleteList*)
Annex 0
Retrieve a complete list of schedules
Return a complete list of schedules
GetSchedules
(Token[0] := *invalidToken*)
Retrieve schedule
GetSchedulesResponse
(Schedule list =: *scheduleList*)
Send schedule list
Receive response
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieves schedule service capabilities
Returns schedule service capabilities
GetSchedules
(Token[0] := *invalidToken*, Token[1] :=
scheduleInfoCompleteList[0].token)
Retrieve schedule info
GetSchedulesResponse
(Schedule list =: *scheduleList*)
Send schedule list
Receive response

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves a complete list of schedule info (out *scheduleInfoCompleteList*) by following the procedure mentioned in Annex 0.
4. Set the following:
    - *invalidToken* := value not equal to any *scheduleInfoCompleteList*.token
5. ONVIF client invokes **GetSchedules** with parameters
    - Token list := *invalidToken*
6. The DUT responds with **GetSchedulesResponse** message with parameters
    - Schedule list =: *schedulesList*
7. If *schedulesList* is not empty, FAIL the test.
8. If *scheduleInfoCompleteList* is empty, skip other steps.
9. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.
10. If *cap*.MaxLimit is less than 2, skip other steps.
11. ONVIF client invokes **GetSchedules** with parameters
    - Token[0]:= *invalidToken*
    - Token[1]:= *scheduleInfoCompleteList*[0].token
12. The DUT responds with **GetSchedulesResponse** message with parameters
    - ScheduleInfo list =: *schedulesList*
13. If *schedulesList* is empty, FAIL the test.
14. If *schedulesList* contains more than one item, FAIL the test.
15. If *schedulesList*[0].token does not equal to *scheduleInfoCompleteList*[0].token, FAIL the test.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

4.3.9 GET SCHEDULE - TOO MANY ITEMS

Test Label: Get Schedules - number of requested items is greater than MaxLimit

Test Case ID: SCHEDULE-3-1-9

ONVIF Core Specification Coverage: Schedule (ONVIF Schedule Service Specification), GetSchedules command (ONVIF Schedule Service Specification)

Command Under Test: GetSchedules

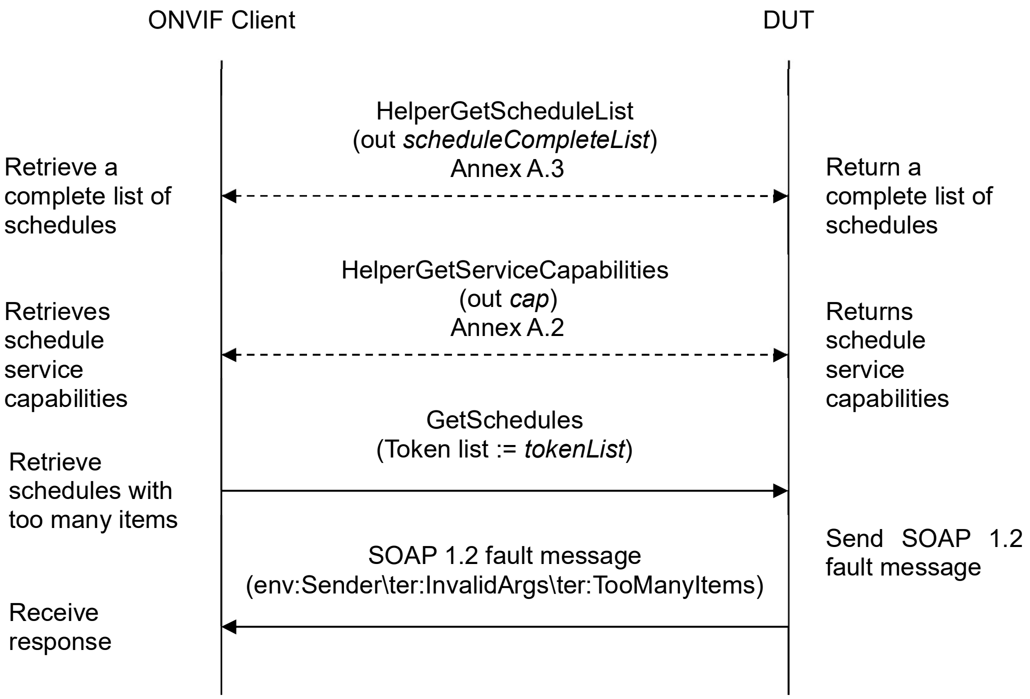
WSDL Reference: schedule.wsdl

Test Purpose: To verify Get Schedule in case there are more items than MaxLimit in request.

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

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetScheduleList
(out *scheduleCompleteList*)
Annex A.3
Retrieve a complete list of schedules
Return a complete list of schedules
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieves schedule service capabilities
Returns schedule service capabilities
GetSchedules
(Token list := *tokenList*)
Retrieve schedules with too many items
SOAP 1.2 fault message
(env:Sender\ter:InvalidArgs\ter:TooManyItems)
Send SOAP 1.2 fault message
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of schedules (out *scheduleCompleteList*) by following the procedure mentioned in Annex A.3.

4. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

5. If *scheduleCompleteList*.token items number is less than *cap*.MaxLimit or equal to *cap*.MaxLimit, skip other steps.

6. Set the following:

    - *tokenList* := [subset of *scheduleCompleteList*.token values with items number equal to *cap*.MaxLimit + 1]

7. ONVIF client invokes **GetSchedules** with parameters

    - Token list := *tokenList*

8. The DUT returns **env:Sender\ter:InvalidArgs\ter:TooManyItems** SOAP 1.2 fault.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send env:Sender\ter:InvalidArgs\ter:TooManyItems SOAP 1.2 fault

4.3.10 CREATE SCHEDULE - NOT EMPTY SCHEDULE TOKEN

Test Label: Create Schedule with not Empty Token Verification

Test Case ID: SCHEDULE-3-1-10

ONVIF Core Specification Coverage: CreateSchedule command (ONVIF Schedule Service Specification)

Command Under Test: CreateSchedule

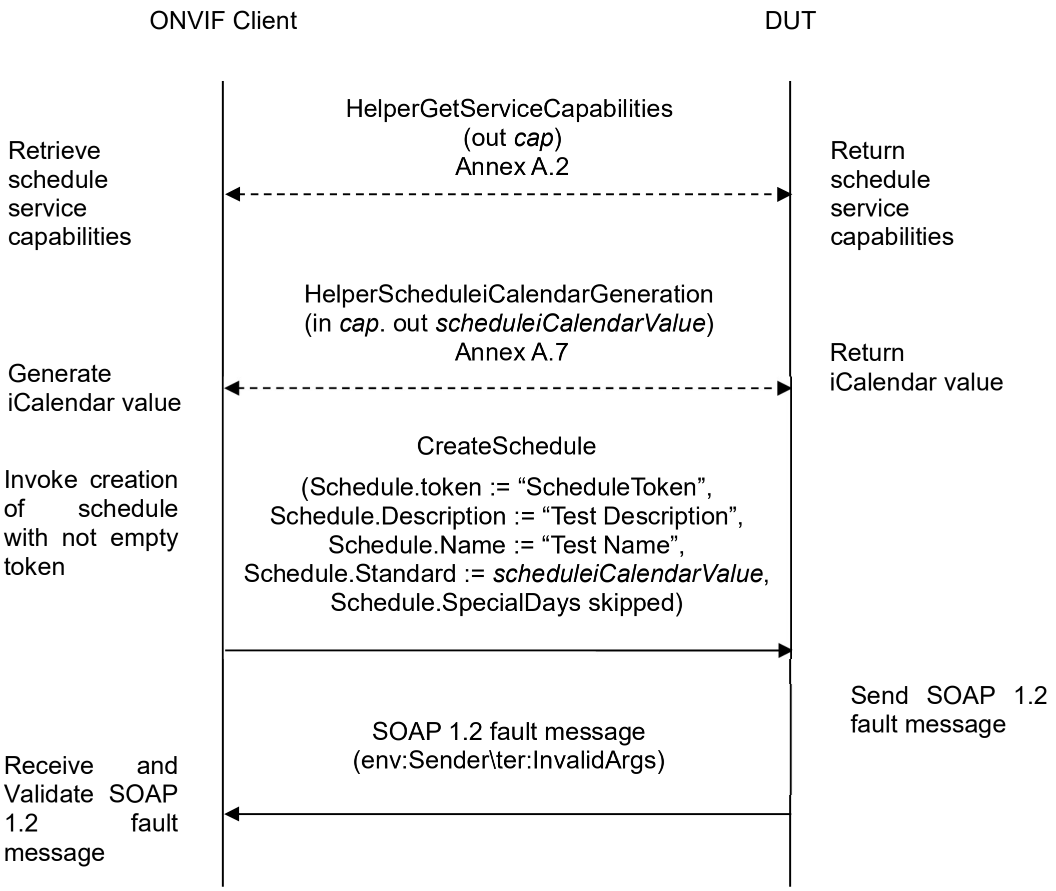
WSDL Reference: schedule.wsdl

Test Purpose: To verify Create Schedule with not empty token.

Pre-requisite: Schedule Service is received from the DUT. The DUT shall have enough free storage capacity for one additional Schedule.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieve schedule service capabilities
Return schedule service capabilities
HelperScheduleiCalendarGeneration
(in *cap*. out *scheduleiCalendarValue*)
Annex A.7
Generate iCalendar value
Return iCalendar value
CreateSchedule
(Schedule.token := “ScheduleToken”,
Schedule.Description := “Test Description”,
Schedule.Name := “Test Name”,
Schedule.Standard := *scheduleiCalendarValue*,
Schedule.SpecialDays skipped)
Invoke creation of schedule with not empty token
Send SOAP 1.2 fault message
SOAP 1.2 fault message
(env:Sender\ter:InvalidArgs)
Receive and Validate SOAP 1.2 fault message

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.
4. ONVIF Client generates appropriate iCalendar value for the Schedule.Standard field (in *cap*, out *scheduleiCalendarValue*) by following the procedure mentioned in Annex A.7.
5. ONVIF client invokes **CreateSchedule** with parameters
    - Schedule.token := "ScheduleToken"
    - Schedule.Description := "Test Description"
    - Schedule.Name := "Test Name"
    - Schedule.Standard := *scheduleiCalendarValue*
    - Schedule.SpecialDays skipped
6. The DUT returns **env:Sender\ter:InvalidArgs** SOAP 1.2 fault.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send env:Sender\ter:InvalidArgs SOAP 1.2 fault.

4.3.11 CREATE SCHEDULE - TOO MANY TIME PERIODS PER DAY

Test Label: Create Schedule with to many Time Periods per Day Verification

Test Case ID: SCHEDULE-3-1-11

ONVIF Core Specification Coverage: CreateSchedule command (ONVIF Schedule Service Specification)

Command Under Test: CreateSchedule

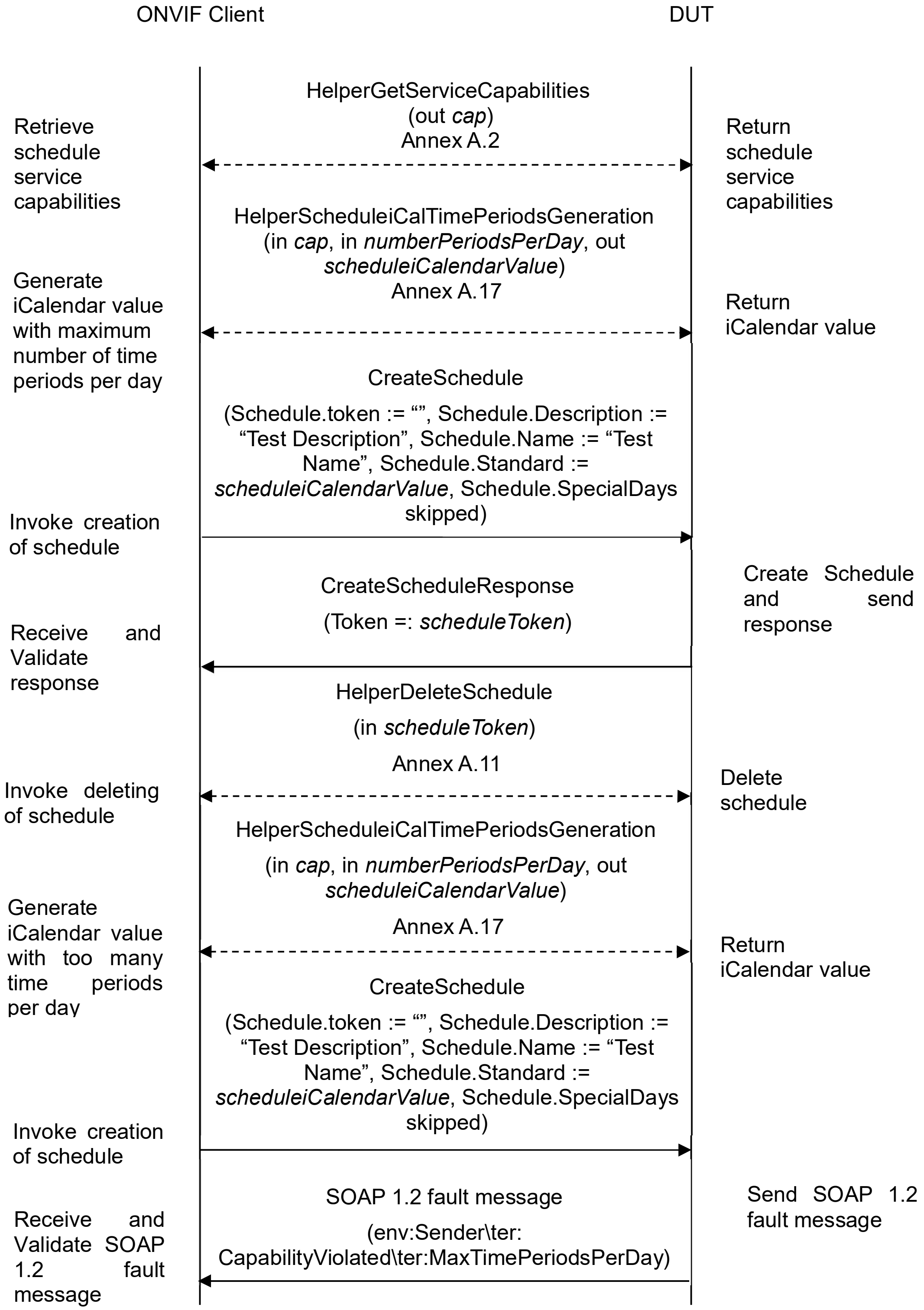
WSDL Reference: schedule.wsdl

Test Purpose: To verify Create Schedule with maximum number of time periods per day and with to many time periods per day.

Pre-requisite: Schedule Service is received from the DUT. The DUT shall have enough free storage capacity for one additional Schedule.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Retrieve schedule service capabilities
HelperGetServiceCapabilities (out *cap*) Annex A.2
Return schedule service capabilities
Generate iCalendar value with maximum number of time periods per day
HelperScheduleiCalTimePeriodsGeneration (in *cap*, in *numberPeriodsPerDay*, out *scheduleiCalendarValue*) Annex A.17
Return iCalendar value
Invoke creation of schedule
CreateSchedule
(Schedule.token := “”, Schedule.Description := “Test Description”, Schedule.Name := “Test Name”, Schedule.Standard := *scheduleiCalendarValue*, Schedule.SpecialDays skipped)
Create Schedule and send response
Receive and Validate response
CreateScheduleResponse
(Token =: *scheduleToken*)
Invoke deleting of schedule
HelperDeleteSchedule
(in *scheduleToken*)
Annex A.11
Delete schedule
Generate iCalendar value with too many time periods per day
HelperScheduleiCalTimePeriodsGeneration
(in *cap*, in *numberPeriodsPerDay*, out *scheduleiCalendarValue*)
Annex A.17
Return iCalendar value
Invoke creation of schedule
CreateSchedule
(Schedule.token := “”, Schedule.Description := “Test Description”, Schedule.Name := “Test Name”, Schedule.Standard := *scheduleiCalendarValue*, Schedule.SpecialDays skipped)
Send SOAP 1.2 fault message
Receive and Validate SOAP 1.2 fault message
SOAP 1.2 fault message
(env:Sender\ter: CapabilityViolated\ter:MaxTimePeriodsPerDay)

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

4. If MaxTimePeriodsPerDay is equal to one, go to step 10.

5. Set the following:

   - *numberPeriodsPerDay* := *cap*.MaxTimePeriodsPerDay

6. ONVIF Client generates appropriate iCalendar value for the Schedule.Standard field (in *cap,* out *scheduleiCalendarValue*) with maximum number of time periods per day (in *numberPeriodsPerDay*) by following the procedure mentioned in Annex A.17.

7. ONVIF client invokes **CreateSchedule** with parameters

   - Schedule.token := “”

   - Schedule.Description := “Test Description”

   - Schedule.Name := “Test Name”

   - Schedule.Standard := *scheduleiCalendarValue*

   - Schedule.SpecialDays skipped

8. The DUT responds with **CreateScheduleResponse** message with parameters

   - Token =: *scheduleToken*

9. ONVIF Client deletes the Schedule (in *scheduleToken*) by following the procedure mentioned in Annex A.11 to restore DUT configuration.

10. If *cap*.MaxTimePeriodsPerDay value is more than 720, skip other steps.

11. Set the following:

    - *numberPeriodsPerDay* := *cap*.MaxTimePeriodsPerDay + 1

12. ONVIF Client generates appropriate iCalendar value for the Schedule.Standard field (in *cap,* out *scheduleiCalendarValue*) with too many number of time periods per day (in *numberPeriodsPerDay*) by following the procedure mentioned in Annex A.17.

13. ONVIF client invokes **CreateSchedule** with parameters

    - Schedule.token := “”

    - Schedule.Description := “Test Description”

    - Schedule.Name := “Test Name”

    - Schedule.Standard := *scheduleiCalendarValue*

- Schedule.SpecialDays skipped

14. The DUT returns **env:Sender\ter: CapabilityViolated\ter:MaxTimePeriodsPerDay** SOAP 1.2 fault.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send **CreateScheduleResponse** to schedule creation with maximum number of time periods per day

The DUT did not send **env:Sender\ter: CapabilityViolated\ter:MaxTimePeriodsPerDay** SOAP 1.2 fault to schedule creation with too many time periods per day.

Note: If the DUT sends other SOAP 1.2 fault message than specified, log WARNING message, and PASS the test.

4.3.12 CREATE SCHEDULE - INVALID TIME RANGE INTERVAL

Test Label: Create Schedule with Invalid Time Range Interval Verification

Test Case ID: SCHEDULE-3-1-12

ONVIF Core Specification Coverage: TimePeriod (ONVIF Schedule Service Specification), CreateSchedule command (ONVIF Schedule Service Specification)

Command Under Test: CreateSchedule

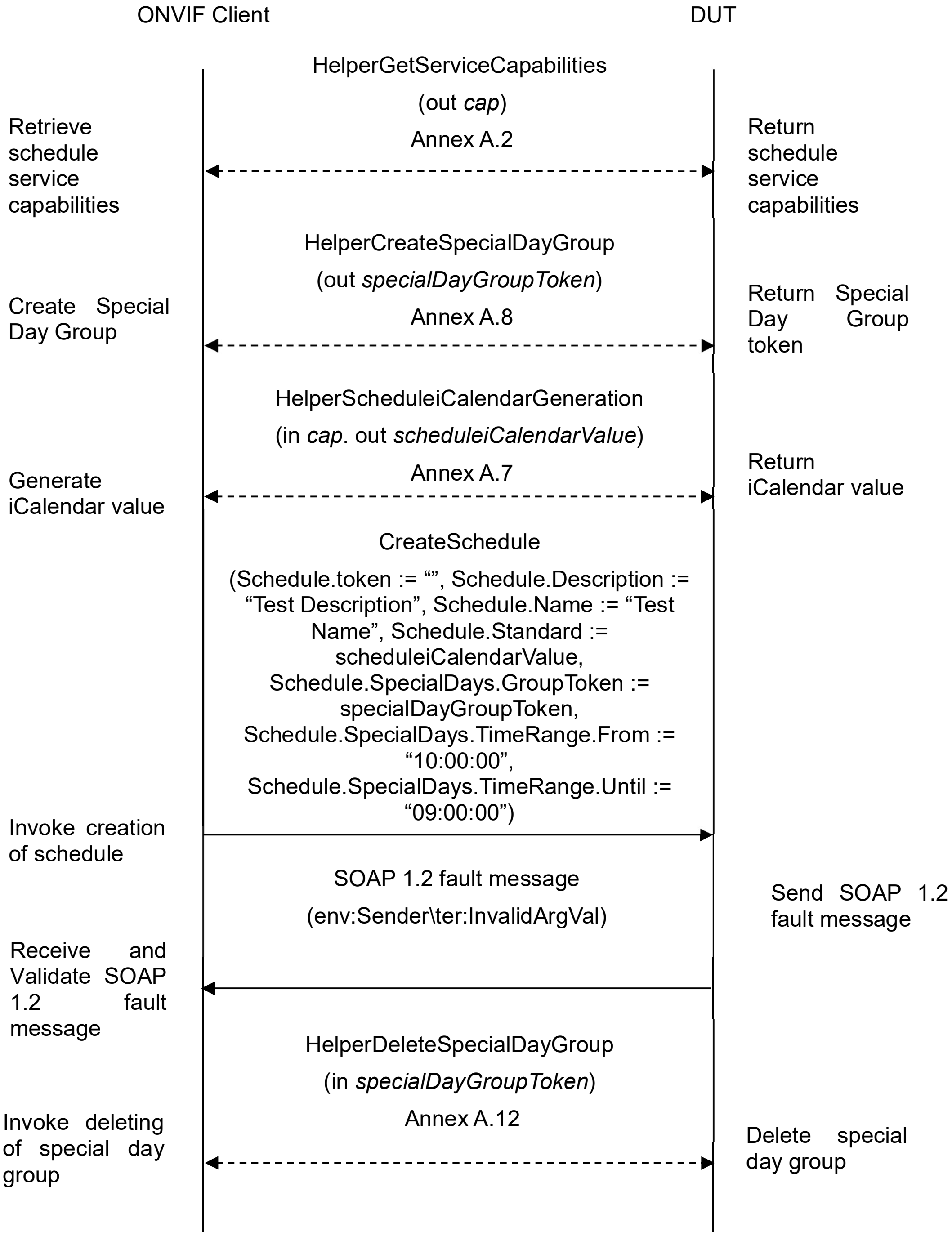
WSDL Reference: schedule.wsdl

Test Purpose: To verify Create Schedule with time range interval where the end time is less than the start time.

Pre-requisite: Schedule Service is received from the DUT. The DUT shall have enough free storage capacity for one additional Schedule. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported. The DUT shall have enough free storage capacity for one additional SpecialDayGroup.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieve schedule service capabilities
Return schedule service capabilities
HelperCreateSpecialDayGroup
(out *specialDayGroupToken*)
Annex A.8
Create Special Day Group
Return Special Day Group token
HelperScheduleiCalendarGeneration
(in *cap*. out *scheduleiCalendarValue*)
Annex A.7
Generate iCalendar value
Return iCalendar value
CreateSchedule
(Schedule.token := “”, Schedule.Description := “Test Description”, Schedule.Name := “Test Name”, Schedule.Standard := scheduleiCalendarValue, Schedule.SpecialDays.GroupToken := specialDayGroupToken, Schedule.SpecialDays.TimeRange.From := “10:00:00”, Schedule.SpecialDays.TimeRange.Until := “09:00:00”)
Invoke creation of schedule
SOAP 1.2 fault message
(env:Sender\ter:InvalidArgVal)
Send SOAP 1.2 fault message
Receive and Validate SOAP 1.2 fault message
HelperDeleteSpecialDayGroup
(in *specialDayGroupToken*)
Annex A.12
Invoke deleting of special day group
Delete special day group

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.
4. ONVIF Client creates SpecialDayGroup (out *specialDayGroupToken*) by following the procedure mentioned in Annex A.8.
5. ONVIF Client generates appropriate iCalendar value for the Schedule.Standard field (in *cap*, out *scheduleiCalendarValue*) by following the procedure mentioned in Annex A.7.
6. ONVIF client invokes **CreateSchedule** with parameters
    - Schedule.token := “”
    - Schedule.Description := “Test Description”
    - Schedule.Name := “Test Name”
    - Schedule.Standard := *scheduleiCalendarValue*
    - Schedule.SpecialDays.GroupToken := *specialDayGroupToken*
    - Schedule.SpecialDays.TimeRange.From := “10:00:00”
    - Schedule.SpecialDays.TimeRange.Until := “09:00:00”
7. The DUT returns **env:Sender\ter:InvalidArgVal** SOAP 1.2 fault.
8. ONVIF Client deletes the SpecialDayGroup (in *specialDayGroupToken*) by following the procedure mentioned in Annex A.12 to restore DUT configuration.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

Note: If the DUT sends other SOAP 1.2 fault message than specified, log WARNING message, and PASS the test.

4.3.13 MODIFY SCHEDULE WITH INVALID TOKEN

Test Label: Modify Schedule Verification

Test Case ID: SCHEDULE-3-1-13

ONVIF Core Specification Coverage: ModifySchedule command (ONVIF Schedule Service Specification)

Command Under Test: ModifySchedule

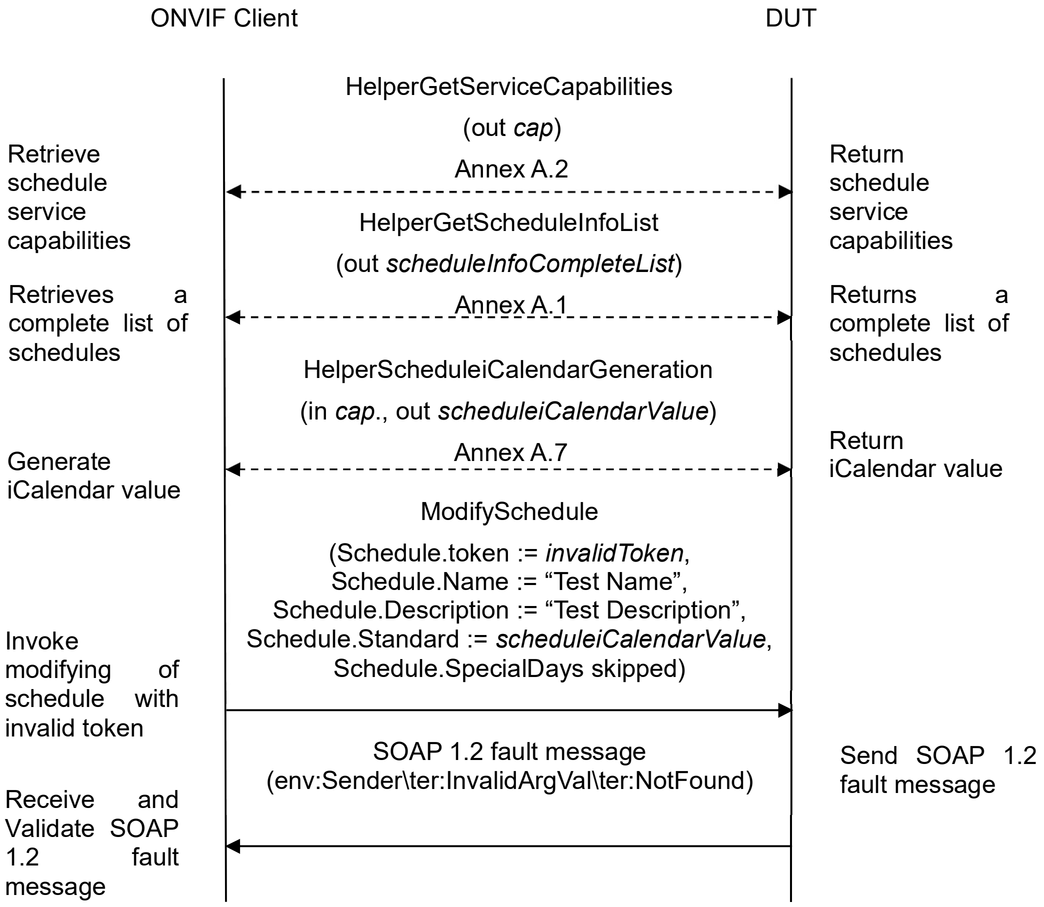
WSDL Reference: schedule.wsdl

Test Purpose: To verify modifiing of schedule with invalid token.

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

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out cap)
Annex A.2
Retrieve schedule service capabilities
Return schedule service capabilities
HelperGetScheduleInfoList
(out scheduleInfoCompleteList)
Annex A.1
Retrieves a complete list of schedules
Returns a complete list of schedules
HelperScheduleiCalendarGeneration
(in cap., out scheduleiCalendarValue)
Annex A.7
Generate iCalendar value
Return iCalendar value
ModifySchedule
(Schedule.token := invalidToken,
Schedule.Name := "Test Name",
Schedule.Description := "Test Description",
Schedule.Standard := scheduleiCalendarValue,
Schedule.SpecialDays skipped)
Invoke modifying of schedule with invalid token
SOAP 1.2 fault message
(env:Sender\ter:InvalidArgVal\ter:NotFound)
Send SOAP 1.2 fault message
Receive and Validate SOAP 1.2 fault message

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.
4. ONVIF Client retrieves a complete list of schedule info (out *scheduleInfoCompleteList*) by following the procedure mentioned in Annex 0.
5. ONVIF Client generates appropriate iCalendar value for the Schedule.Standard field (in *cap*, out *scheduleiCalendarValue*) by following the procedure mentioned in Annex A.7.
6. Set the following:
    - *invalidToken* := value not equal to any *scheduleInfoCompleteList*.token
7. ONVIF client invokes **ModifySchedule** with parameters
    - Schedule.token := *invalidToken*
    - Schedule.Name := “Test Name”
    - Schedule.Description := “Test Description”
    - Schedule.Standard := *scheduleiCalendarValu*e
    - Schedule.SpecialDays skipped
8. The DUT returns **env:Sender\ter:InvalidArgVal\ter:NotFound** SOAP 1.2 fault.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

Note: If the DUT sends other SOAP 1.2 fault message than specified, log WARNING message, and PASS the test.

4.3.14 MODIFY SCHEDULE - TOO MANY TIME PERIODS PER DAY

Test Label: Modify Schedule with to many Time Periods per Day Verification

Test Case ID: SCHEDULE-3-1-14

ONVIF Core Specification Coverage: ModifySchedule command (ONVIF Schedule Service Specification)

Command Under Test: ModifySchedule

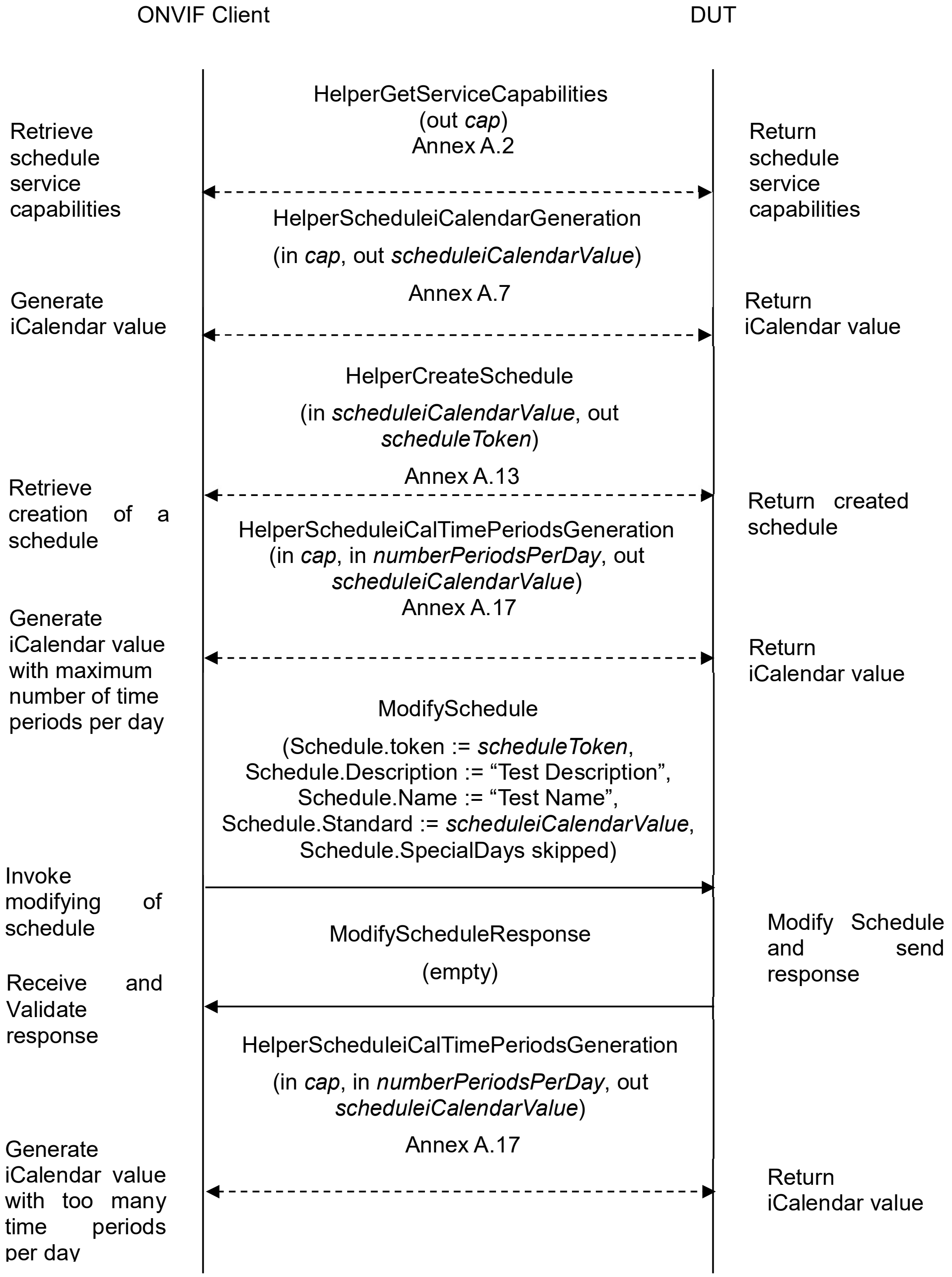
WSDL Reference: schedule.wsdl

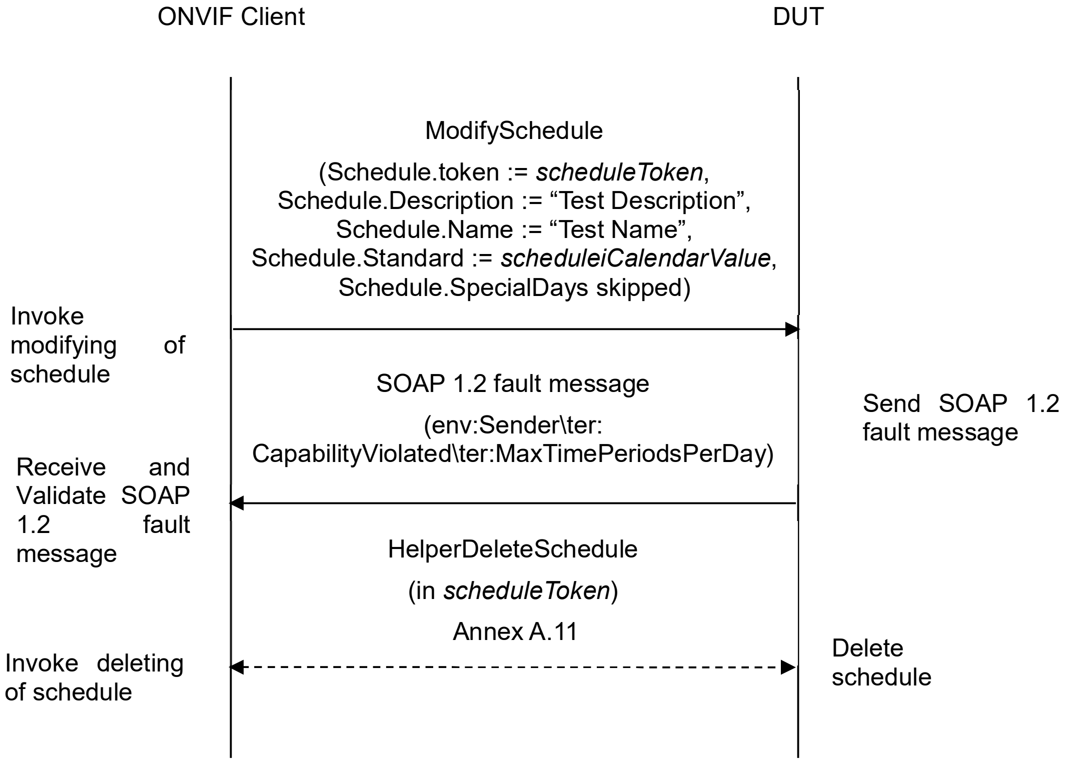
Test Purpose: To verify Modify Schedule with maximum number of time periods per day and with to many time periods per day.

Pre-requisite: Schedule Service is received from the DUT. The DUT shall have enough free storage capacity for one additional Schedule.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieve schedule service capabilities
Return schedule service capabilities
HelperScheduleiCalendarGeneration
(in *cap*, out *scheduleiCalendarValue*)
Annex A.7
Generate iCalendar value
Return iCalendar value
HelperCreateSchedule
(in *scheduleiCalendarValue*, out *scheduleToken*)
Annex A.13
Retrieve creation of a schedule
Return created schedule
HelperScheduleiCalTimePeriodsGeneration
(in *cap*, in *numberPeriodsPerDay*, out *scheduleiCalendarValue*)
Annex A.17
Generate iCalendar value with maximum number of time periods per day
Return iCalendar value
ModifySchedule
(Schedule.token := *scheduleToken*,
Schedule.Description := “Test Description”,
Schedule.Name := “Test Name”,
Schedule.Standard := *scheduleiCalendarValue*,
Schedule.SpecialDays skipped)
Invoke modifying of schedule
Modify Schedule and send response
ModifyScheduleResponse
(empty)
Receive and Validate response
HelperScheduleiCalTimePeriodsGeneration
(in *cap*, in *numberPeriodsPerDay*, out *scheduleiCalendarValue*)
Annex A.17
Generate iCalendar value with too many time periods per day
Return iCalendar value


ONVIF Client
DUT
ModifySchedule
(Schedule.token := *scheduleToken*, Schedule.Description := “Test Description”, Schedule.Name := “Test Name”, Schedule.Standard := *scheduleiCalendarValue*, Schedule.SpecialDays skipped)
Invoke modifying of schedule
SOAP 1.2 fault message
(env:Sender\ter: CapabilityViolated\ter:MaxTimePeriodsPerDay)
Send SOAP 1.2 fault message
Receive and Validate SOAP 1.2 fault message
HelperDeleteSchedule
(in *scheduleToken*)
Annex A.11
Invoke deleting of schedule
Delete schedule

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

4. ONVIF Client generates appropriate iCalendar value for the Schedule.Standard field (in *cap*, out *scheduleiCalendarValue*) by following the procedure mentioned in Annex A.7.

5. ONVIF Client creates Schedule with Schedule token (out *scheduleToken*), with iCalendar value of the Schedule.Standard field (in *scheduleiCalendarValue*) and with skipped SpecialDayGroup element by following the procedure mentioned in Annex A.13.

6. If MaxTimePeriodsPerDay is equal to one, go to step 11.

7. Set the following:

   - *numberPeriodsPerDay* := *cap*.MaxTimePeriodsPerDay

8. ONVIF Client generates appropriate iCalendar value for the Schedule.Standard field (in *cap,* out *scheduleiCalendarValue*) with maximum number of time periods per day (in *numberPeriodsPerDay*) by following the procedure mentioned in Annex A.17.

9. ONVIF client invokes **ModifySchedule** with parameters
    - Schedule.token := *scheduleToken*
    - Schedule.Description := "Test Description"
    - Schedule.Name := "Test Name"
    - Schedule.Standard := *scheduleiCalendarValue*
    - Schedule.SpecialDays skipped
10. The DUT responds with **ModifyScheduleResponse** message
11. If *cap*.MaxTimePeriodsPerDay value is more than 720, go to step 16.
12. Set the following:
    - *numberPeriodsPerDay* := *cap*.MaxTimePeriodsPerDay + 1
13. ONVIF Client generates appropriate iCalendar value for the Schedule.Standard field (in *cap,* out *scheduleiCalendarValue*) with too many number of time periods per day (in *numberPeriodsPerDay*) by following the procedure mentioned in Annex A.17.
14. ONVIF client invokes **ModifySchedule** with parameters
    - Schedule.token := *scheduleToken*
    - Schedule.Description := "Test Description"
    - Schedule.Name := "Test Name"
    - Schedule.Standard := *scheduleiCalendarValue*
    - Schedule.SpecialDays skipped
15. The DUT returns **env:Sender\ter: CapabilityViolated\ter:MaxTimePeriodsPerDay** SOAP 1.2 fault.
16. ONVIF Client deletes the Schedule (in *scheduleToken*) by following the procedure mentioned in Annex A.11 to restore DUT configuration.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send **ModifyScheduleResponse** to schedule modifying with maximum number of time periods per day

The DUT did not send **env:Sender\ter: CapabilityViolated\ter:MaxTimePeriodsPerDay** SOAP 1.2 fault to schedule modifying with too many time periods per day.

Note: If the DUT sends other SOAP 1.2 fault message than specified, log WARNING message, and PASS the test.

4.3.15 MODIFY SCHEDULE - INVALID TIME RANGE INTERVAL

Test Label: Modify Schedule with Invalid Time Range Interval Verification

Test Case ID: SCHEDULE-3-1-15

ONVIF Core Specification Coverage: TimePeriod (ONVIF Schedule Service Specification), ModifySchedule command (ONVIF Schedule Service Specification)

Command Under Test: ModifySchedule

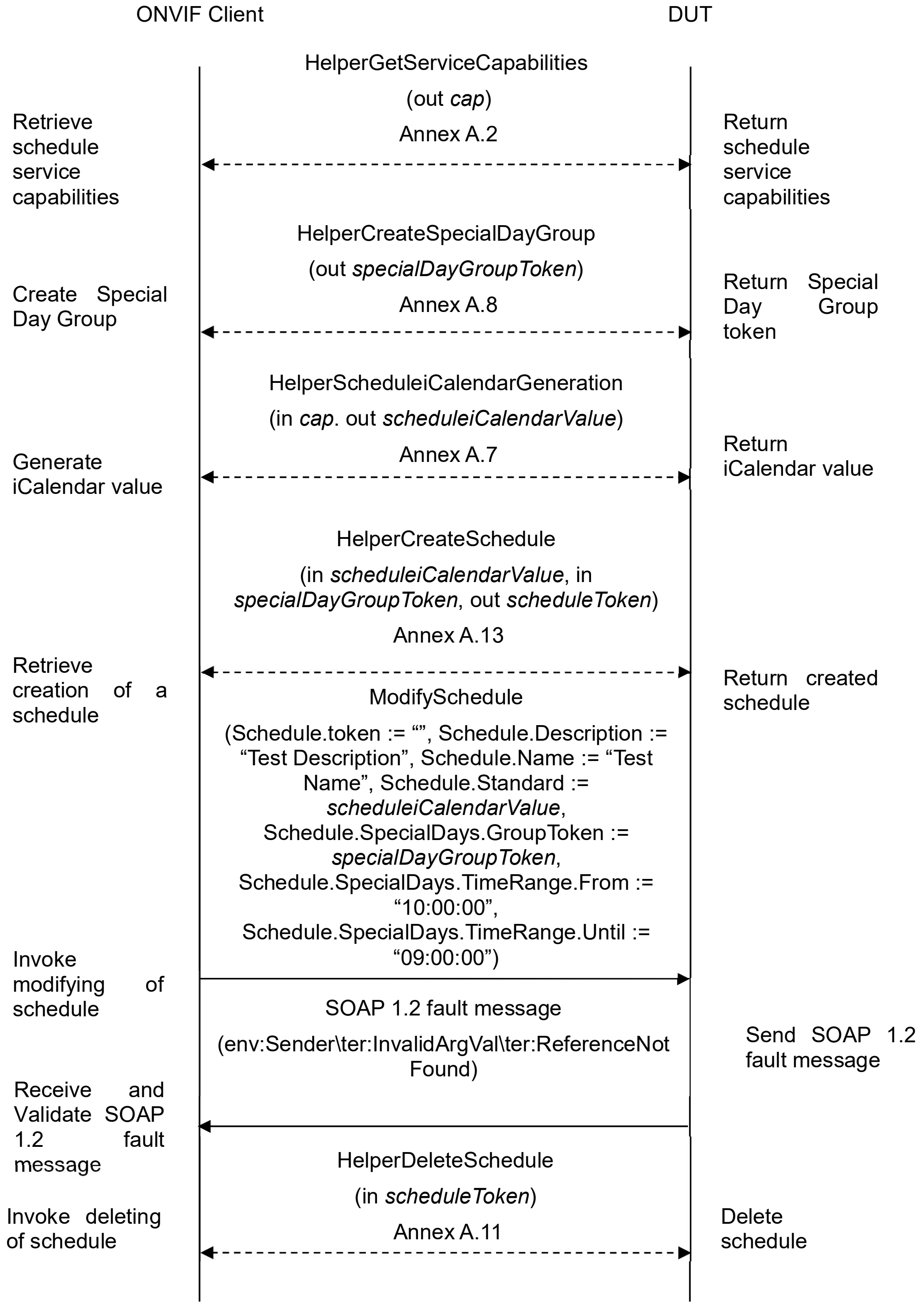
WSDL Reference: schedule.wsdl

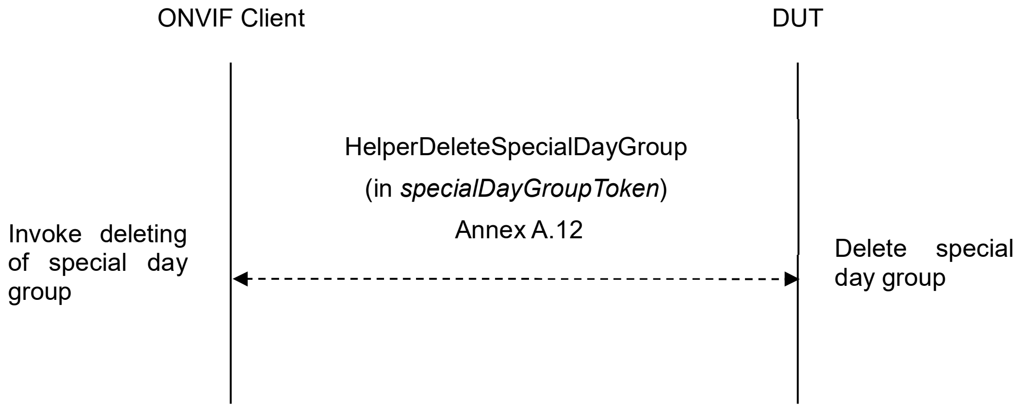
Test Purpose: To verify Modify Schedule with time range interval where the end time is less than the start time.

Pre-requisite: Schedule Service is received from the DUT. The DUT shall have enough free storage capacity for one additional Schedule. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported. The DUT shall have enough free storage capacity for one additional SpecialDayGroup.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieve schedule service capabilities
Return schedule service capabilities
HelperCreateSpecialDayGroup
(out *specialDayGroupToken*)
Annex A.8
Create Special Day Group
Return Special Day Group token
HelperScheduleiCalendarGeneration
(in *cap*. out *scheduleiCalendarValue*)
Annex A.7
Generate iCalendar value
Return iCalendar value
HelperCreateSchedule
(in *scheduleiCalendarValue*, in *specialDayGroupToken*, out *scheduleToken*)
Annex A.13
Retrieve creation of a schedule
Return created schedule
ModifySchedule
(Schedule.token := “”, Schedule.Description := “Test Description”, Schedule.Name := “Test Name”, Schedule.Standard := *scheduleiCalendarValue*, Schedule.SpecialDays.GroupToken := *specialDayGroupToken*, Schedule.SpecialDays.TimeRange.From := “10:00:00”, Schedule.SpecialDays.TimeRange.Until := “09:00:00”)
Invoke modifying of schedule
SOAP 1.2 fault message
(env:Sender\ter:InvalidArgVal\ter:ReferenceNotFound)
Send SOAP 1.2 fault message
Receive and Validate SOAP 1.2 fault message
HelperDeleteSchedule
(in *scheduleToken*)
Annex A.11
Invoke deleting of schedule
Delete schedule


ONVIF Client
DUT
HelperDeleteSpecialDayGroup
(in *specialDayGroupToken*)
Annex A.12
Invoke deleting of special day group
Delete special day group

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

4. ONVIF Client creates SpecialDayGroup (out *specialDayGroupToken*) by following the procedure mentioned in Annex A.8.

5. ONVIF Client generates appropriate iCalendar value for the Schedule.Standard field (in *cap*, out *scheduleiCalendarValue*) by following the procedure mentioned in Annex A.7.

6. ONVIF Client creates Schedule with Schedule token (out *scheduleToken*), with iCalendar value of the Schedule.Standard field (in *scheduleiCalendarValue*) and with SpecialDayGroupToken (in *specialDayGroupToken*) by following the procedure mentioned in Annex A.13.

7. ONVIF client invokes **ModifySchedule** with parameters

   - Schedule.token := “*scheduleToken*”
   - Schedule.Description := “Test Description”
   - Schedule.Name := “Test Name”
   - Schedule.Standard := *scheduleiCalendarValue*
   - Schedule.SpecialDays.GroupToken := *specialDayGroupToken*
   - Schedule.SpecialDays.TimeRange.From := “10:00:00”
   - Schedule.SpecialDays.TimeRange.Until := “09:00:00”

8. The DUT returns **env:Sender\ter:InvalidArgVal\ter:ReferenceNotFound** SOAP 1.2 fault.

9. ONVIF Client deletes the Schedule (in *scheduleToken*) by following the procedure mentioned in

Annex A.11 to restore DUT configuration.

10. ONVIF Client deletes the SpecialDayGroup (in *specialDayGroupToken*) by following the procedure mentioned in Annex A.12 to restore DUT configuration.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

Note: If the DUT sends other SOAP 1.2 fault message than specified, log WARNING message, and PASS the test.

4.3.16 DELETE SCHEDULE WITH INVALID TOKEN

Test Label: Delete Schedule with Invalid Token Verification

Test Case ID: SCHEDULE-3-1-16

ONVIF Core Specification Coverage: DeleteSchedule command (ONVIF Schedule Service Specification)

Command Under Test: DeleteSchedule

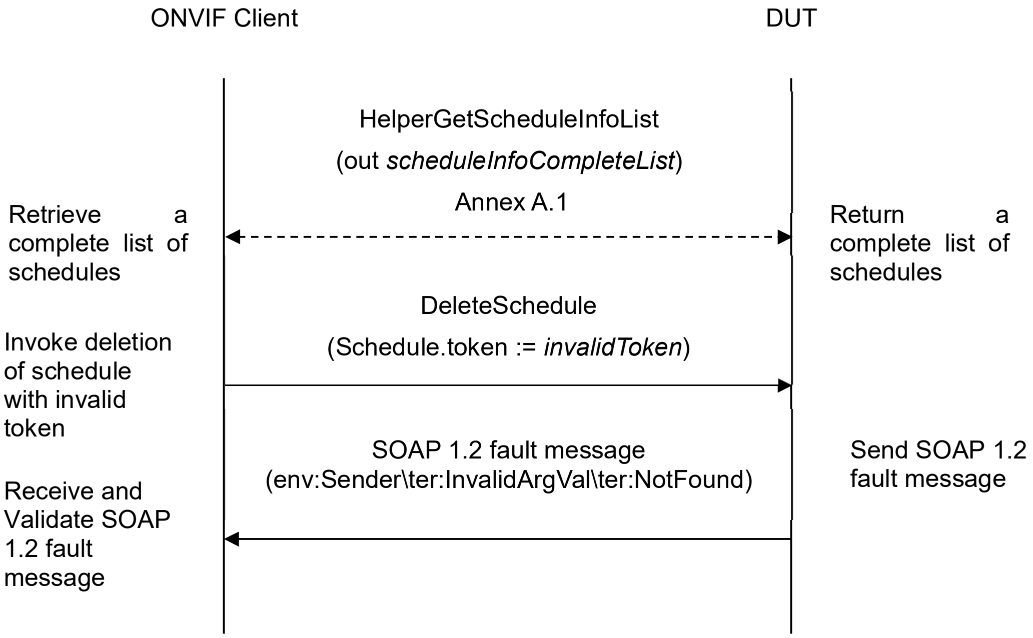
WSDL Reference: schedule.wsdl

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

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

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetScheduleInfoList
(out *scheduleInfoCompleteList*)
Annex A.1
Retrieve a complete list of schedules
Return a complete list of schedules
DeleteSchedule
(Schedule.token := *invalidToken*)
Invoke deletion of schedule with invalid token
SOAP 1.2 fault message
(env:Sender\ter:InvalidArgVal\ter:NotFound)
Send SOAP 1.2 fault message
Receive and Validate SOAP 1.2 fault message

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client retrieves a complete list of schedule info (out *scheduleInfoCompleteList*) by following the procedure mentioned in Annex 0.
4. Set the following:

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

5. ONVIF Client invokes **DeleteSchedule** with parameters
   - Token := *invalidToken*
6. The DUT returns **env:Sender\ter:InvalidArgVal\ter:NotFound** SOAP 1.2 fault.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

Note: If the DUT sends other SOAP 1.2 fault message than specified, log WARNING message, and PASS the test.

4.4 Special Day Group Info

4.4.1 GET SPECIAL DAY GROUP INFO

Test Label: Get Special Day Group Info Verification

Test Case ID: SCHEDULE-4-1-1

ONVIF Core Specification Coverage: SpecialDayGroupInfo (ONVIF Schedule Service Specification), GetSpecialDayGroupInfo command (ONVIF Schedule Service Specification)

Command Under Test: GetSpecialDayGroupInfo

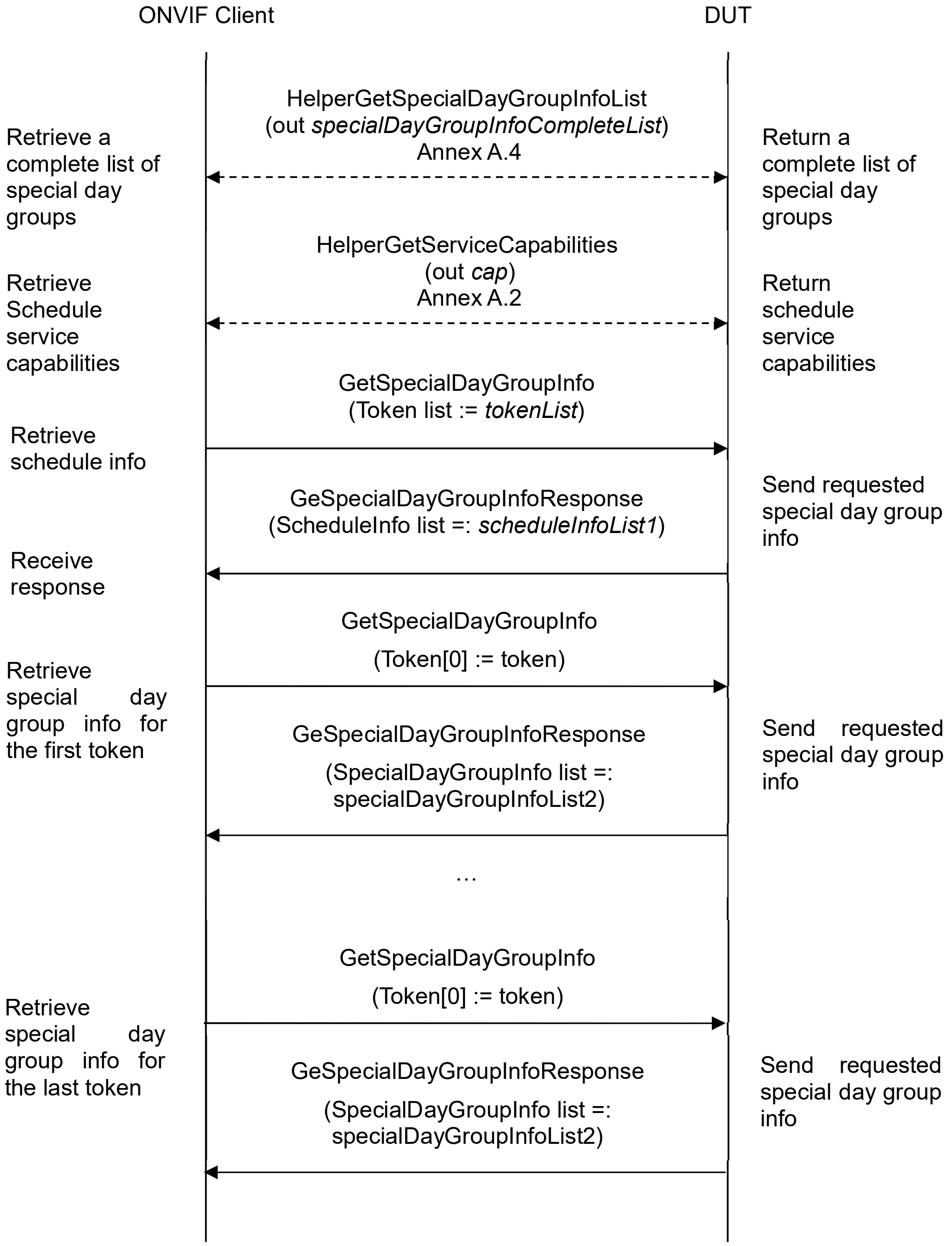
WSDL Reference: schedule.wsdl

Test Purpose: To verify Get Specia Day Group Info.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetSpecialDayGroupInfoList
(out *specialDayGroupInfoCompleteList*)
Annex A.4
Retrieve a complete list of special day groups
Return a complete list of special day groups
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieve Schedule service capabilities
Return schedule service capabilities
GetSpecialDayGroupInfo
(Token list := *tokenList*)
Retrieve schedule info
GeSpecialDayGroupInfoResponse
(ScheduleInfo list =: *scheduleInfoList1*)
Send requested special day group info
Receive response
GetSpecialDayGroupInfo
(Token[0] := token)
Retrieve special day group info for the first token
GeSpecialDayGroupInfoResponse
(SpecialDayGroupInfo list =: specialDayGroupInfoList2)
Send requested special day group info
…
GetSpecialDayGroupInfo
(Token[0] := token)
Retrieve special day group info for the last token
GeSpecialDayGroupInfoResponse
(SpecialDayGroupInfo list =: specialDayGroupInfoList2)
Send requested special day group info

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of special day group information (out *specialDayGroupInfoCompleteList*) by following the procedure mentioned in Annex A.4.

4. If *specialDayGroupInfoCompleteList* is empty, skip other steps.

5. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

6. Set the following:

   - *tokenList* := [subset of *specialDayGroupInfoCompleteList*.token values with items number equal to *cap*.MaxLimit]

7. ONVIF client invokes **GetSpecialDayGroupInfo** with parameters

   - Token list := *tokenList*

8. The DUT responds with **GetSpecialDayGroupInfoResponse** message with parameters

   - SpecialDayGroupInfo list =: *specialDayGroupInfoList1*

9. If *specialDayGroupInfoList1* does not contain SpecialDayGroupInfo item for each token from *tokenList*, FAIL the test and skip other steps.

10. If *specialDayGroupInfoList1* contains at least two SpecialDayGroupInfo items with equal token, FAIL the test and skip other steps.

11. If *specialDayGroupInfoList1* contains other SpecialDayGroupInfo items than listed in *tokenList*, FAIL the test and skip other steps.

12. For each SpecialDayGroupInfo.token *token* from *specialDayGroupInfoCompleteList* repeat the following steps:

    12.1. ONVIF client invokes **GetSpecialDayGroupInfo** with parameters

    - Token[0] := *token*

    12.2. The DUT responds with **GetSpecialDayGroupInfoResponse** message with parameters

    - SpecialDayGroupInfo list =: *specialDayGroupInfoList2*

    12.3. If *specialDayGroupInfoList2* does not contain only one SpecialDayGroupInfo item with token equal to *token*, FAIL the test and skip other steps.

    12.4. If *specialDayGroupInfoList2*[0] item is not equal to *specialDayGroupInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

Note: If number of items in *specialDayGroupInfoCompleteList* is less than *cap*.MaxLimit, then all *specialDayGroupInfoCompleteList*.Token items shall be used for the step 6.

Note: The following fields are compared at step 12.4:

- SpecialDayGroupInfo:
  - token
  - Name
  - Description

4.4.2 GET SPECIAL DAY GROUP INFO LIST - LIMIT

Test Label: Get Special Day Group Info List Verification with Limit

Test Case ID: SCHEDULE-4-1-2

ONVIF Core Specification Coverage: SpecialDayGroupInfo (ONVIF Schedule Service Specification), GetSpecialDayGroupInfoList command (ONVIF Schedule Service Specification)

Command Under Test: GetSpecialDayGroupInfoList

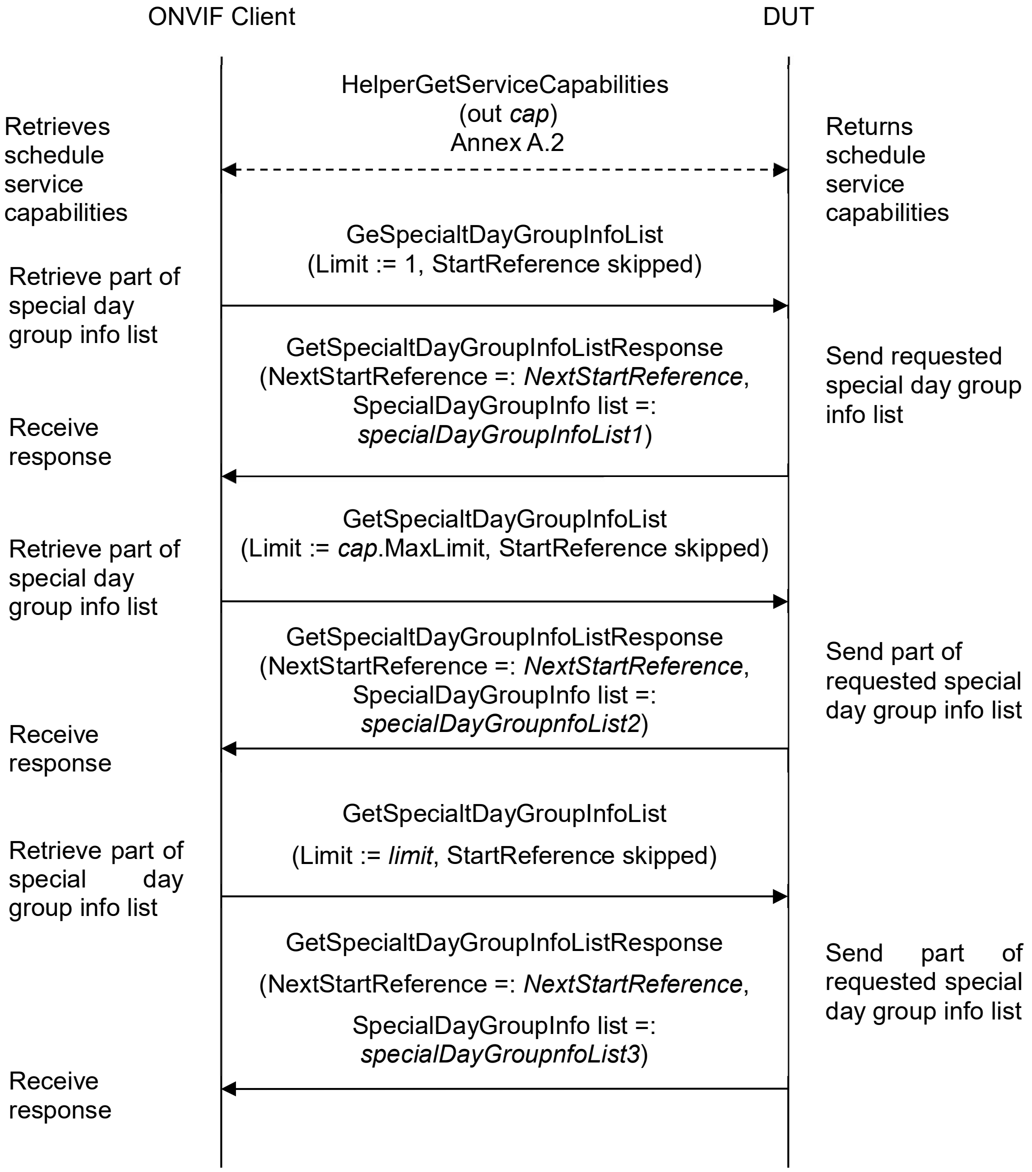
WSDL Reference: schedule.wsdl

Test Purpose: To verify Get Special Day Group Info List using Limit.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieves schedule service capabilities
Returns schedule service capabilities
GeSpecialtDayGroupInfoList
(Limit := 1, StartReference skipped)
Retrieve part of special day group info list
GetSpecialtDayGroupInfoListResponse
(NextStartReference =: *NextStartReference*,
SpecialDayGroupInfo list =:
specialDayGroupInfoList1)
Send requested special day group info list
Receive response
GetSpecialtDayGroupInfoList
(Limit := *cap*.MaxLimit, StartReference skipped)
Retrieve part of special day group info list
GetSpecialtDayGroupInfoListResponse
(NextStartReference =: *NextStartReference*,
SpecialDayGroupInfo list =:
specialDayGroupnfoList2)
Send part of requested special day group info list
Receive response
GetSpecialtDayGroupInfoList
(Limit := *limit*, StartReference skipped)
Retrieve part of special day group info list
GetSpecialtDayGroupInfoListResponse
(NextStartReference =: *NextStartReference*,
SpecialDayGroupInfo list =:
specialDayGroupnfoList3)
Send part of requested special day group info list
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

4. ONVIF client invokes **GetSpecialDayGroupInfoList** with parameters

   - Limit := 1

- StartReference skipped

5. The DUT responds with **GetSpecialDayGroupInfoListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - SpecialDayGroupInfo list =: *specialDayGroupInfoList1*
6. If *specialDayGroupInfoList1* contains more SpecialDayGroupInfo items than 1, FAIL the test and skip other steps.
7. If *cap*.MaxLimit is equal to 1, skip other steps.
8. ONVIF client invokes **GetSpecialDayGroupInfoList** with parameters
   - Limit := *cap*.MaxLimit
   - StartReference skipped
9. The DUT responds with **GetSpecialDayGroupInfoListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - SpecialDayGroupInfo list =: *specialDayGroupInfoList2*
10. If *specialDayGroupInfoList2* contains more SpecialDayGroupInfo items than *cap*.MaxLimit, FAIL the test and skip other steps.
11. If *cap*.MaxLimit is equal to 2, skip other steps.
12. Set the following:
    - *limit* := [number between 1 and *cap*.MaxLimit]
13. ONVIF client invokes **GetSpecialDayGroupInfoList** with parameters
    - Limit := *limit*
    - StartReference skipped
14. The DUT responds with **GetSpecialDayGroupInfoListResponse** message with parameters
    - NextStartReference =: *nextStartReference*
    - SpecialDayGroupInfo list =: *specialDayGroupInfoList3*
15. If *specialDayGroupInfoList3* contains more SpecialDayGroupInfo items than *limit*, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

4.4.3 GET SPECIAL DAY GROUP INFO LIST - START REFERENCE AND LIMIT

Test Label: Get Special Day Group Info List Verification with Start Reference and Limit

Test Case ID: SCHEDULE-4-1-3

ONVIF Core Specification Coverage: SpecialDayGroupInfo (ONVIF Schedule Service Specification), GetSpecialDayGroupInfoList command (ONVIF Schedule Service Specification)

Command Under Test: GetSpecialDayGroupInfoList

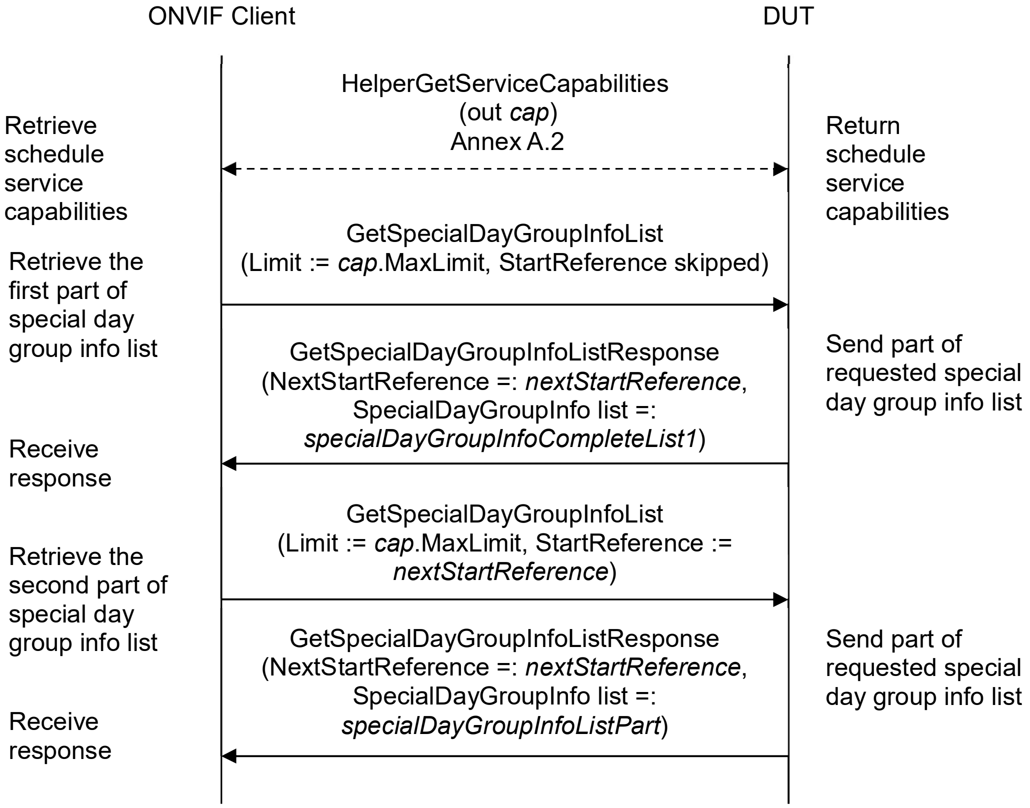
WSDL Reference: schedule.wsdl

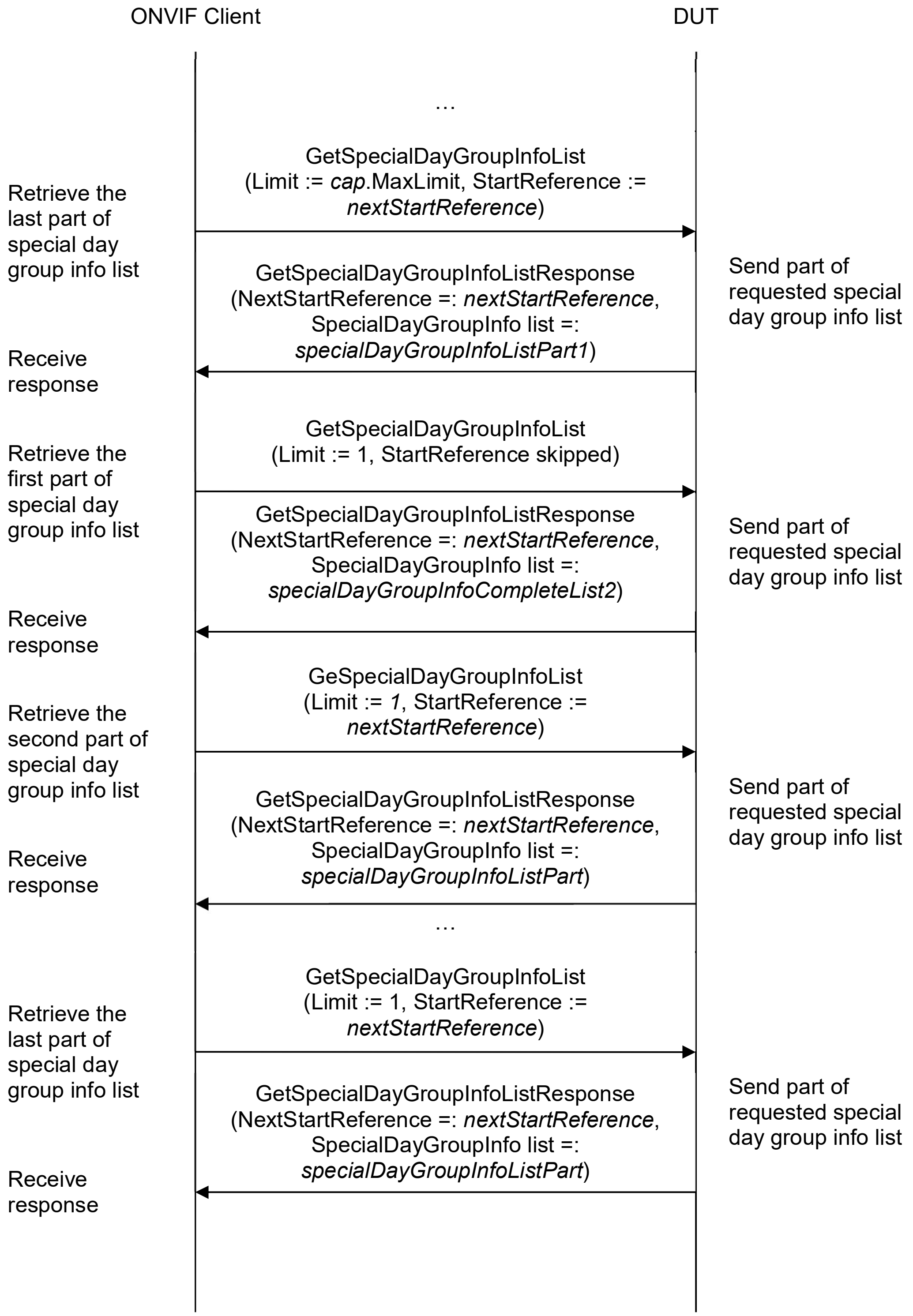
Test Purpose: To verify Get Special Day Group Info List using StartReference and Limit.

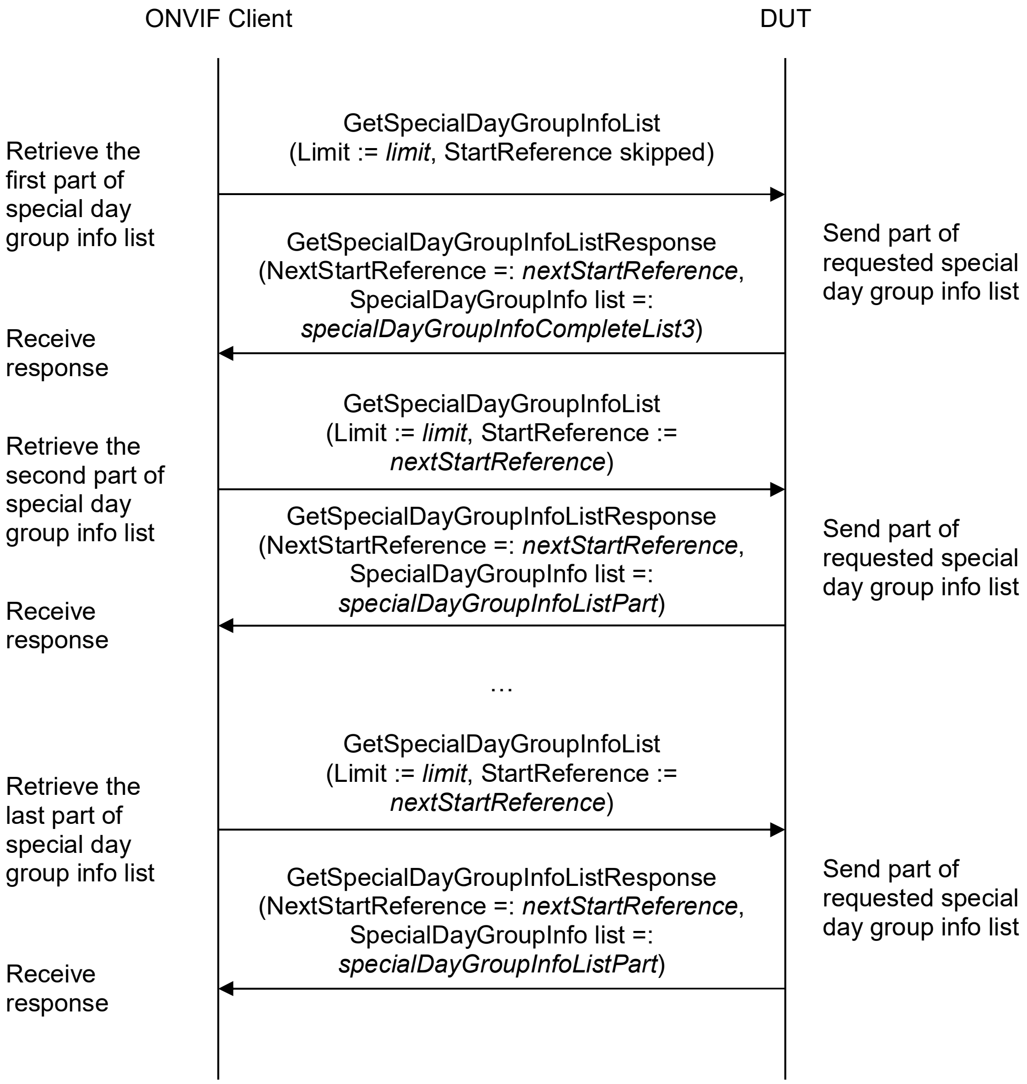
Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out cap)
Annex A.2
Retrieve schedule service capabilities
Return schedule service capabilities
GetSpecialDayGroupInfoList
(Limit := cap.MaxLimit, StartReference skipped)
Retrieve the first part of special day group info list
GetSpecialDayGroupInfoListResponse
(NextStartReference =: nextStartReference,
SpecialDayGroupInfo list =:
specialDayGroupInfoCompleteList1)
Send part of requested special day group info list
Receive response
GetSpecialDayGroupInfoList
(Limit := cap.MaxLimit, StartReference :=
nextStartReference)
Retrieve the second part of special day group info list
GetSpecialDayGroupInfoListResponse
(NextStartReference =: nextStartReference,
SpecialDayGroupInfo list =:
specialDayGroupInfoListPart)
Send part of requested special day group info list
Receive response


ONVIF Client
DUT
…
GetSpecialDayGroupInfoList
(Limit := *cap*.MaxLimit, StartReference :=
nextStartReference)
Retrieve the last part of special day group info list
GetSpecialDayGroupInfoListResponse
(NextStartReference =: *nextStartReference*,
SpecialDayGroupInfo list =:
specialDayGroupInfoListPart1)
Send part of requested special day group info list
Receive response
GetSpecialDayGroupInfoList
(Limit := 1, StartReference skipped)
Retrieve the first part of special day group info list
GetSpecialDayGroupInfoListResponse
(NextStartReference =: *nextStartReference*,
SpecialDayGroupInfo list =:
specialDayGroupInfoCompleteList2)
Send part of requested special day group info list
Receive response
GeSpecialDayGroupInfoList
(Limit := *1*, StartReference :=
nextStartReference)
Retrieve the second part of special day group info list
GetSpecialDayGroupInfoListResponse
(NextStartReference =: *nextStartReference*,
SpecialDayGroupInfo list =:
specialDayGroupInfoListPart)
Send part of requested special day group info list
Receive response
…
GetSpecialDayGroupInfoList
(Limit := 1, StartReference :=
nextStartReference)
Retrieve the last part of special day group info list
GetSpecialDayGroupInfoListResponse
(NextStartReference =: *nextStartReference*,
SpecialDayGroupInfo list =:
specialDayGroupInfoListPart)
Send part of requested special day group info list
Receive response


ONVIF Client
DUT
Retrieve the first part of special day group info list
GetSpecialDayGroupInfoList (Limit := *limit*, StartReference skipped)
GetSpecialDayGroupInfoListResponse (NextStartReference =: *nextStartReference*, SpecialDayGroupInfo list =: *specialDayGroupInfoCompleteList3*)
Send part of requested special day group info list
Receive response
Retrieve the second part of special day group info list
GetSpecialDayGroupInfoList (Limit := *limit*, StartReference := *nextStartReference*)
GetSpecialDayGroupInfoListResponse (NextStartReference =: *nextStartReference*, SpecialDayGroupInfo list =: *specialDayGroupInfoListPart*)
Send part of requested special day group info list
Receive response
…
Retrieve the last part of special day group info list
GetSpecialDayGroupInfoList (Limit := *limit*, StartReference := *nextStartReference*)
GetSpecialDayGroupInfoListResponse (NextStartReference =: *nextStartReference*, SpecialDayGroupInfo list =: *specialDayGroupInfoListPart*)
Send part of requested special day group info list
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

4. ONVIF client invokes **GetSpecialDayGroupInfoList** with parameters

   - Limit := *cap*.MaxLimit

   - StartReference skipped

5. The DUT responds with **GetSpecialDayGroupInfoListResponse** message with parameters

- NextStartReference =: *nextStartReference*
- SpecialDayGroupInfo list =: *specialDayGroupInfoCompleteList1*

6. If *specialDayGroupInfoCompleteList1* contains more SpecialDayGroupInfo items than *cap*.MaxLimit, FAIL the test and skip other steps.

7. Until *nextStartReference* is not null, repeat the following steps:

   7.1. ONVIF client invokes **GetSpecialDayGroupInfoList** with parameters

   - Limit := *cap*.MaxLimit
   - StartReference := *nextStartReference*

   7.2. The DUT responds with **GetSpecialDayGroupInfoListResponse** message with parameters

   - NextStartReference =: *nextStartReference*
   - SpecialDayGroupInfo list =: *specialDayGroupInfoListPart*

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

   7.4. Set the following:

   - *specialDayGroupInfoCompleteList1* := *specialDayGroupInfoCompleteList1* + *specialDayGroupInfoListPart*

8. If *specialDayGroupInfoCompleteList1* contains at least two SpecialDayGroupInfo item with equal token, FAIL the test and skip other steps.

9. If *cap*.MaxLimit is equal to 1, skip other steps.

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

- Limit := 1
- StartReference skipped

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

- NextStartReference =: *nextStartReference*
- SpecialDayGroupInfo list =: *specialDayGroupInfoCompleteList2*

12. If *specialDayGroupInfoCompleteList2* contains more SpecialDayGroupInfo items than 1, FAIL the test and skip other steps.

13. Until *nextStartReference* is not null, repeat the following steps:

   13.1. ONVIF client invokes **GetSpecialDayGroupInfoList** with parameters

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

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

- NextStartReference =: *nextStartReference*
- SpecialDayGroupInfo list =: *specialDayGroupInfoListPart*

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

13.4. Set the following:

- *specialDayGroupInfoCompleteList2* := *specialDayGroupInfoCompleteList2* + *specialDayGroupInfoListPart*

14. If *specialDayGroupInfoCompleteList2* contains at least two SpecialDayGroupInfo item with equal token, FAIL the test and skip other steps.

15. If *specialDayGroupInfoCompleteList2* does not contain all SpecialDayGroupInfo items from *specialDayGroupInfoCompleteList1*, FAIL the test and skip other steps.

16. If *specialDayGroupInfoCompleteList2* contains SpecialDayGroupInfo item other than SpecialDayGroupInfo items from *specialDayGroupInfoCompleteList1*, FAIL the test and skip other steps.

17. If *cap*.MaxLimit is equal to 2, skip other steps.

18. Set the following:

- *limit* := [number between 1 and *cap*.MaxLimit]

19. ONVIF client invokes **GetSpecialDayGroupInfoList** with parameters

- Limit := *limit*
- StartReference skipped

20. The DUT responds with **GetSpecialDayGroupInfoListResponse** message with parameters

- NextStartReference =: *nextStartReference*
- SpecialDayGroupInfo list =: *specialDayGroupInfoCompleteList3*

21. If *specialDayGroupInfoCompleteList3* contains more SpecialDayGroupInfo items than *limit*, FAIL the test and skip other steps.

22. Until *nextStartReference* is not null, repeat the following steps:

22.1. ONVIF client invokes **GetSpecialDayGroupInfoList** with parameters

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

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

- NextStartReference =: *nextStartReference*
- SpecialDayGroupInfo list =: *specialDayGroupInfoListPart*

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

22.4. Set the following:

- *specialDayGroupInfoCompleteList3* := *specialDayGroupInfoCompleteList3* + *specialDayGroupInfoListPart*

23. If *specialDayGroupInfoCompleteList3* contains at least two SpecialDayGroupInfo item with equal token, FAIL the test and skip other steps.

24. If *specialDayGroupInfoCompleteList3* does not contain all SpecialDayGroupInfo items from *specialDayGroupInfoCompleteList1*, FAIL the test and skip other steps.

25. If *specialDayGroupInfoCompleteList3* contains SpecialDayGroupInfo item other than SpecialDayGroupInfo items from *specialDayGroupInfoCompleteList1*, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

4.4.4 GET SPECIAL DAY GROUP INFO LIST - NO LIMIT

Test Label: Get Special Day Group Info List Verification without Limit

Test Case ID: SCHEDULE-4-1-4

ONVIF Core Specification Coverage: SpecialDayGroupInfo (ONVIF Schedule Service Specification), GetSpecialDayGroupInfoList command (ONVIF Schedule Service Specification)

Command Under Test: GetSpecialDayGroupInfoList

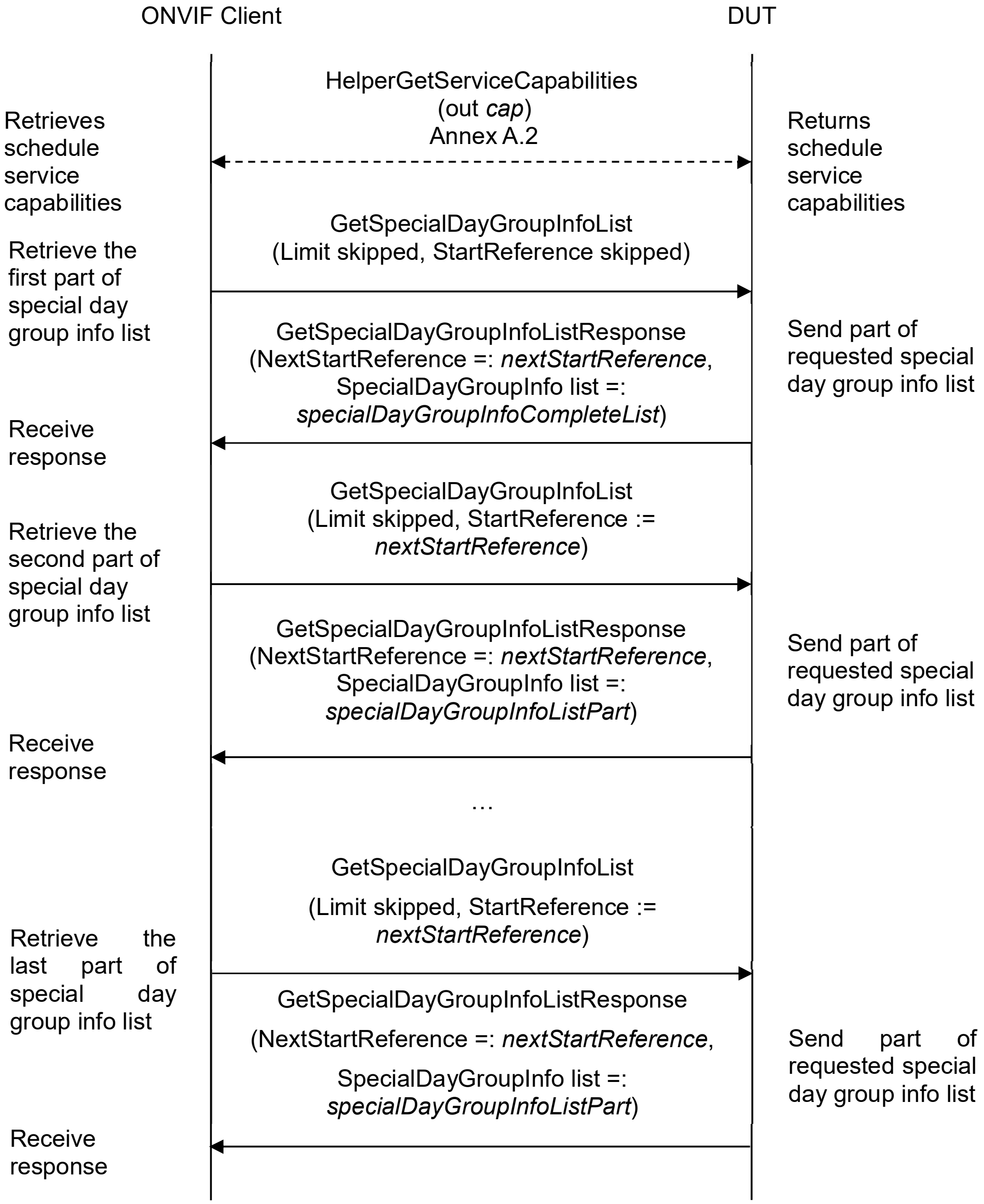
WSDL Reference: schedule.wsdl

Test Purpose: To verify Get Special Day Group Info List without using Limit.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities (out *cap*) Annex A.2
Retrieves schedule service capabilities
Returns schedule service capabilities
GetSpecialDayGroupInfoList (Limit skipped, StartReference skipped)
Retrieve the first part of special day group info list
GetSpecialDayGroupInfoListResponse (NextStartReference =: *nextStartReference*, SpecialDayGroupInfo list =: *specialDayGroupInfoCompleteList*)
Send part of requested special day group info list
Receive response
GetSpecialDayGroupInfoList (Limit skipped, StartReference := *nextStartReference*)
Retrieve the second part of special day group info list
GetSpecialDayGroupInfoListResponse (NextStartReference =: *nextStartReference*, SpecialDayGroupInfo list =: *specialDayGroupInfoListPart*)
Send part of requested special day group info list
Receive response
…
GetSpecialDayGroupInfoList (Limit skipped, StartReference := *nextStartReference*)
Retrieve the last part of special day group info list
GetSpecialDayGroupInfoListResponse (NextStartReference =: *nextStartReference*, SpecialDayGroupInfo list =: *specialDayGroupInfoListPart*)
Send part of requested special day group info list
Receive response

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in

Annex A.2.

4. ONVIF client invokes **GetSpecialDayGroupInfoList** with parameters
   - Limit skipped
   - StartReference skipped
5. The DUT responds with **GetSpecialDayGroupInfoList Response** message with parameters
   - NextStartReference =: *nextStartReference*
   - SpecialDayGroupInfo list =: *specialDayGroupInfoCompleteList*
6. If *specialDayGroupInfoCompleteList* contains more SpecialDayGroupInfo items than *cap*.MaxLimit, FAIL the test and skip other steps.
7. Until *nextStartReference* is not null, repeat the following steps:

   7.1. ONVIF client invokes **GetSpecialDayGroupInfoList** with parameters
   - Limit skipped
   - StartReference := *nextStartReference*

   7.2. The DUT responds with **GetSpecialDayGroupInfoListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - SpecialDayGroupInfo list =: *specialDayGroupInfoListPart*

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

   7.4. Set the following:
   - *specialDayGroupInfoCompleteList* := *specialDayGroupInfoCompleteList* + *specialDayGroupInfoListPart*
8. If *specialDayGroupInfoCompleteList* contains at least two SpecialDayGroupInfo item with equal token, FAIL the test.
9. If *specialDayGroupInfoCompleteList* contains more SpecialDayGroupInfo items than *cap*.MaxSpecialDayGroups, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

4.4.5 GET SPECIAL DAY GROUP INFO WITH INVALID TOKEN

Test Label: Get Special Day Group Info with Invalid Token Verification

Test Case ID: SCHEDULE-4-1-5

ONVIF Core Specification Coverage: SpecialDayGroupInfo (ONVIF Schedule Service Specification), GetSpecialDayGroupInfo command (ONVIF Schedule Service Specification)

Command Under Test: GetSpecialDayGroupInfo

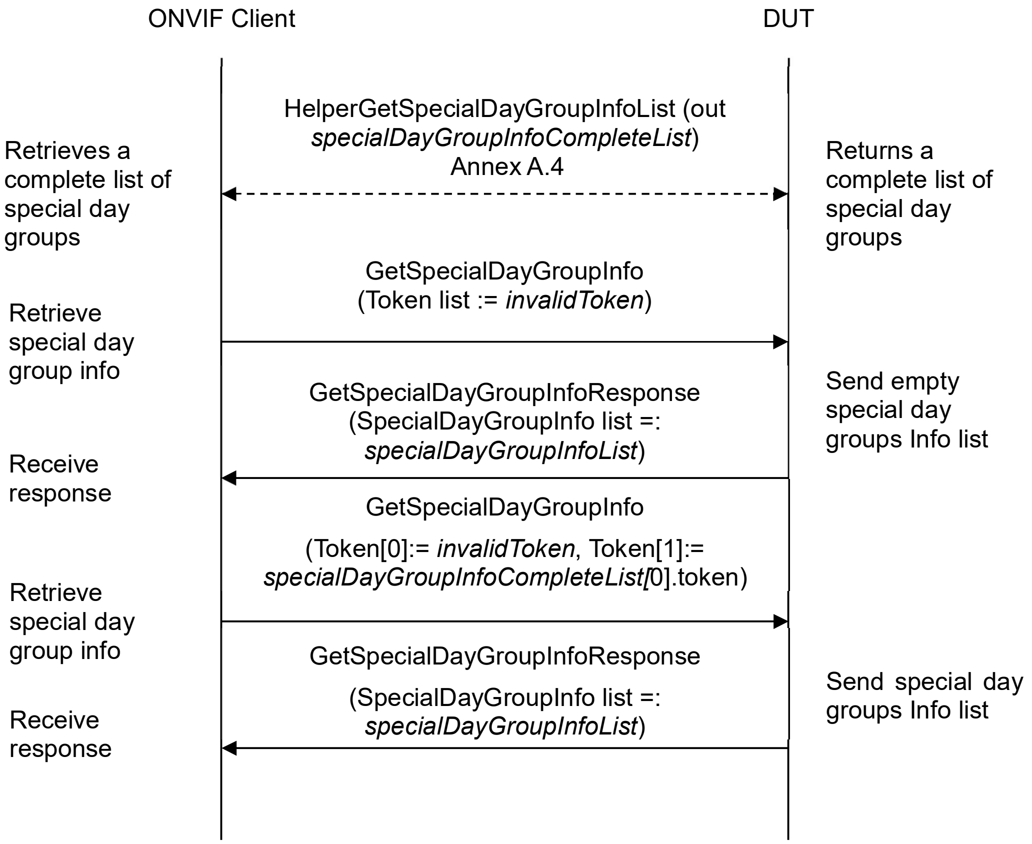
WSDL Reference: schedule.wsdl

Test Purpose: To verify Get Special Day Group Info with invalid token.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Retrieves a complete list of special day groups
HelperGetSpecialDayGroupInfoList (out *specialDayGroupInfoCompleteList*) Annex A.4
Returns a complete list of special day groups
Retrieve special day group info
GetSpecialDayGroupInfo (Token list := *invalidToken*)
GetSpecialDayGroupInfoResponse (SpecialDayGroupInfo list =: *specialDayGroupInfoList*)
Send empty special day groups Info list
Receive response
GetSpecialDayGroupInfo
(Token[0]:= *invalidToken*, Token[1]:= *specialDayGroupInfoCompleteList*[0].token)
Retrieve special day group info
GetSpecialDayGroupInfoResponse
(SpecialDayGroupInfo list =: *specialDayGroupInfoList*)
Send special day groups Info list
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client retrieves a complete list of special day group information (out *specialDayGroupInfoCompleteList*) by following the procedure mentioned in Annex A.4.
4. Set the following:
   - *invalidToken* := value not equal to any *specialDayGroupInfoCompleteList*.token
5. ONVIF client invokes **GetSpecialDayGroupInfo** with parameters
   - Token list := *invalidToken*
6. The DUT responds with **GetSpecialDayGroupInfoResponse** message with parameters
   - SpecialDayGroupInfo list =: *specialDayGroupInfoList*
7. If *specialDayGroupInfoList* is not empty, FAIL the test.
8. If *specialDayGroupInfoCompleteList* is empty, skip other steps.
9. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.
10. If *cap*.MaxLimit is less than 2, skip other steps.
11. ONVIF client invokes **GetSpecialDayGroupInfo** with parameters
    - Token[0]:= *invalidToken*
    - Token[1]:= *specialDayGroupInfoCompleteList*[0].token
12. The DUT responds with **GetSpecialDayGroupInfo** message with parameters
    - SpecialDayGroupInfo list =: *specialDayGroupInfoList*
13. If *specialDayGroupInfoList* is empty, FAIL the test.
14. If *specialDayGroupInfoList* contains more than one item, FAIL the test.
15. If *specialDayGroupInfoList*[0].token does not equal to *specialDayGroupInfoCompleteList*[0].token, FAIL the test.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

4.4.6 GET SPECIAL DAY GROUP INFO - TOO MANY ITEMS

Test Label: Get Special Day Group Info - number of requested items is greater than MaxLimit

Test Case ID: SCHEDULE-4-1-6

ONVIF Core Specification Coverage: SpecialDayGroupInfo (ONVIF Schedule Service Specification), GetSpecialDayGroupInfo command (ONVIF Schedule Service Specification)

Command Under Test: GetSpecialDayGroupInfo

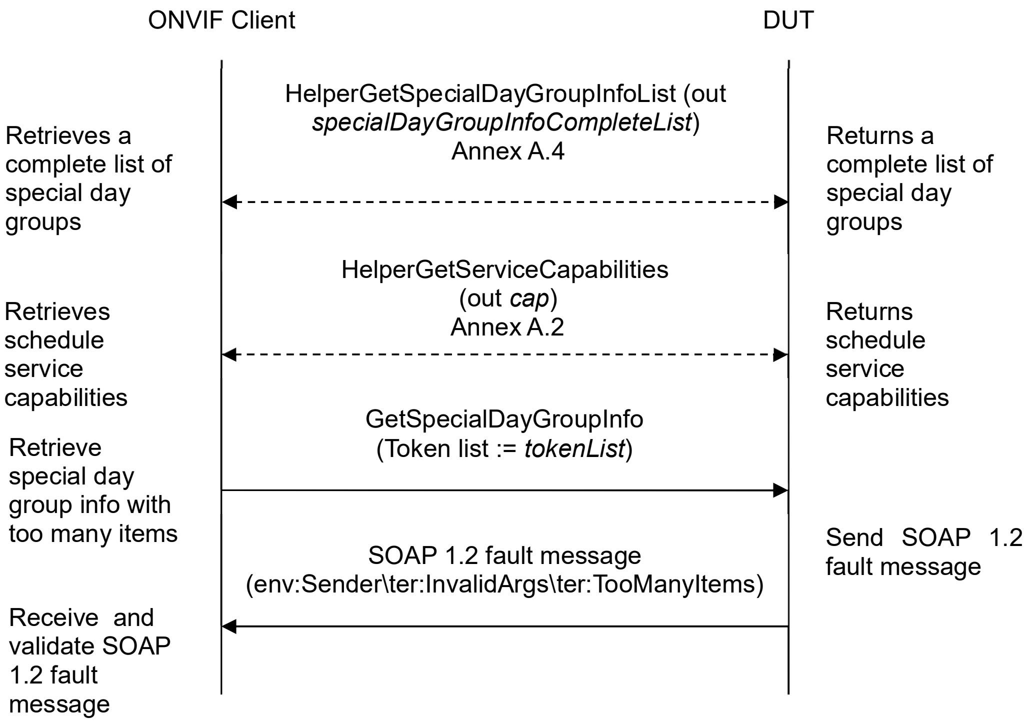
WSDL Reference: schedule.wsdl

Test Purpose: To verify Get Special Day Group Info in case there are more items than MaxLimit in request.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetSpecialDayGroupInfoList (out *specialDayGroupInfoCompleteList*) Annex A.4
Retrieves a complete list of special day groups
Returns a complete list of special day groups
HelperGetServiceCapabilities (out *cap*) Annex A.2
Retrieves schedule service capabilities
Returns schedule service capabilities
GetSpecialDayGroupInfo (Token list := *tokenList*)
Retrieve special day group info with too many items
SOAP 1.2 fault message (env:Sender\ter:InvalidArgs\ter:TooManyItems)
Send SOAP 1.2 fault message
Receive and validate SOAP 1.2 fault message

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of special day group information (out *specialDayGroupInfoCompleteList*) by following the procedure mentioned in Annex A.4.

4. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

5. If *specialDayGroupInfoCompleteList.*token items number is less than *cap*.MaxLimit or equal to *cap*.MaxLimit, skip other steps.

6. Set the following:

    - *tokenList* := [subset of *specialDayGroupInfoCompleteList*.token values with items number equal to *cap*.MaxLimit + 1]

7. ONVIF client invokes **GetSpecialDayGroupInfo** with parameters

    - Token list := *tokenList*

8. The DUT returns **env:Sender\ter:InvalidArgs\ter:TooManyItems** SOAP 1.2 fault.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send env:Sender\ter:InvalidArgs\ter:TooManyItems SOAP 1.2 fault.

4.5 Special Day Group

4.5.1 GET SPECIAL DAY GROUPS

Test Label: Get Special Day Groups Verification

Test Case ID: SCHEDULE-5-1-1

ONVIF Core Specification Coverage: SpecialDayGroup (ONVIF Schedule Service Specification), GetSpecialDayGroups command (ONVIF Schedule Service Specification)

Command Under Test: GetSpecialDayGroups

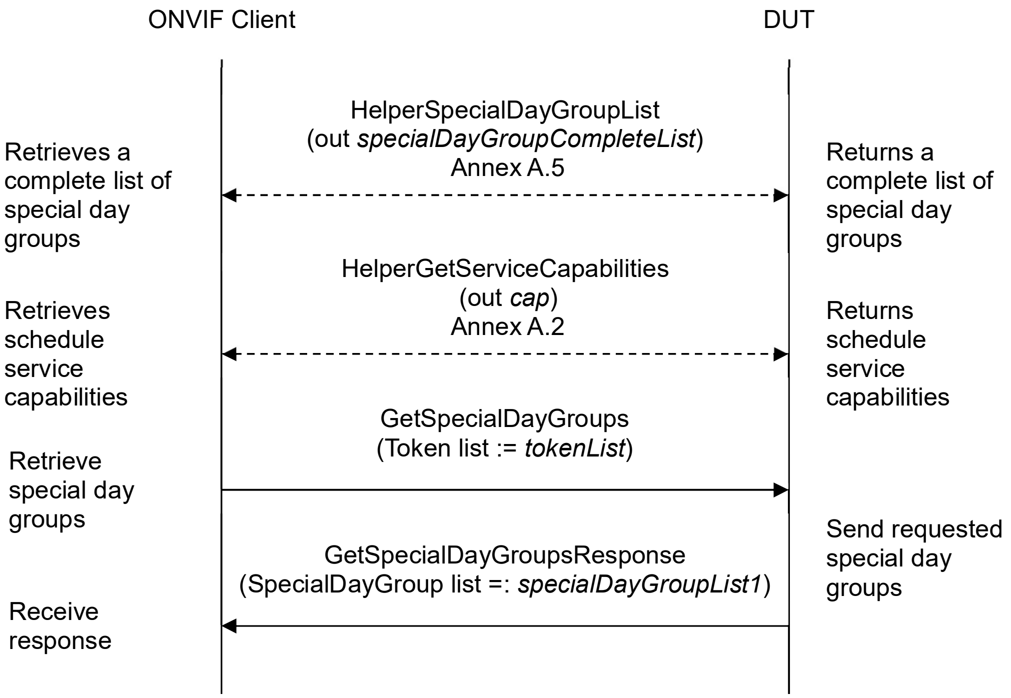
WSDL Reference: schedule.wsdl

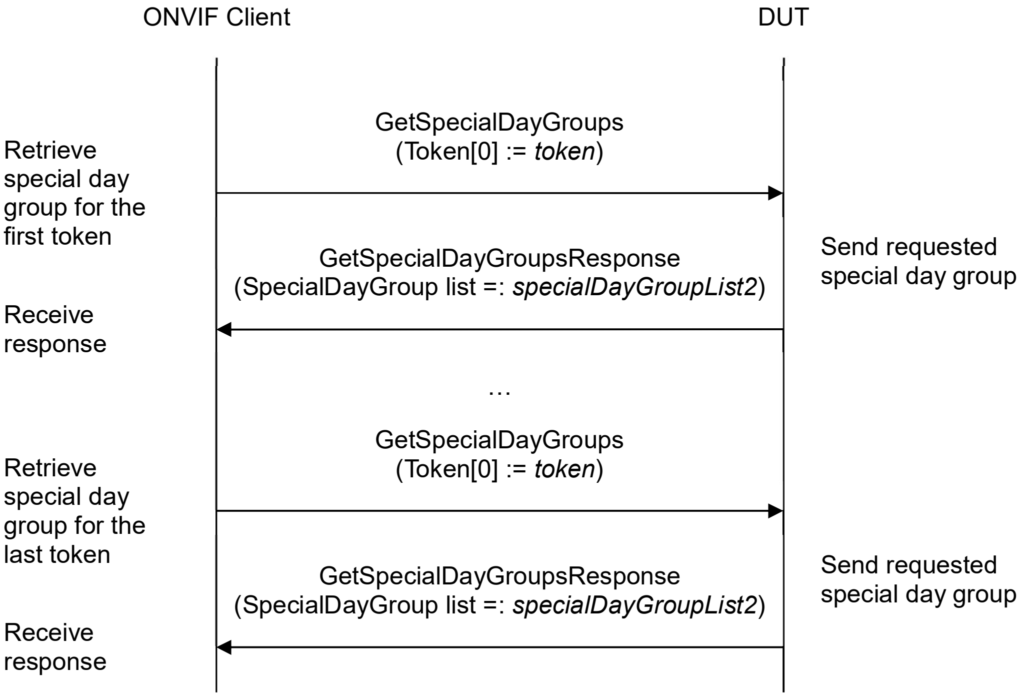
Test Purpose: To verify Get Special Day Groups.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperSpecialDayGroupList
(out *specialDayGroupCompleteList*)
Annex A.5
Retrieves a complete list of special day groups
Returns a complete list of special day groups
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieves schedule service capabilities
Returns schedule service capabilities
GetSpecialDayGroups
(Token list := *tokenList*)
Retrieve special day groups
GetSpecialDayGroupsResponse
(SpecialDayGroup list =: *specialDayGroupList1*)
Send requested special day groups
Receive response


ONVIF Client
DUT
Retrieve special day group for the first token
GetSpecialDayGroups (Token[0] := *token*)
GetSpecialDayGroupsResponse (SpecialDayGroup list =: *specialDayGroupList2*)
Send requested special day group
Receive response
…
Retrieve special day group for the last token
GetSpecialDayGroups (Token[0] := *token*)
GetSpecialDayGroupsResponse (SpecialDayGroup list =: *specialDayGroupList2*)
Send requested special day group
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of Special Day Groups (out *specialDayGroupCompleteList*) by following the procedure mentioned in Annex A.5.

4. If *specialDayGroupCompleteList* is empty, skip other steps.

5. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

6. Set the following:

   - *tokenList* := [subset of *specialDayGroupCompleteList*.token values with items number equal to *cap*.MaxLimit]

7. ONVIF client invokes **GetSpecialDayGroups** with parameters

   - Token list := *tokenList*

8. The DUT responds with **GetSpecialDayGroupsResponse** message with parameters

   - SpecialDayGroup list =: *specialDayGroupList1*

9. If *specialDayGroupList1* does not contain SpecialDayGroup item for each token from *tokenList*, FAIL the test and skip other steps.

10. If *specialDayGroupList1* contains at least two SpecialDayGroup items with equal token, FAIL the test and skip other steps.

11. If *specialDayGroupList1* contains other SpecialDayGroup items than listed in *tokenList*, FAIL the test and skip other steps.

12. For each SpecialDayGroup.token *token* from *specialDayGroupCompleteList* repeat the following steps:

    12.1. ONVIF client invokes **GetSpecialDayGroups** with parameters

    - Token[0] := *token*

    12.2. The DUT responds with **GetSpecialDayGroupsResponse** message with parameters

    - Schedule list =: *specialDayGroupList2*

    12.3. If *specialDayGroupList2* does not contain only one SpecialDayGroup item with token equal to *token*, FAIL the test and skip other steps.

    12.4. If *specialDayGroupList2*[0] item does not have equal field values to *specialDayGroup CompleteList*[token = *token*] item, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

Note: If number of items in *specialDayGroupCompleteList* is less than *cap*.MaxLimit, then all *specialDayGroupCompleteList*.Token items shall be used for the step 6.

Note: The following fields are compared at step 12.4:

- Schedule:
    - token
    - Name
    - Description
    - Days

4.5.2 GET SPECIAL DAY GROUP LIST - LIMIT

Test Label: Get Special Day Group List Verification with Limit

Test Case ID: SCHEDULE-5-1-2

ONVIF Core Specification Coverage: SpecialDayGroup (ONVIF Schedule Service Specification), GetSpecialDayGroupList command (ONVIF Schedule Service Specification)

Command Under Test: GetSpecialDayGroupList

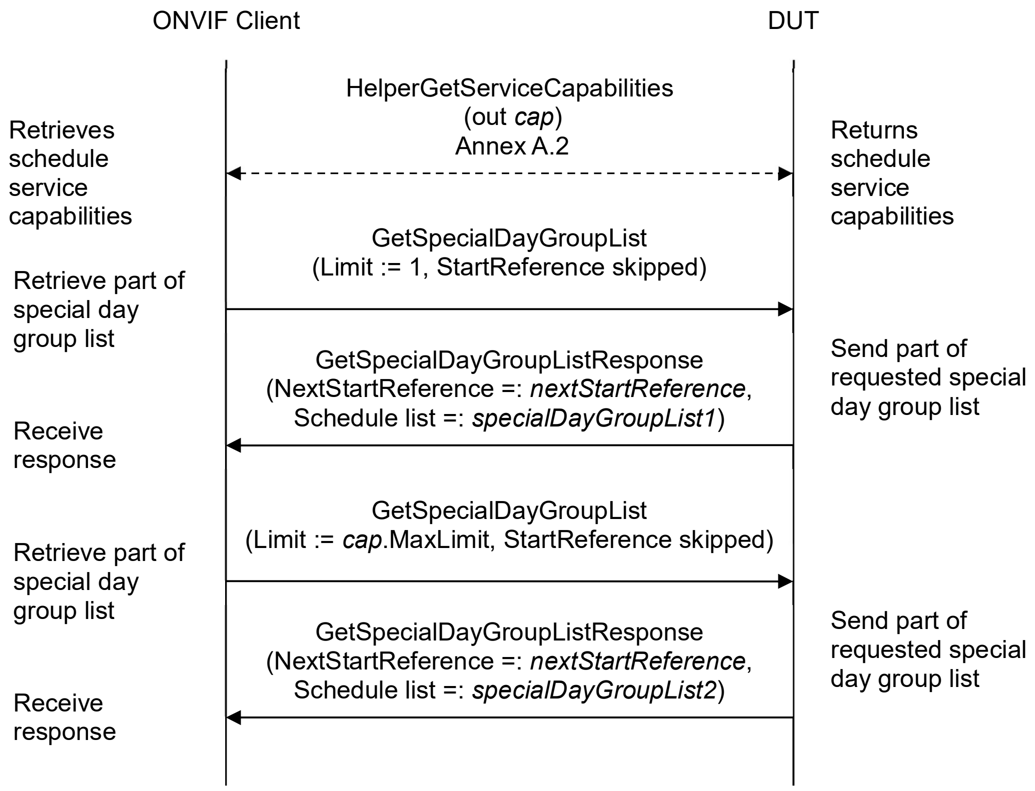
WSDL Reference: schedule.wsdl

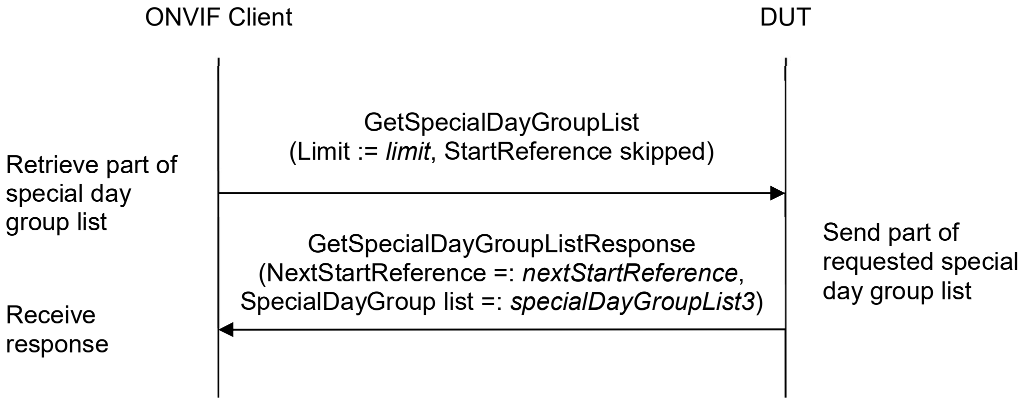
Test Purpose: To verify Get SpecialDayGroup List using Limit.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieves schedule service capabilities
Returns schedule service capabilities
GetSpecialDayGroupList
(Limit := 1, StartReference skipped)
Retrieve part of special day group list
GetSpecialDayGroupListResponse
(NextStartReference =: *nextStartReference*,
Schedule list =: *specialDayGroupList1*)
Send part of requested special day group list
Receive response
GetSpecialDayGroupList
(Limit := *cap*.MaxLimit, StartReference skipped)
Retrieve part of special day group list
GetSpecialDayGroupListResponse
(NextStartReference =: *nextStartReference*,
Schedule list =: *specialDayGroupList2*)
Send part of requested special day group list
Receive response


ONVIF Client
DUT
Retrieve part of special day group list
GetSpecialDayGroupList (Limit := *limit*, StartReference skipped)
GetSpecialDayGroupListResponse (NextStartReference =: *nextStartReference*, SpecialDayGroup list =: *specialDayGroupList3*)
Send part of requested special day group list
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

4. ONVIF client invokes **GetSpecialDayGroupList** with parameters

   - Limit := 1

   - StartReference skipped

5. The DUT responds with **GetSpecialDayGroupListResponse** message with parameters

   - NextStartReference =: *nextStartReference*

   - SpecialDayGroup list =: *specialDayGroupList1*

6. If *specialDayGroupList1* contains more SpecialDayGroup items than 1, FAIL the test and skip other steps.

7. If *cap*.MaxLimit is equal to 1, skip other steps.

8. ONVIF client invokes **GetSpecialDayGroupList** with parameters

   - Limit := *cap*.MaxLimit

   - StartReference skipped

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

   - NextStartReference =: *nextStartReference*

   - SpecialDayGroup list =: *specialDayGroupList2*

10. If *specialDayGroupList2* contains more SpecialDayGroup items than *cap*.MaxLimit, FAIL the test and skip other steps.

11. If *cap*.MaxLimit is equal to 2, skip other steps.

12. Set the following:

    - *limit* := [number between 1 and *cap*.MaxLimit]

13. ONVIF client invokes **GetSpecialDayGroupList** with parameters

    - Limit := *limit*

    - StartReference skipped

14. The DUT responds with **GetSpecialDayGroupListResponse** message with parameters

    - NextStartReference =: *nextStartReference*

    - SpecialDayGroup list =: *specialDayGroupList3*

15. If *specialDayGroupList3* contains more SpecialDayGroup items than *limit*, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

4.5.3 GET SPECIAL DAY GROUP LIST - START REFERENCE AND LIMIT

Test Label: Get Special Day Group List Verification with Start Reference and Limit

Test Case ID: SCHEDULE-5-1-3

ONVIF Core Specification Coverage: SpecialDayGroup (ONVIF Schedule Service Specification), GetSpecialDayGroupList command (ONVIF Schedule Service Specification)

Command Under Test: GetSpecialDayGroupList

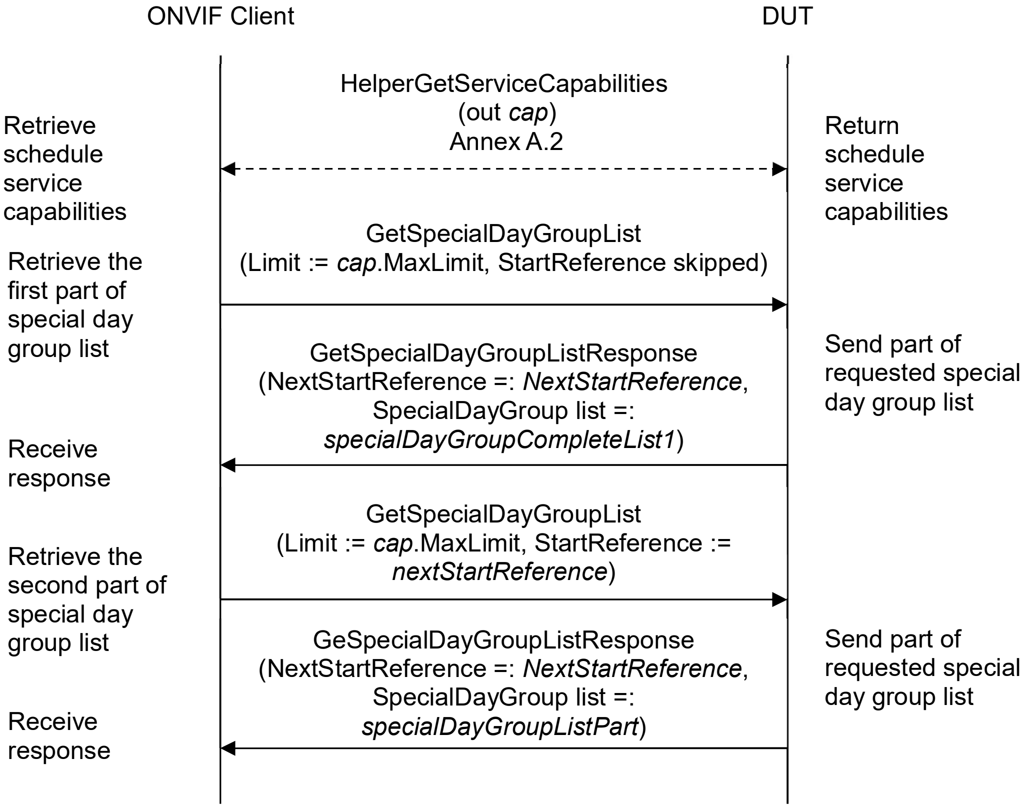
WSDL Reference: schedule.wsdl

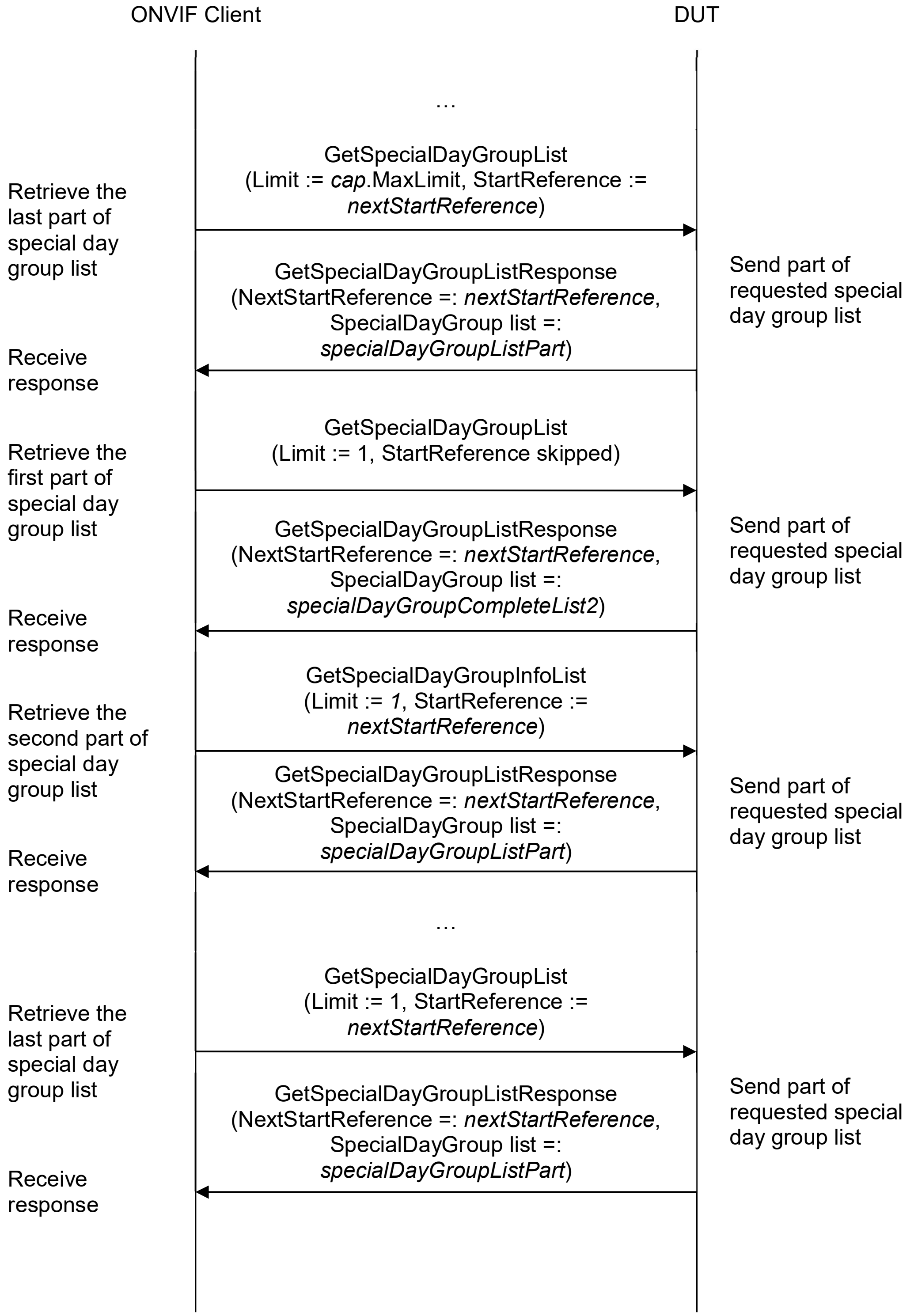
Test Purpose: To verify Get SpecialDayGroup List using StartReference and Limit.

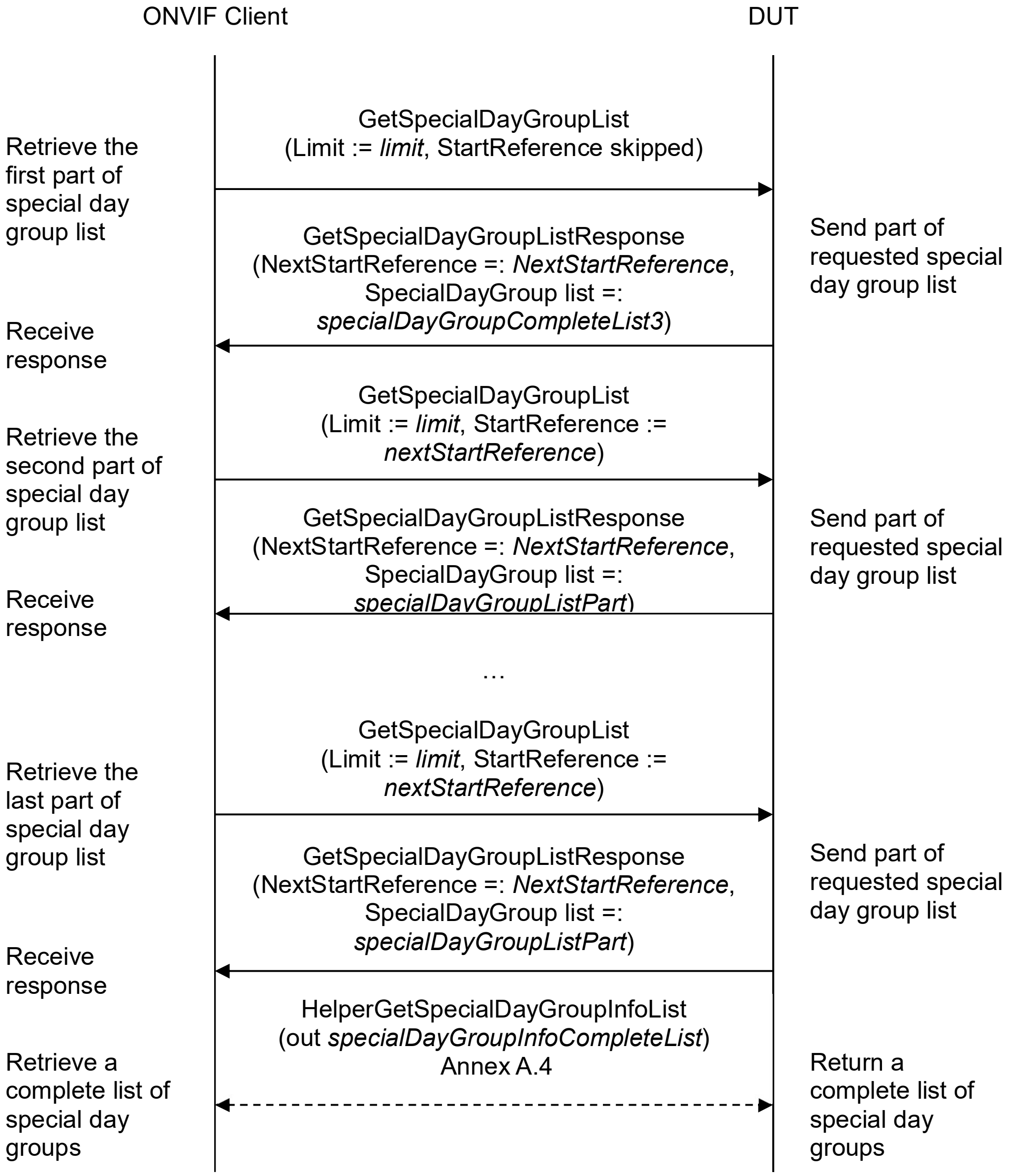
Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Retrieve schedule service capabilities
HelperGetServiceCapabilities (out *cap*) Annex A.2
Return schedule service capabilities
Retrieve the first part of special day group list
GetSpecialDayGroupList (Limit := *cap*.MaxLimit, StartReference skipped)
GetSpecialDayGroupListResponse (NextStartReference =: *NextStartReference*, SpecialDayGroup list =: *specialDayGroupCompleteList1*)
Send part of requested special day group list
Receive response
Retrieve the second part of special day group list
GetSpecialDayGroupList (Limit := *cap*.MaxLimit, StartReference := *nextStartReference*)
GeSpecialDayGroupListResponse (NextStartReference =: *NextStartReference*, SpecialDayGroup list =: *specialDayGroupListPart*)
Send part of requested special day group list
Receive response


ONVIF Client
DUT
…
GetSpecialDayGroupList
(Limit := *cap*.MaxLimit, StartReference :=
nextStartReference)
Retrieve the last part of special day group list
GetSpecialDayGroupListResponse
(NextStartReference =: *nextStartReference*,
SpecialDayGroup list =:
specialDayGroupListPart)
Send part of requested special day group list
Receive response
GetSpecialDayGroupList
(Limit := 1, StartReference skipped)
Retrieve the first part of special day group list
GetSpecialDayGroupListResponse
(NextStartReference =: *nextStartReference*,
SpecialDayGroup list =:
specialDayGroupCompleteList2)
Send part of requested special day group list
Receive response
GetSpecialDayGroupInfoList
(Limit := *1*, StartReference :=
nextStartReference)
Retrieve the second part of special day group list
GetSpecialDayGroupListResponse
(NextStartReference =: *nextStartReference*,
SpecialDayGroup list =:
specialDayGroupListPart)
Send part of requested special day group list
Receive response
…
GetSpecialDayGroupList
(Limit := 1, StartReference :=
nextStartReference)
Retrieve the last part of special day group list
GetSpecialDayGroupListResponse
(NextStartReference =: *nextStartReference*,
SpecialDayGroup list =:
specialDayGroupListPart)
Send part of requested special day group list
Receive response


ONVIF Client
DUT
Retrieve the first part of special day group list
GetSpecialDayGroupList (Limit := *limit*, StartReference skipped)
GetSpecialDayGroupListResponse (NextStartReference =: *NextStartReference*, SpecialDayGroup list =: *specialDayGroupCompleteList3*)
Send part of requested special day group list
Receive response
Retrieve the second part of special day group list
GetSpecialDayGroupList (Limit := *limit*, StartReference := *nextStartReference*)
GetSpecialDayGroupListResponse (NextStartReference =: *NextStartReference*, SpecialDayGroup list =: *specialDayGroupListPart*)
Send part of requested special day group list
Receive response
…
Retrieve the last part of special day group list
GetSpecialDayGroupList (Limit := *limit*, StartReference := *nextStartReference*)
GetSpecialDayGroupListResponse (NextStartReference =: *NextStartReference*, SpecialDayGroup list =: *specialDayGroupListPart*)
Send part of requested special day group list
Receive response
Retrieve a complete list of special day groups
HelperGetSpecialDayGroupInfoList (out *specialDayGroupInfoCompleteList*) Annex A.4
Return a complete list of special day groups

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

4. ONVIF client invokes **GetSpecialDayGroupList** with parameters

- Limit := *cap*.MaxLimit
- StartReference skipped

5. The DUT responds with **GetSpecialDayGroupListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - SpecialDayGroup list =: *specialDayGroupCompleteList1*

6. If *specialDayGroupCompleteList1* contains more SpecialDayGroup items than *cap*.MaxLimit, FAIL the test and skip other steps.

7. Until *nextStartReference* is not null, repeat the following steps:

   7.1. ONVIF client invokes **GetSpecialDayGroupList** with parameters
   - Limit := *cap*.MaxLimit
   - StartReference := *nextStartReference*

   7.2. The DUT responds with **GetSpecialDayGroupListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - SpecialDayGroup list =: *specialDayGroupListPart*

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

   7.4. Set the following:
   - *specialDayGroupCompleteList1* := *specialDayGroupCompleteList1* + *specialDayGroupListPart*

8. If *specialDayGroupCompleteList1* contains at least two SpecialDayGroup item with equal token, FAIL the test and skip other steps.

9. If *cap*.MaxLimit is equal to 1, do the following steps:

   9.1. ONVIF Client retrieves a complete list of SpecialDayGroupInfo (out *specialDayGroupInfoCompleteList*) by following the procedure mentioned in Annex A.4.

   9.2. If *specialDayGroupCompleteList1* does not contain all tokens from *specialDayGroupInfoCompleteList*, FAIL the test and skip other steps.

   9.3. If *specialDayGroupCompleteList1* contains tokens other than tokens from *specialDayGroupInfoCompleteList*, FAIL the test and skip other steps.

   9.4. For each SpecialDayGroupInfo.token *token* from *specialDayGroupInfoCompleteList* repeat the following steps:

   9.4.1. If *specialDayGroupCompleteList1*[token = *token*] item does not have equal field values to *specialDayGroupInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.

   9.5. Skip other steps.

10. ONVIF client invokes **GetSpecialDayGroupList** with parameters
    - Limit := 1
    - StartReference skipped
11. The DUT responds with **GetSpecialDayGroupListResponse** message with parameters
    - NextStartReference =: *nextStartReference*
    - SpecialDayGroup list =: *specialDayGroupCompleteList2*
12. If *specialDayGroupCompleteList2* contains more SpecialDayGroup items than 1, FAIL the test and skip other steps.
13. Until *nextStartReference* is not null, repeat the following steps:

    13.1. ONVIF client invokes **GetSpecialDayGroupList** with parameters
    - Limit := 1
    - StartReference := *nextStartReference*

    13.2. The DUT responds with **GetSpecialDayGroupListResponse** message with parameters
    - NextStartReference =: *nextStartReference*
    - SpecialDayGroup list =: *specialDayGroupListPart*

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

    13.4. Set the following:
    - *specialDayGroupCompleteList2* := *specialDayGroupCompleteList2* + *specialDayGroupListPart*
14. If *specialDayGroupCompleteList2* contains at least two SpecialDayGroup item with equal token, FAIL the test and skip other steps.
15. If *specialDayGroupCompleteList2* does not contain all SpecialDayGroup items from *specialDayGroupCompleteList1*, FAIL the test and skip other steps.
16. If *specialDayGroupCompleteList2* contains SpecialDayGroup items other than SpecialDayGroup items from *specialDayGroupCompleteList1*, FAIL the test and skip other steps.
17. If *cap*.MaxLimit is equal to 2 do the following steps:

    17.1. ONVIF Client retrieves a complete list of SpecialDayGroupInfo (out *specialDayGroupInfoCompleteList*) by following the procedure mentioned in Annex A.4.

    17.2. If *specialDayGroupCompleteList2* does not contain all tokens from *specialDayGroupInfoCompleteList*, FAIL the test and skip other steps.

    17.3. If *specialDayGroupCompleteList2* contains tokens other than tokens from *specialDayGroupInfoCompleteList*, FAIL the test and skip other steps.

    17.4. For each SpecialDayGroupInfo.token *token* from *specialDayGroupInfoCompleteList* repeat the following steps:

17.4.1. If *specialDayGroupCompleteList2*[token = *token*] item does not have equal field values to *specialDayGroupInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.

17.5. Skip other steps.

18. Set the following:

- *limit* := [number between 1 and *cap*.MaxLimit]

19. ONVIF client invokes **GetSpecialDayGroupList** with parameters

- Limit := *limit*
- StartReference skipped

20. The DUT responds with **GetSpecialDayGroupListResponse** message with parameters

- NextStartReference =: *nextStartReference*
- Schedule list =: *specialDayGroupCompleteList3*

21. If *specialDayGroupCompleteList3* contains more SpecialDayGroup items than *limit*, FAIL the test and skip other steps.

22. Until *nextStartReference* is not null, repeat the following steps:

22.1. ONVIF client invokes **GetSpecialDayGroupList** with parameters

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

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

- NextStartReference =: *nextStartReference*
- Schedule list =: *specialDayGroupListPart*

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

22.4. Set the following:

- *specialDayGroupCompleteList3* := *specialDayGroupCompleteList3* + *specialDayGroupListPart*

23. If *specialDayGroupCompleteList3* contains at least two SpecialDayGroup item with equal token, FAIL the test and skip other steps.

24. If *specialDayGroupCompleteList3* does not contain all SpecialDayGroup items from *specialDayGroupCompleteList1*, FAIL the test and skip other steps.

25. If *specialDayGroupCompleteList3* contains SpecialDayGroup items other than schedules from *specialDayGroupCompleteList1*, FAIL the test and skip other steps.

26. ONVIF Client retrieves a complete list of SpecialDayGroupInfo (out *specialDayGroupInfoCompleteList*) by following the procedure mentioned in Annex A.4.

27. If *specialDayGroupCompleteList3* does not contain all tokens from *specialDayGroupInfoCompleteList*, FAIL the test and skip other steps.

28. If *specialDayGroupCompleteList3* contains tokens other than tokens from *specialDayGroupInfoCompleteList*, FAIL the test and skip other steps.

29. For each SpecialDayGroupInfo.token *token* from *specialDayGroupInfoCompleteList* repeat the following steps:

    29.1. If *specialDayGroupCompleteList3*[token = *token*] item does not have equal field values to *specialDayGroupInfoCompleteList*[token = *token*] item, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

Note: The following fields are compared at steps 9.4.1, 17.4.1, 29.1:

- SpecialDayGroupInfo:
    - token
    - Name
    - Description

4.5.4 GET SPECIAL DAY GROUP LIST - NO LIMIT

Test Label: Get Special Day Group List Verification without Limit

Test Case ID: SCHEDULE-5-1-4

ONVIF Core Specification Coverage: SpecialDayGroup (ONVIF Schedule Service Specification), GetSpecialDayGroupList command (ONVIF Schedule Service Specification)

Command Under Test: GetSpecialDayGroupList

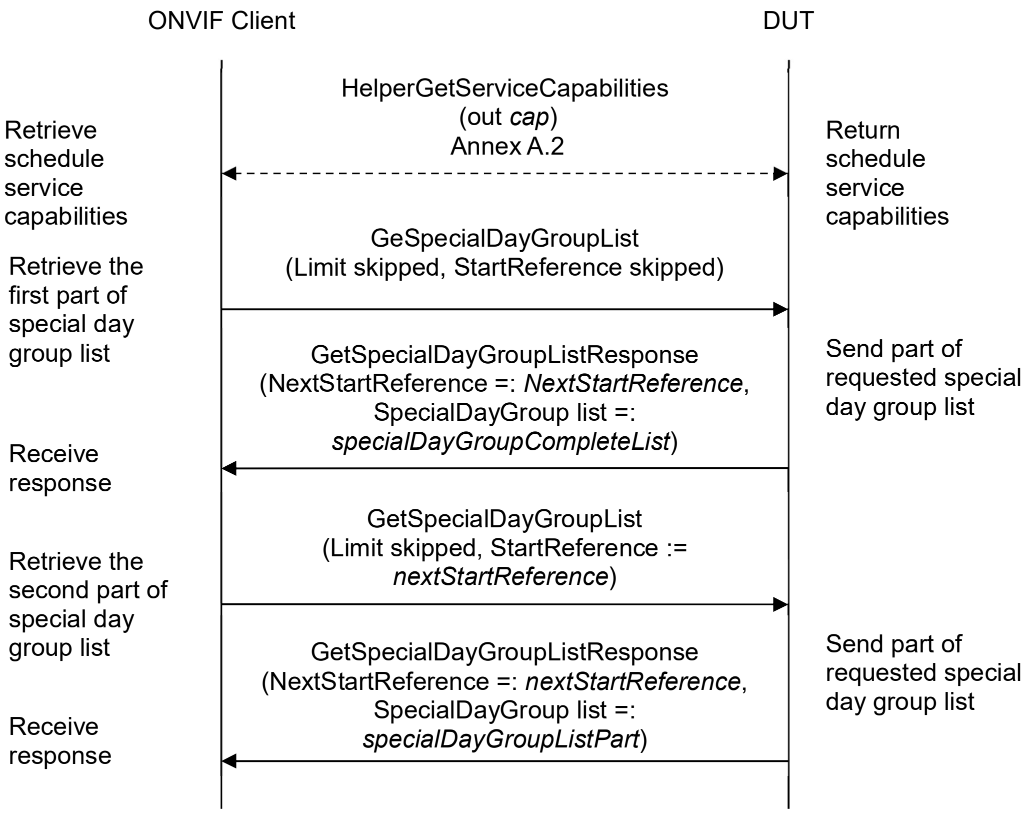
WSDL Reference: schedule.wsdl

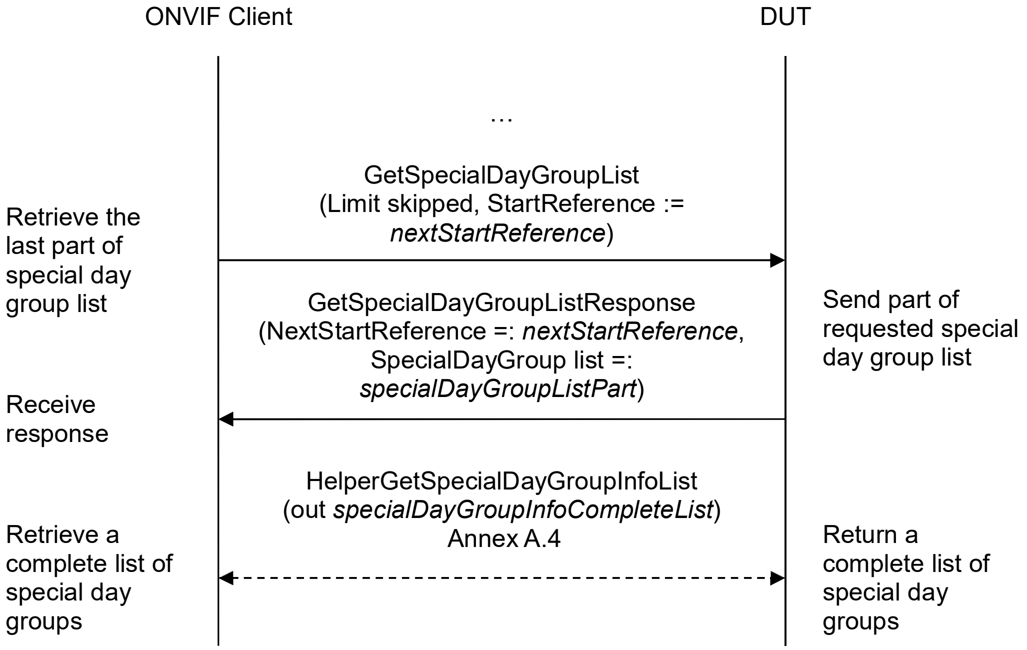
Test Purpose: To verify Get SpecialDayGroup List without using Limit.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieve schedule service capabilities
Return schedule service capabilities
GeSpecialDayGroupList
(Limit skipped, StartReference skipped)
Retrieve the first part of special day group list
GetSpecialDayGroupListResponse
(NextStartReference =: *NextStartReference*,
SpecialDayGroup list =:
specialDayGroupCompleteList)
Send part of requested special day group list
Receive response
GetSpecialDayGroupList
(Limit skipped, StartReference :=
nextStartReference)
Retrieve the second part of special day group list
GetSpecialDayGroupListResponse
(NextStartReference =: *nextStartReference*,
SpecialDayGroup list =:
specialDayGroupListPart)
Send part of requested special day group list
Receive response


ONVIF Client
DUT
…
GetSpecialDayGroupList (Limit skipped, StartReference := *nextStartReference*)
Retrieve the last part of special day group list
GetSpecialDayGroupListResponse (NextStartReference =: *nextStartReference*, SpecialDayGroup list =: *specialDayGroupListPart*)
Send part of requested special day group list
Receive response
HelperGetSpecialDayGroupInfoList (out *specialDayGroupInfoCompleteList*) Annex A.4
Retrieve a complete list of special day groups
Return a complete list of special day groups

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

4. ONVIF client invokes **GetSpecialDayGroupList** with parameters

   - Limit skipped

   - StartReference skipped

5. The DUT responds with **GetSpecialDayGroupListResponse** message with parameters

   - NextStartReference =: *nextStartReference*

   - SpecialDayGroup list =: *specialDayGroupCompleteList*

6. If *specialDayGroupCompleteList* contains more SpecialDayGroup items than *cap*.MaxLimit, FAIL the test and skip other steps.

7. Until *nextStartReference* is not null, repeat the following steps:

   7.1. ONVIF client invokes **GetSpecialDayGroupList** with parameters

   - Limit skipped

   - StartReference := *nextStartReference*

7.2. The DUT responds with **GetSpecialDayGroupListResponse** message with parameters

- NextStartReference =: *nextStartReference*
- SpecialDayGroup list =: *specialDayGroupListPart*

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

7.4. Set the following:

- *specialDayGroupCompleteList* := *specialDayGroupCompleteList* + *specialDayGroupListPart*

8. If *specialDayGroupCompleteList* contains at least two SpecialDayGroup item with equal token, FAIL the test.

9. ONVIF Client retrieves a complete list of SpecialDayGroupInfo (out *specialDayGroupInfoCompleteList*) by following the procedure mentioned in Annex A.4.

10. If *specialDayGroupCompleteList* does not contain all tokens from *specialDayGroupInfoCompleteList*, FAIL the test and skip other steps.

11. If *specialDayGroupCompleteList* contains tokens other than tokens from *specialDayGroupInfoCompleteList*, FAIL the test and skip other steps.

12. For each SpecialDayGroupInfo.token *token* from *specialDayGroupInfoCompleteList* repeat the following steps:

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

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

Note: The following fields are compared at step 12.1:

- SpecialDayGroupInfo:
  - token
  - Name
  - Description

4.5.5 CREATE SPECIAL DAY GROUP

Test Label: Create Special Day Group Verification

Test Case ID: SCHEDULE-5-1-5

ONVIF Core Specification Coverage: SpecialDayGroup (ONVIF Schedule Service Specification), CreateSpecialDayGroup command (ONVIF Schedule Service Specification)

Command Under Test: CreateSpecialDayGroup

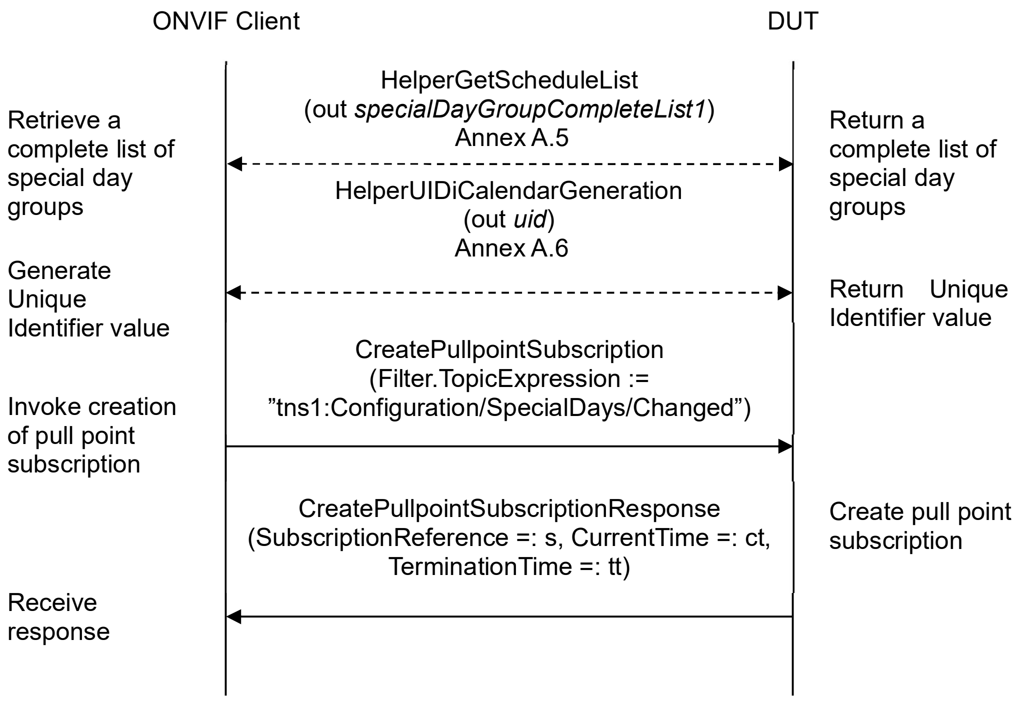
WSDL Reference: schedule.wsdl and event.wsdl

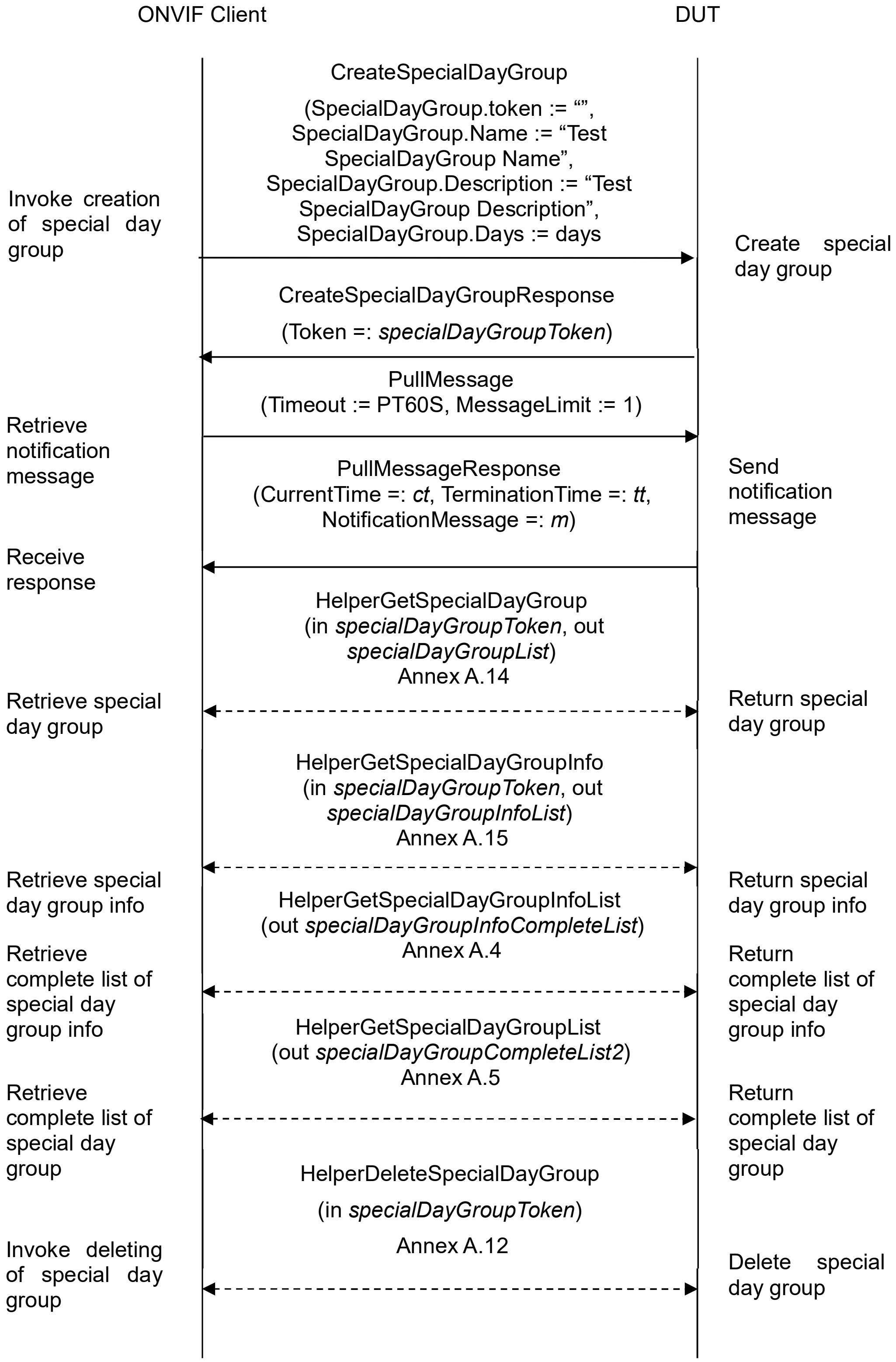
Test Purpose: To verify creation of special day group and generating of appropriate notifications.

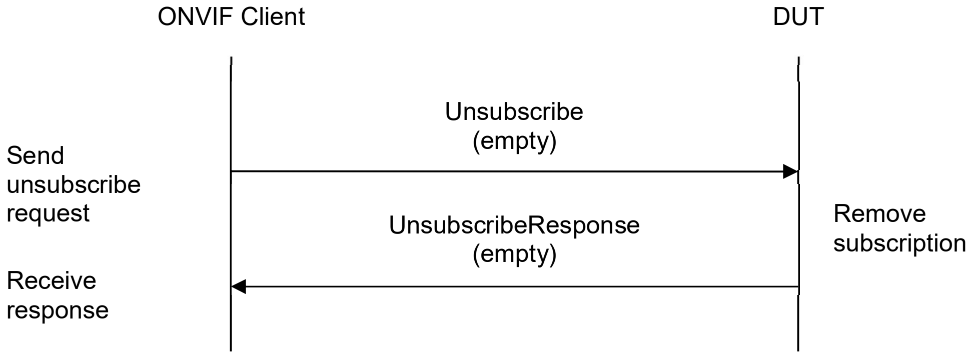
Pre-requisite: Schedule Service is received from the DUT. Event Service was received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported. The DUT shall have enough free storage capacity for one additional SpecialDayGroup.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetScheduleList
(out *specialDayGroupCompleteList1*)
Annex A.5
Retrieve a complete list of special day groups
Return a complete list of special day groups
HelperUIDiCalendarGeneration
(out *uid*)
Annex A.6
Generate Unique Identifier value
Return Unique Identifier value
CreatePullpointSubscription
(Filter.TopicExpression :=
"tns1:Configuration/SpecialDays/Changed")
Invoke creation of pull point subscription
CreatePullpointSubscriptionResponse
(SubscriptionReference =: s, CurrentTime =: ct,
TerminationTime =: tt)
Create pull point subscription
Receive response


ONVIF Client
DUT
CreateSpecialDayGroup
(SpecialDayGroup.token := “”, SpecialDayGroup.Name := “Test SpecialDayGroup Name”, SpecialDayGroup.Description := “Test SpecialDayGroup Description”, SpecialDayGroup.Days := days
Invoke creation of special day group
Create special day group
CreateSpecialDayGroupResponse
(Token =: *specialDayGroupToken*)
PullMessage
(Timeout := PT60S, MessageLimit := 1)
Retrieve notification message
PullMessageResponse
(CurrentTime =: *ct*, TerminationTime =: *tt*, NotificationMessage =: *m*)
Send notification message
Receive response
HelperGetSpecialDayGroup
(in *specialDayGroupToken*, out *specialDayGroupList*)
Annex A.14
Retrieve special day group
Return special day group
HelperGetSpecialDayGroupInfo
(in *specialDayGroupToken*, out *specialDayGroupInfoList*)
Annex A.15
Retrieve special day group info
Return special day group info
HelperGetSpecialDayGroupInfoList
(out *specialDayGroupInfoCompleteList*)
Annex A.4
Retrieve complete list of special day group info
Return complete list of special day group info
HelperGetSpecialDayGroupList
(out *specialDayGroupCompleteList2*)
Annex A.5
Retrieve complete list of special day group
Return complete list of special day group
HelperDeleteSpecialDayGroup
(in *specialDayGroupToken*)
Annex A.12
Invoke deleting of special day group
Delete special day group



Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves an initial complete list of SpecialDaysGroup (out *specialDayGroupCompleteList1*) by following the procedure mentioned in Annex A.5.

4. ONVIF Client generates Unique Identifier value for UID field in iCalendar (out *uid*) by following the procedure mentioned in Annex A.6.

5. Set the following:

    - *days* := “BEGIN:VCALENDAR

      BEGIN:VEVENT

      SUMMARY:Test special days

      DTSTART:<current year><current month><current day>T000000

      DTEND:<the date of the next day>T000000

      UID:*uid*

      END:VEVENT

      END:VCALENDAR”

6. ONVIF Client invokes **CreatePullPointSubscription** with parameters

    - Filter.TopicExpression := “tns1:Configuration/SpecialDays/Changed”

7. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters

    - SubscriptionReference =: *s*

    - CurrentTime =: *ct*

    - TerminationTime =: *tt*

8. ONVIF client invokes **CreateSpecialDayGroup** with parameters
    - SpecialDayGroup.token := “”
    - SpecialDayGroup.Name := “Test SpecialDayGroup Name”
    - SpecialDayGroup.Description := “Test SpecialDayGroup Description”
    - SpecialDayGroup.Days := *days*
9. The DUT responds with **CreateSpecialDayGroupResponse** message with parameters
    - Token =: *specialDayGroupToken*
10. Until *timeout1* timeout expires, repeat the following steps:

    10.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.

    10.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
    - Timeout := PT60S
    - MessageLimit := 1

    10.3. The DUT responds with **PullMessagesResponse** message with parameters
    - CurrentTime =: *ct*
    - TerminationTime =: *tt*
    - NotificationMessage =: *m*

    10.4. If *m* is not null and the TopicExpression item in *m* is not equal to “tns1:Configuration/SpecialDays/Changed”, FAIL the test and go to the step 21.

    10.5. If *m* is not null and does not contain Source.SimpleItem item with Name =: "SpecialDaysToken" and Value =: *specialDayGroupToken*, FAIL the test and go to the step 21.

    10.6. If *m* is not null and contains Source.SimpleItem item with Name =: "SpecialDaysToken" and Value =: *specialDayGroupToken*, go to the step 12.
11. If *timeout1* timeout expires for step 10 without Notification with SpecialDaysToken source simple item equal to *specialDayGroupToken*, FAIL the test and go to the step 21.
12. ONVIF Client retrieves a SpecialDayGroup (in *specialDayGroupToken*, out *specialDayGroupList*) by following the procedure mentioned in Annex A.14.
13. If *specialDayGroupList*[0] item does not have equal field values to values from step 8, FAIL the test and go step 21.
14. ONVIF Client retrieves a SpecialDayGroup info (in *specialDayGroupToken*, out *specialDayGroupInfoList*) by following the procedure mentioned in Annex A.15.
15. If *specialDayGroupInfoList*[0] item does not have equal field values to values from step 8, FAIL the test and go step 21.
16. ONVIF Client retrieves a complete SpecialDayGroup information list (out

specialDayGroupInfoCompleteList) by following the procedure mentioned in Annex A.4.

17. If *specialDayGroupInfoCompleteList* does not have *specialDayGroupInfo*[token = *specialDayGroupToken*] item with equal field values to values from step 8, FAIL the test and go step 21.

18. ONVIF Client retrieves a complete list of SpecialDayGroups (out *specialDayGroupCompleteList2*) by following the procedure mentioned in Annex A.5.

19. If *specialDayGroupCompleteList2* does not have SpecialDayGroup[token = *specialDayGroupToken*] item with equal field values to values from step 8, FAIL the test and go step 21.

20. For each SpecialDayGroup.token (*token*) from *specialDayGroupCompleteList1* do the following:

    20.1. If *specialDayGroupCompleteList2* does not have SpecialDayGroup[token = *token*] item, FAIL the test and go step 21.

21. ONVIF Client deletes the SpecialDayGroup (in *specialDayGroupToken*) by following the procedure mentioned in Annex A.12 to restore DUT configuration.

22. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

23. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

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

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

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

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

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

- SpecialDayGroup:
    - token
    - Name
    - Description
    - Days

Note: The following fields are compared at step 15, 17:

- SpecialDayGroupInfo:

- token
- Name
- Description

Note: For comparison SpecialDayGroup.Days field values the ONVIF Client compares all iCalendar fields each with other excluding UID field.

4.5.6 MODIFY SPECIAL DAY GROUP

Test Label: Modify Special Day Group Verification

Test Case ID: SCHEDULE-5-1-6

ONVIF Core Specification Coverage: SpecialDayGroup (ONVIF Schedule Service Specification), ModifySpecialDayGroup command (ONVIF Schedule Service Specification)

Command Under Test: ModifySpecialDayGroup

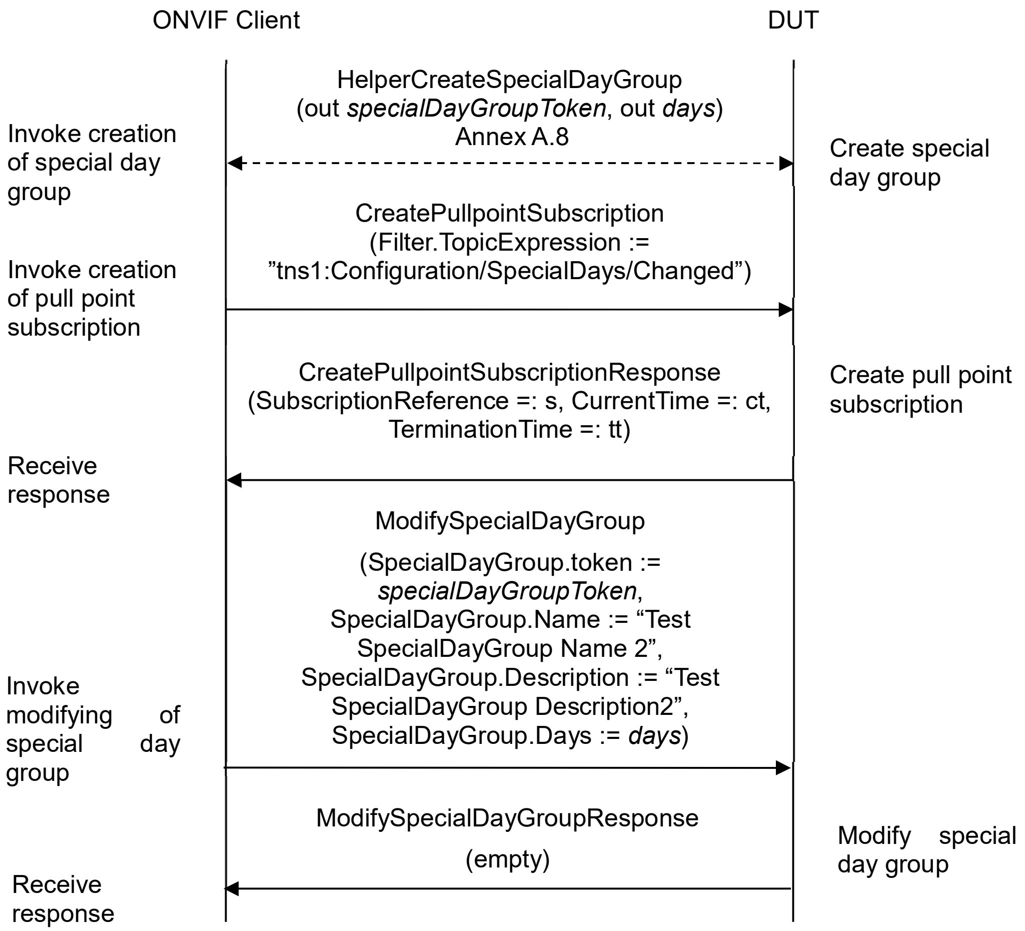
WSDL Reference: schedule.wsdl and event.wsdl

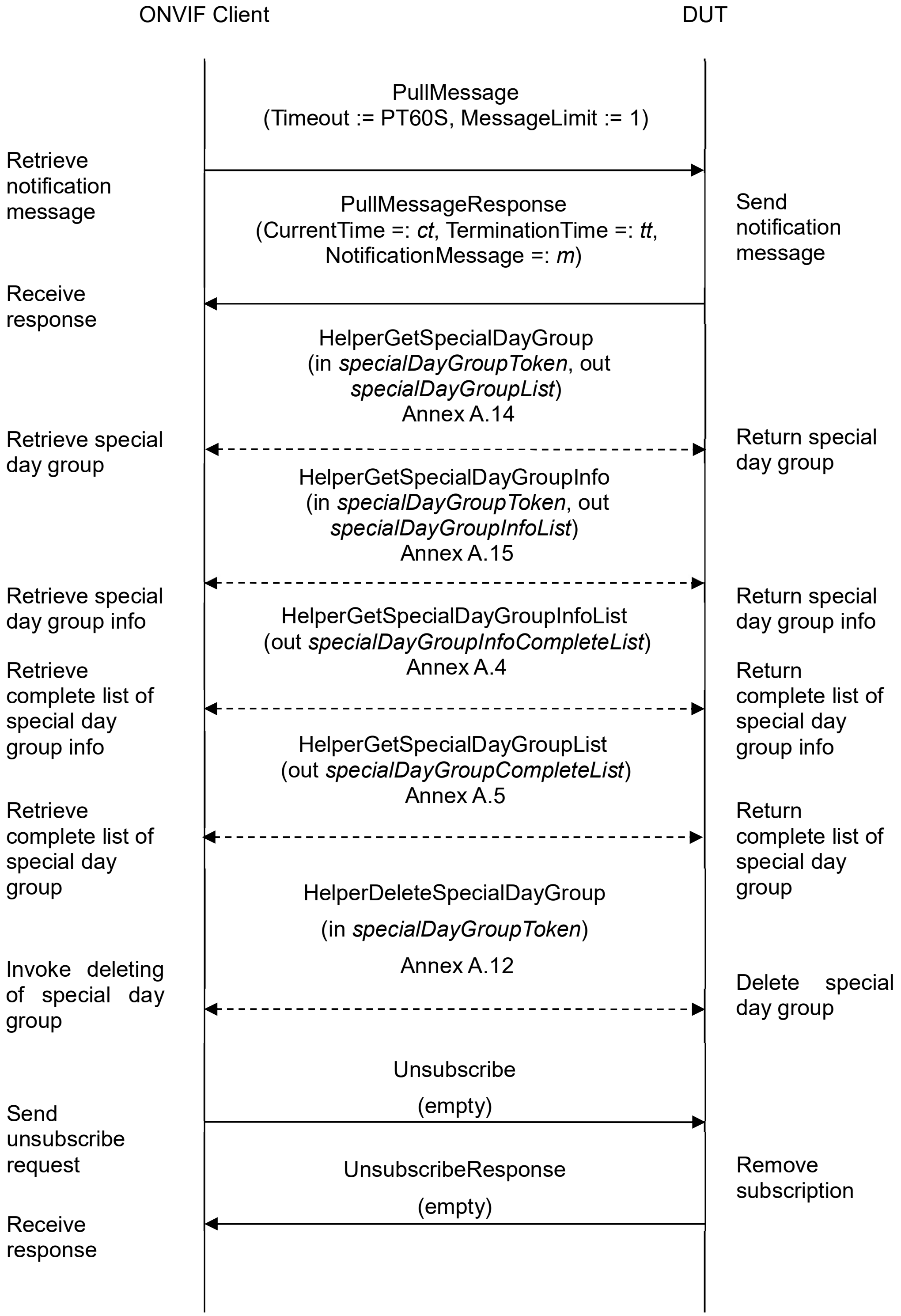
Test Purpose: To verify modifiing of special day group and generating of apropriate notifications.

Pre-requisite: Schedule Service is received from the DUT. Event Service was received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported. The DUT shall have enough free storage capacity for one additional SpecialDayGroup.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperCreateSpecialDayGroup
(out *specialDayGroupToken*, out *days*)
Annex A.8
Invoke creation of special day group
Create special day group
CreatePullpointSubscription
(Filter.TopicExpression :=
"tns1:Configuration/SpecialDays/Changed")
Invoke creation of pull point subscription
CreatePullpointSubscriptionResponse
(SubscriptionReference =: s, CurrentTime =: ct,
TerminationTime =: tt)
Create pull point subscription
Receive response
ModifySpecialDayGroup
(SpecialDayGroup.token :=
specialDayGroupToken,
SpecialDayGroup.Name := “Test
SpecialDayGroup Name 2”,
SpecialDayGroup.Description := “Test
SpecialDayGroup Description2”,
SpecialDayGroup.Days := *days*)
Invoke modifying of special day group
ModifySpecialDayGroupResponse
(empty)
Modify special day group
Receive response


ONVIF Client
DUT
PullMessage
(Timeout := PT60S, MessageLimit := 1)
Retrieve notification message
PullMessageResponse
(CurrentTime =: *ct*, TerminationTime =: *tt*, NotificationMessage =: *m*)
Send notification message
Receive response
HelperGetSpecialDayGroup
(in *specialDayGroupToken*, out *specialDayGroupList*)
Annex A.14
Retrieve special day group
Return special day group
HelperGetSpecialDayGroupInfo
(in *specialDayGroupToken*, out *specialDayGroupInfoList*)
Annex A.15
Retrieve special day group info
Return special day group info
HelperGetSpecialDayGroupInfoList
(out *specialDayGroupInfoCompleteList*)
Annex A.4
Retrieve complete list of special day group info
Return complete list of special day group info
HelperGetSpecialDayGroupList
(out *specialDayGroupCompleteList*)
Annex A.5
Retrieve complete list of special day group
Return complete list of special day group
HelperDeleteSpecialDayGroup
(in *specialDayGroupToken*)
Annex A.12
Invoke deleting of special day group
Delete special day group
Unsubscribe
(empty)
Send unsubscribe request
UnsubscribeResponse
(empty)
Remove subscription
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client creates SpecialDayGroup (out *specialDayGroupToken*) with iCalendar value of Days field (out *days*) by following the procedure mentioned in Annex A.8.

4. Set the following:

   days := *days* with changed <day> value in DTSTART field (increase day value in one day) and with changed <day> value in DTEND field (increase day value in one day)

5. ONVIF Client invokes **CreatePullPointSubscription** with parameters

   - Filter.TopicExpression := "tns1:Configuration/SpecialDays/Changed"

6. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters

   - SubscriptionReference =: *s*
   - CurrentTime =: *ct*
   - TerminationTime =: *tt*

7. ONVIF client invokes **ModifySpecialDayGroup** with parameters

   - SpecialDayGroup.token := *specialDayGroupToken*
   - SpecialDayGroup.Name := "Test SpecialDayGroup Name 2"
   - SpecialDayGroup.Description := "Test SpecialDayGroup Description2"
   - SpecialDayGroup.Days := *days*

8. The DUT responds with empty **ModifySpecialDayGroupResponse** message.

9. Until *timeout1* timeout expires, repeat the following steps:

   9.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.

   9.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters

   - Timeout := PT60S
   - MessageLimit := 1

   9.3. The DUT responds with **PullMessagesResponse** message with parameters

   - CurrentTime =: *ct*
   - TerminationTime =: *tt*
   - NotificationMessage =: *m*

   9.4. If *m* is not null and the TopicExpression item in *m* is not equal to "tns1:Configuration/SpecialDays/Changed", FAIL the test and go to the step 19.

9.5. If *m* is not null and does not contain Source.SimpleItem item with Name =: "SpecialDaysToken" and Value =: *specialDayGroupToken*, FAIL the test and go to the step 19.

9.6. If *m* is not null and contains Source.SimpleItem item with Name =: "SpecialDaysToken" and Value =: *specialDayGroupToken*, go to the step 11.

10. If *timeout1* timeout expires for step 9 without Notification with SpecialDaysToken source simple item equal to *specialDayGroupToken*, FAIL the test and go to the step 19.

11. ONVIF Client retrieves a SpecialDayGroup (in *specialDayGroupToken*, out *specialDayGroupList*) by following the procedure mentioned in Annex A.14.

12. If *specialDayGroupList*[0] item does not have equal field values to values from step 7, FAIL the test and go step 19.

13. ONVIF Client retrieves a SpecialDayGroup info (in *specialDayGroupToken*, out *specialDayGroupInfoList*) by following the procedure mentioned in Annex A.15.

14. If *specialDayGroupInfoList*[0] item does not have equal field values to values from step 7, FAIL the test and go step 19.

15. ONVIF Client retrieves a complete SpecialDayGroup information list (out *specialDayGroupInfoCompleteList*) by following the procedure mentioned in Annex A.4.

16. If *specialDayGroupInfoCompleteList* does not have *specialDayGroupInfo*.[token = *specialDayGroupToken*] item with equal field values to values from step 7, FAIL the test and go step 19.

17. ONVIF Client retrieves a complete list of SpecialDayGroups (out *specialDayGroupCompleteList*) by following the procedure mentioned in Annex A.5.

18. If *specialDayGroupCompleteList* does not have *specialDayGroup*.[token = *specialDayGroupToken*] item with equal field values to values from step 7, FAIL the test and go step 19.

19. ONVIF Client deletes the SpecialDayGroup (in *specialDayGroupToken*) by following the procedure mentioned in Annex A.12 to restore DUT configuration.

20. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

21. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

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

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

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

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

Note: The following fields are compared at steps 12, 18:

- SpecialDayGroup:
    - token
    - Name
    - Description
    - Days

Note: The following fields are compared at step 14, 16:

- SpecialDayGroup:
    - token
    - Name
    - Description

Note: For comparison SpecialDayGroup.Days field values the ONVIF Client compares all iCalendar fields each with other excluding UID field.

4.5.7 DELETE SPECIAL DAY GROUP

Test Label: Delete Special Day Group Verification

Test Case ID: SCHEDULE-5-1-7

ONVIF Core Specification Coverage: SpecialDayGroup (ONVIF Schedule Service Specification), DeleteSpecialDayGroup command (ONVIF Schedule Service Specification)

Command Under Test: DeleteSpecialDayGroup

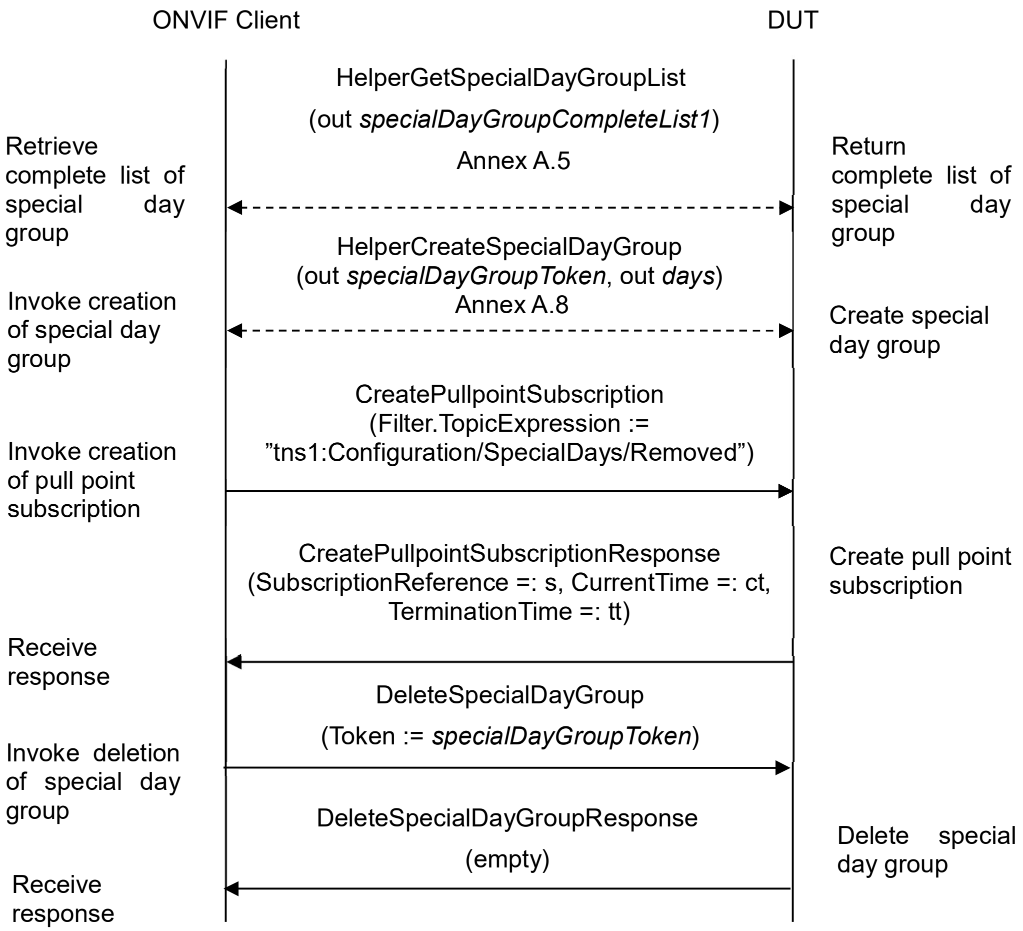
WSDL Reference: schedule.wsdl and event.wsdl

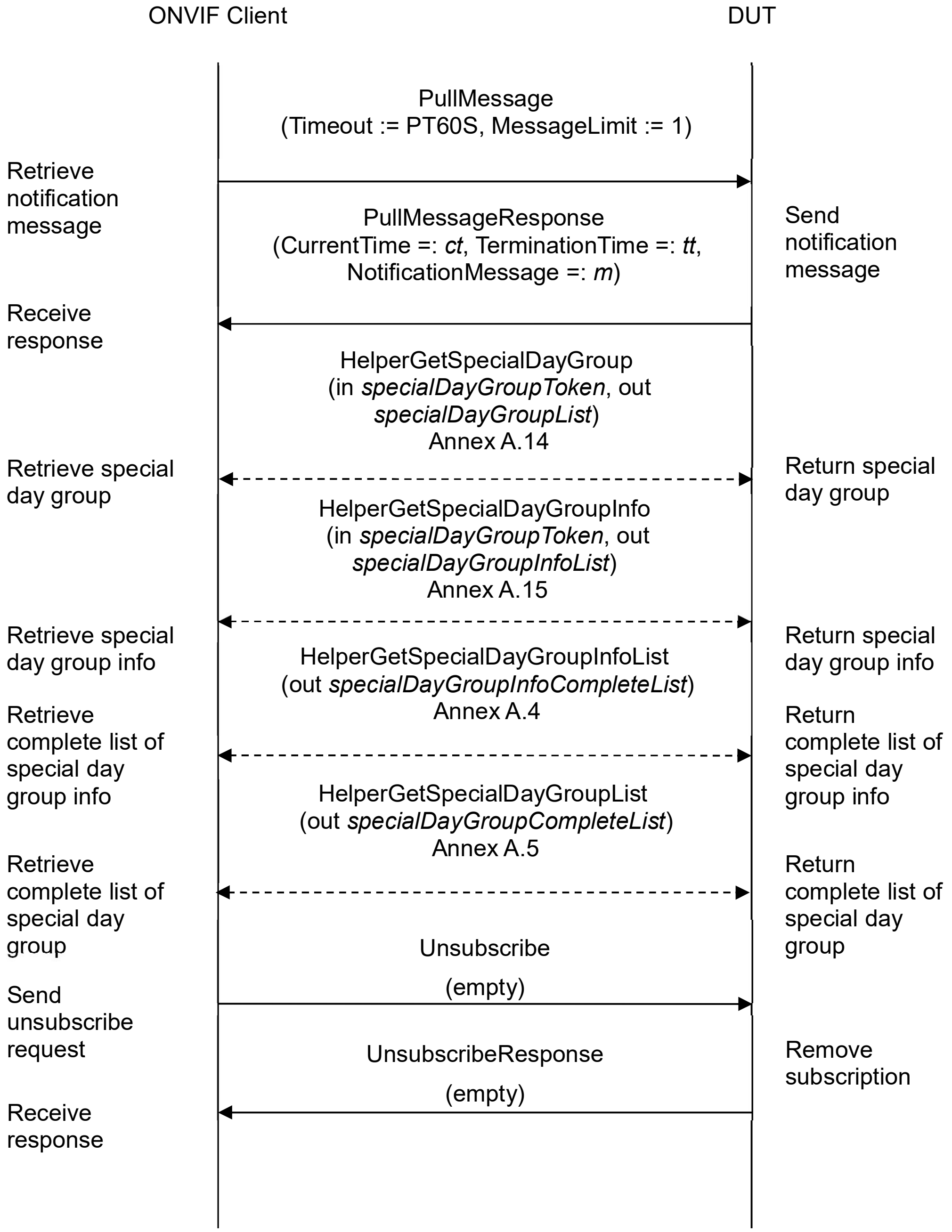
Test Purpose: To verify deleting of special day group and generating of apropriate notifications.

Pre-requisite: Schedule Service is received from the DUT. Event Service was received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported. The DUT shall have enough free storage capacity for one additional SpecialDayGroup.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetSpecialDayGroupList
(out *specialDayGroupCompleteList1*)
Annex A.5
Retrieve complete list of special day group
Return complete list of special day group
HelperCreateSpecialDayGroup
(out *specialDayGroupToken*, out *days*)
Annex A.8
Invoke creation of special day group
Create special day group
CreatePullpointSubscription
(Filter.TopicExpression :=
"tns1:Configuration/SpecialDays/Removed")
Invoke creation of pull point subscription
CreatePullpointSubscriptionResponse
(SubscriptionReference =: s, CurrentTime =: ct,
TerminationTime =: tt)
Create pull point subscription
Receive response
DeleteSpecialDayGroup
(Token := *specialDayGroupToken*)
Invoke deletion of special day group
DeleteSpecialDayGroupResponse
(empty)
Delete special day group
Receive response


ONVIF Client
DUT
PullMessage
(Timeout := PT60S, MessageLimit := 1)
Retrieve notification message
PullMessageResponse
(CurrentTime =: *ct*, TerminationTime =: *tt*, NotificationMessage =: *m*)
Send notification message
Receive response
HelperGetSpecialDayGroup
(in *specialDayGroupToken*, out *specialDayGroupList*)
Annex A.14
Retrieve special day group
Return special day group
HelperGetSpecialDayGroupInfo
(in *specialDayGroupToken*, out *specialDayGroupInfoList*)
Annex A.15
Retrieve special day group info
Return special day group info
HelperGetSpecialDayGroupInfoList
(out *specialDayGroupInfoCompleteList*)
Annex A.4
Retrieve complete list of special day group info
Return complete list of special day group info
HelperGetSpecialDayGroupList
(out *specialDayGroupCompleteList*)
Annex A.5
Retrieve complete list of special day group
Return complete list of special day group
Unsubscribe
(empty)
Send unsubscribe request
UnsubscribeResponse
(empty)
Remove subscription
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves an initial complete list of SpecialDaysGroup (out

specialDayGroupsCompleteList1) by following the procedure mentioned in Annex A.5.

4. ONVIF Client creats SpecialDayGroup (out *specialDayGroupToken*) by following the procedure mentioned in Annex A.8.

5. ONVIF Client invokes **CreatePullPointSubscription** with parameters

    - Filter.TopicExpression := “tns1:Configuration/SpecialDays/Removed”

6. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters

    - SubscriptionReference =: *s*
    - CurrentTime =: *ct*
    - TerminationTime =: *tt*

7. ONVIF Client invokes **DeleteSpecialDayGroup** with parameters

    - Token := *specialDayGroupToken*

8. The DUT responds with empty **DeleteSpecialDayGroupResponse** message.

9. Until *timeout1* timeout expires, repeat the following steps:

    9.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.

    9.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters

    - Timeout := PT60S
    - MessageLimit := 1

    9.3. The DUT responds with **PullMessagesResponse** message with parameters

    - CurrentTime =: *ct*
    - TerminationTime =: *tt*
    - NotificationMessage =: *m*

    9.4. If *m* is not null and the TopicExpression item in *m* is not equal to “tns1:Configuration/SpecialDays/Removed”, FAIL the test and go to the step 20.

    9.5. If *m* is not null and does not contain Source.SimpleItem item with Name =: "SpecialDaysToken" and Value =: *specialDayGroupToken*, FAIL the test and go to the step 20.

    9.6. If *m* is not null and contains Source.SimpleItem item with Name =: "SpecialDaysToken" and Value =: *specialDayGroupToken*, go to the step 11.

10. If *timeout1* timeout expires for step 9 without Notification with SpecialDaysToken source simple item equal to *specialDayGroupToken*, FAIL the test and go to the step 20.

11. ONVIF Client retrieves a SpecialDayGroup (in *specialDayGroupToken*, out *specialDayGroupList*) by following the procedure mentioned in Annex A.14.

12. If *specialDayGroupList* is not empty, FAIL the test and go step 20.

13. ONVIF Client retrieves a SpecialDayGroup info (in *specialDayGroupToken*, out *specialDayGroupInfoList*) by following the procedure mentioned in Annex A.15.

14. If *specialDayGroupInfoList* is not empty, FAIL the test and go step 20.

15. ONVIF Client retrieves a complete SpecialDayGroup information list (out *specialDayGroupInfoCompleteList*) by following the procedure mentioned in Annex A.4.

16. If *specialDayGroupInfoCompleteList* contains *specialDayGroupInfo*.[token = *specialDayGroupToken*] item, FAIL the test and go step 20.

17. ONVIF Client retrieves a complete list of SpecialDayGroups (out *specialDayGroupCompleteList2*) by following the procedure mentioned in Annex A.5.

18. If *specialDayGroupCompleteList2* contains SpecialDayGroup[token = *specialDayGroupToken*] item, FAIL the test and go step 20.

19. For each SpecialDayGroup.token (*token*) from *specialDayGroupCompleteList1* do the following:

    19.1. If *specialDayGroupCompleteList2* does not have SpecialDayGroup[token = *token*] item, FAIL the test and go step 20.

20. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

21. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

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

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

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

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

4.5.8 GET SPECIAL DAY GROUPS WITH INVALID TOKEN

Test Label: Get Special Day Groups with Invalid Token Verification

Test Case ID: SCHEDULE-5-1-8

ONVIF Core Specification Coverage: SpecialDayGroup (ONVIF Schedule Service Specification), GetSpecialDayGroups command (ONVIF Schedule Service Specification)

Command Under Test: GetSpecialDayGroups

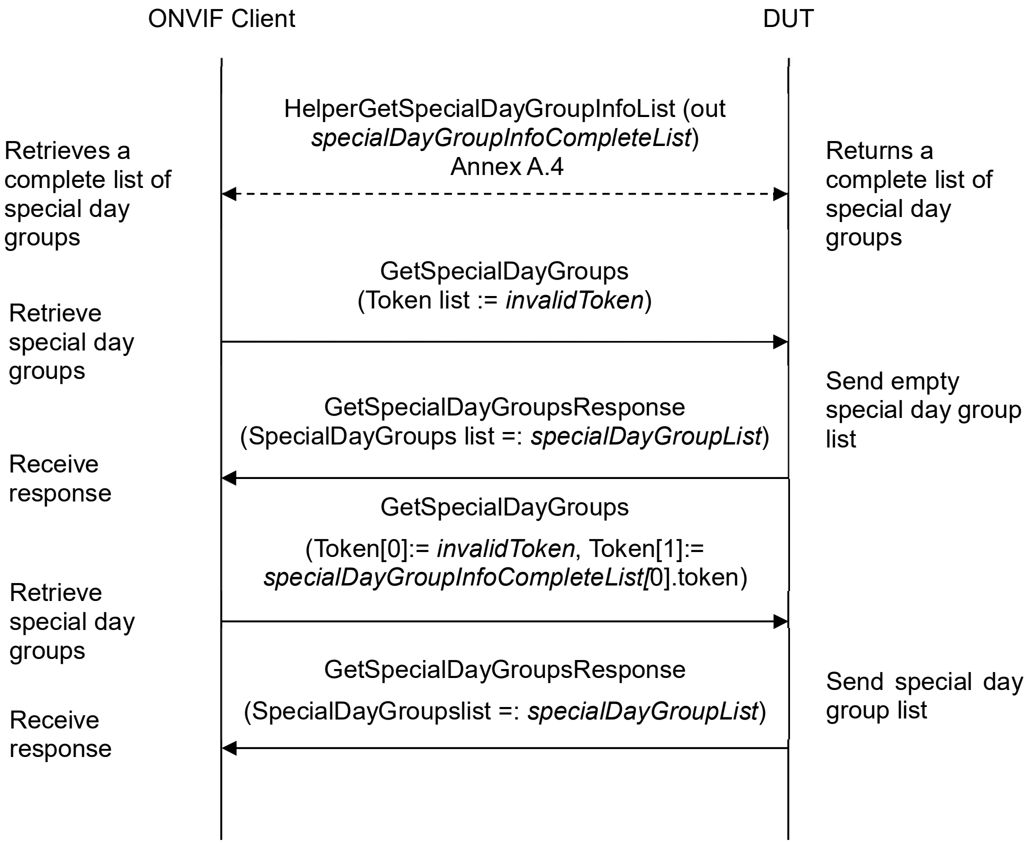
WSDL Reference: schedule.wsdl

Test Purpose: To verify Get Special Day Groups with invalid token.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Retrieves a complete list of special day groups
HelperGetSpecialDayGroupInfoList (out *specialDayGroupInfoCompleteList*) Annex A.4
Returns a complete list of special day groups
Retrieve special day groups
GetSpecialDayGroups (Token list := *invalidToken*)
Send empty special day group list
GetSpecialDayGroupsResponse (SpecialDayGroups list =: *specialDayGroupList*)
Receive response
GetSpecialDayGroups
(Token[0]:= *invalidToken*, Token[1]:= *specialDayGroupInfoCompleteList*[0].token)
Retrieve special day groups
GetSpecialDayGroupsResponse
(SpecialDayGroupslist =: *specialDayGroupList*)
Send special day group list
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client retrieves a complete list of special day group information (out *specialDayGroupInfoCompleteList*) by following the procedure mentioned in Annex A.4.
4. Set the following:
    - *invalidToken* := value not equal to any *specialDayGroupInfoCompleteList*.token
5. ONVIF client invokes **GetSpecialDayGroups** with parameters
    - Token list := *invalidToken*
6. The DUT responds with **GetSpecialDayGroupsResponse** message with parameters
    - SpecialDayGroup list =: *specialDayGroupList*
7. If *specialDayGroupList* is not empty, FAIL the test.
8. If *specialDayGroupInfoCompleteList* is empty, skip other steps.
9. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.
10. If *cap*.MaxLimit is less than 2, skip other steps.
11. ONVIF client invokes **GetSpecialDayGroups** with parameters
    - Token[0]:= *invalidToken*
    - Token[1]:= *specialDayGroupInfoCompleteList*[0].token
12. The DUT responds with **GetSpecialDayGroupsResponse** message with parameters
    - ScheduleInfo list =: *specialDayGroupList*
13. If *specialDayGroupList* is empty, FAIL the test.
14. If *specialDayGroupList* contains more than one item, FAIL the test.
15. If *specialDayGroupList*[0].token does not equal to *specialDayGroupInfoCompleteList*[0].token, FAIL the test.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

4.5.9 GET SPECIAL DAY GROUPS - TOO MANY ITEMS

Test Label: Get Special Day Groups - number of requested items is greater than MaxLimit

Test Case ID: SCHEDULE-5-1-9

ONVIF Core Specification Coverage: SpecialDayGroups (ONVIF Schedule Service Specification), GetSpecialDayGroups command (ONVIF Schedule Service Specification)

Command Under Test: GetSpecialDayGroups

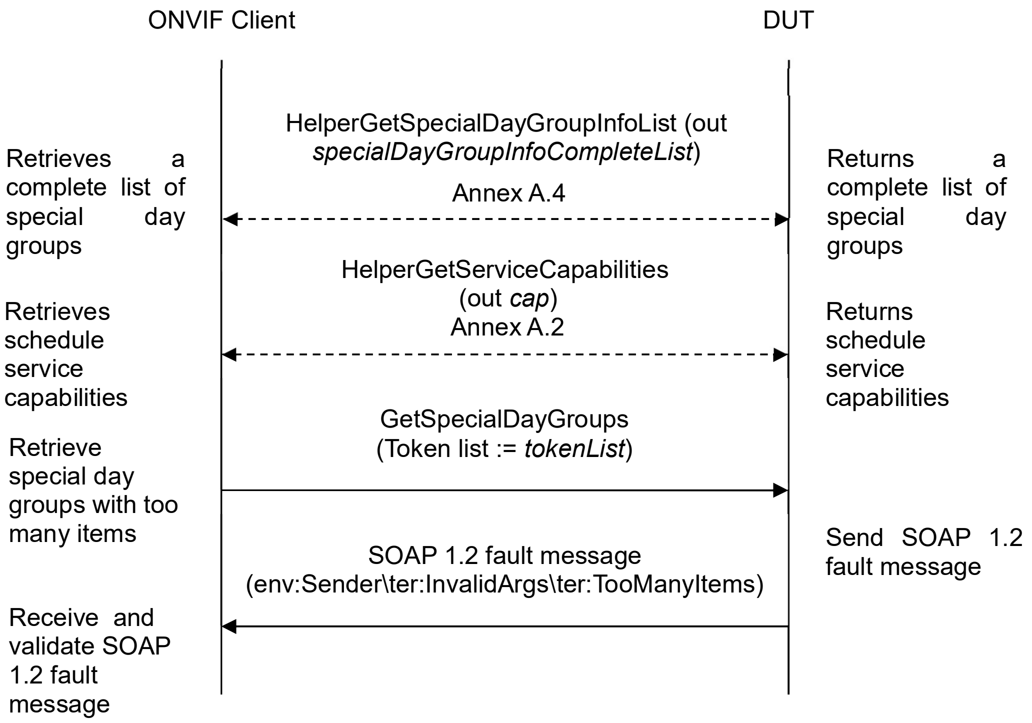
WSDL Reference: schedule.wsdl

Test Purpose: To verify Get Special Day Groups in case there are more items than MaxLimit in request.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetSpecialDayGroupInfoList (out *specialDayGroupInfoCompleteList*)
Annex A.4
Retrieves a complete list of special day groups
Returns a complete list of special day groups
HelperGetServiceCapabilities (out *cap*)
Annex A.2
Retrieves schedule service capabilities
Returns schedule service capabilities
GetSpecialDayGroups (Token list := *tokenList*)
Retrieve special day groups with too many items
SOAP 1.2 fault message (env:Sender\ter:InvalidArgs\ter:TooManyItems)
Send SOAP 1.2 fault message
Receive and validate SOAP 1.2 fault message

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of special day group information (out *specialDayGroupInfoCompleteList*) by following the procedure mentioned in Annex A.4.

4. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

5. If *specialDayGroupInfoCompleteList.*token items number is less than *cap*.MaxLimit or equal to *cap*.MaxLimit, skip other steps.

6. Set the following:

    - *tokenList* := [subset of *specialDayGroupInfoCompleteList*.token values with items number equal to *cap*.MaxLimit + 1]

7. ONVIF client invokes **GetSpecialDayGroups** with parameters

    - Token list := *tokenList*

8. The DUT returns **env:Sender\ter:InvalidArgs\ter:TooManyItems** SOAP 1.2 fault.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send env:Sender\ter:InvalidArgs\ter:TooManyItems SOAP 1.2 fault.

4.5.10 CREATE SPECIAL DAY GROUP - NOT EMPTY SPECIAL DAY GROUP TOKEN

Test Label: Create Special Day Group with not Empty Token Verification

Test Case ID: SCHEDULE-5-1-10

ONVIF Core Specification Coverage: SpecialDayGroup (ONVIF Schedule Service Specification), CreateSpecialDayGroup command (ONVIF Schedule Service Specification)

Command Under Test: CreateSpecialDayGroup

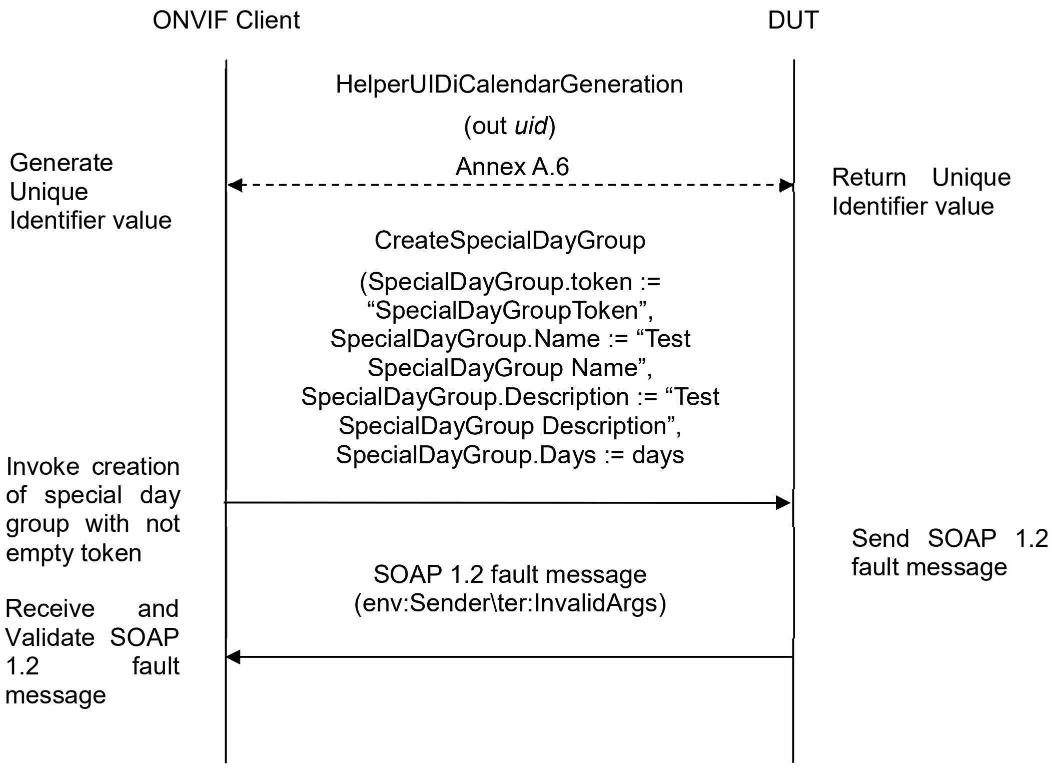
WSDL Reference: schedule.wsdl

Test Purpose: To verify creation of special day group with not empty token.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported. The DUT shall have enough free storage capacity for one additional SpecialDayGroup.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperUIDiCalendarGeneration
(out *uid*)
Annex A.6
Generate Unique Identifier value
Return Unique Identifier value
CreateSpecialDayGroup
(SpecialDayGroup.token := “SpecialDayGroupToken”, SpecialDayGroup.Name := “Test SpecialDayGroup Name”, SpecialDayGroup.Description := “Test SpecialDayGroup Description”, SpecialDayGroup.Days := days
Invoke creation of special day group with not empty token
Send SOAP 1.2 fault message
SOAP 1.2 fault message (env:Sender\ter:InvalidArgs)
Receive and Validate SOAP 1.2 fault message

Test Procedure:

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

3. ONVIF Client generates Unique Identifier value for UID field in iCalendar (out *uid*) by following the procedure mentioned in Annex A.6.
4. Set the following:
    - *days* := "BEGIN:VCALENDAR

      BEGIN:VEVENT

      SUMMARY:Test special days

      DTSTART:<current year><current month><current day>T000000

      DTEND:<the date of the next day>T000000

      UID:*uid*

      END:VEVENT

      END:VCALENDAR"
5. ONVIF client invokes **CreateSpecialDayGroup** with parameters
    - SpecialDayGroup.token := "SpecialDayGroupToken"
    - SpecialDayGroup.Name := "Test SpecialDayGroup Name"
    - SpecialDayGroup.Description := "Test SpecialDayGroup Description"
    - SpecialDayGroup.Days := *days*
6. The DUT returns **env:**Sender**\ter:InvalidArgs** SOAP 1.2 fault.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send env:Sender\ter:InvalidArgs SOAP 1.2 fault.

4.5.11 MODIFY SPECIAL DAY GROUP WITH INVALID TOKEN

Test Label: Modify Special Day Group with Invalid Token Verification

Test Case ID: SCHEDULE-5-1-11

ONVIF Core Specification Coverage: ModifySpecialDayGroup command (ONVIF Schedule Service Specification)

Command Under Test: ModifySpecialDayGroup

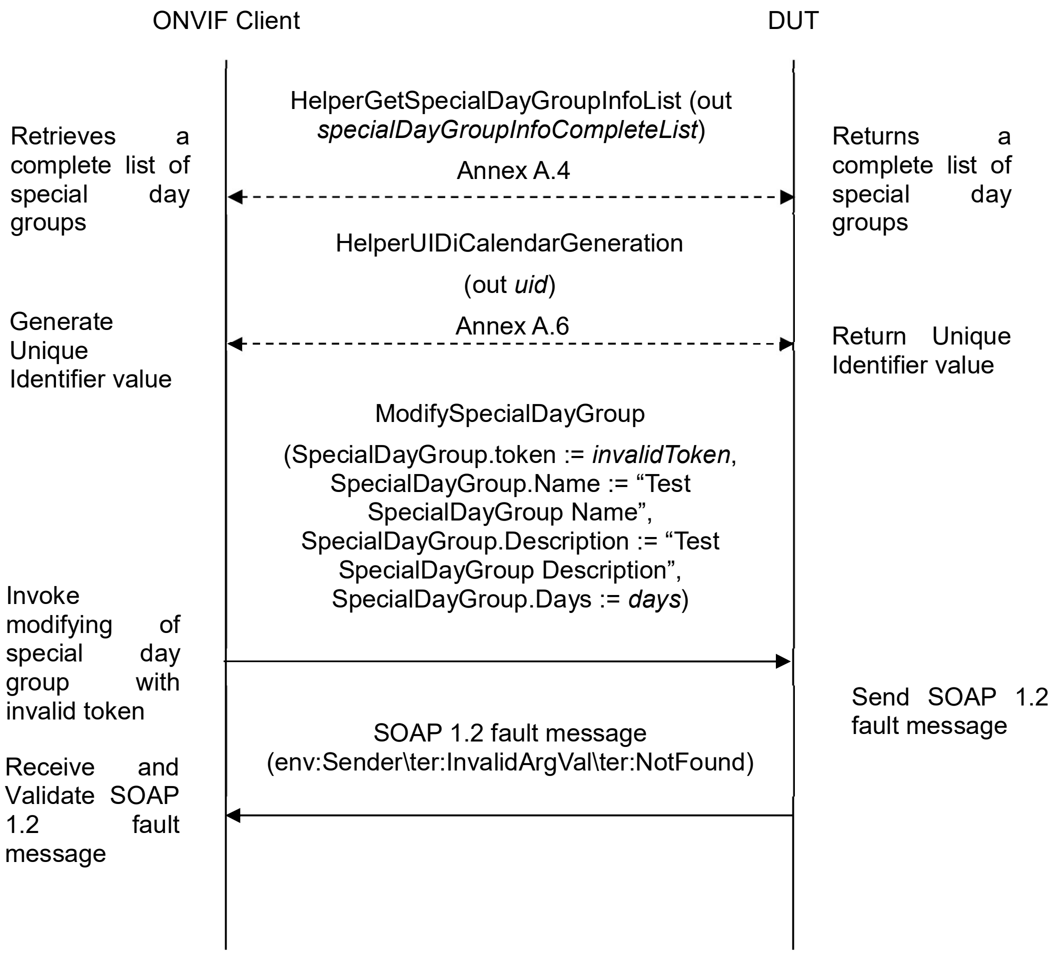
WSDL Reference: schedule.wsdl

Test Purpose: To verify modifiing of special day group with invalid token.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Retrieves a complete list of special day groups
HelperGetSpecialDayGroupInfoList (out *specialDayGroupInfoCompleteList*)
Annex A.4
Returns a complete list of special day groups
Generate Unique Identifier value
HelperUIDiCalendarGeneration
(out *uid*)
Annex A.6
Return Unique Identifier value
ModifySpecialDayGroup
(SpecialDayGroup.token := *invalidToken*, SpecialDayGroup.Name := “Test SpecialDayGroup Name”, SpecialDayGroup.Description := “Test SpecialDayGroup Description”, SpecialDayGroup.Days := *days*)
Invoke modifying of special day group with invalid token
Send SOAP 1.2 fault message
SOAP 1.2 fault message (env:Sender\ter:InvalidArgVal\ter:NotFound)
Receive and Validate SOAP 1.2 fault message

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of special day group information (out *specialDayGroupInfoCompleteList*) by following the procedure mentioned in Annex A.4.

4. ONVIF Client generates Unique Identifier value for UID field in iCalendar (out *uid*) by following the procedure mentioned in Annex A.6.

5. Set the following:

   - *days* := "BEGIN:VCALENDAR

     BEGIN:VEVENT

     SUMMARY:Test special days

     DTSTART:<current year><current month><current day>T000000

     DTEND:<the date of the next day>T000000

     UID:*uid*

     END:VEVENT

     END:VCALENDAR"

6. Set the following:

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

7. ONVIF client invokes **ModifySpecialDayGroup** with parameters

   - SpecialDayGroup.token := *invalidToken*

   - SpecialDayGroup.Name := "Test SpecialDayGroup Name"

   - SpecialDayGroup.Description := "Test SpecialDayGroup Description"

   - SpecialDayGroup.Days := *days*

8. The DUT returns **env:Sender\ter:InvalidArgVal\ter:NotFound** SOAP 1.2 fault.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

Note: If the DUT sends other SOAP 1.2 fault message than specified, log WARNING message, and PASS the test.

4.5.12 DELETE SPECIAL DAY GROUP WITH INVALID TOKEN

Test Label: Delete Special Day Group with Invalid Token Verification

Test Case ID: SCHEDULE-5-1-12

ONVIF Core Specification Coverage: DeleteSpecialDayGroup command (ONVIF Schedule Service Specification)

Command Under Test: DeleteSpecialDayGroup

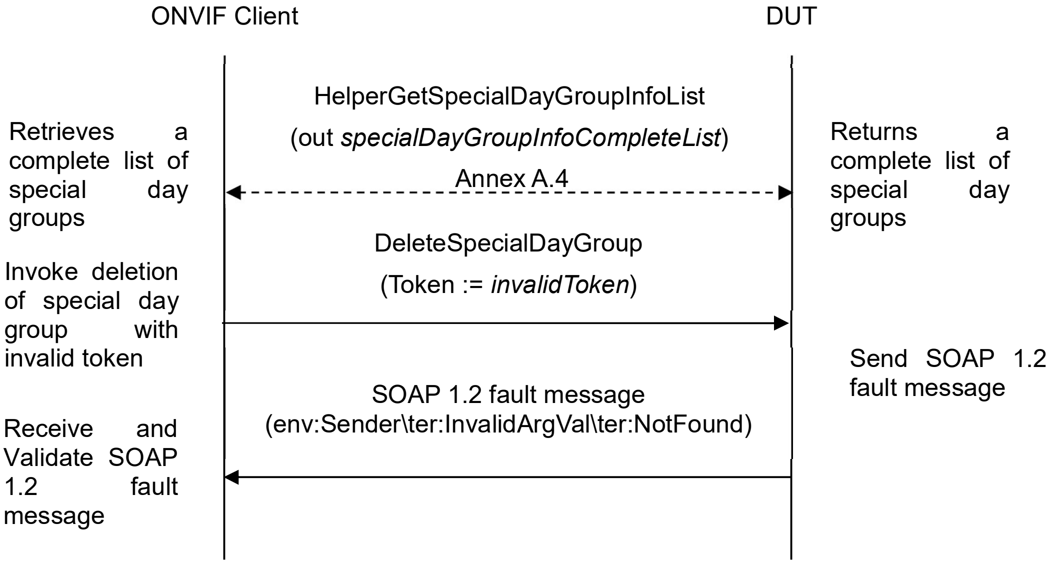
WSDL Reference: schedule.wsdl

Test Purpose: To verify deleting of special day group with invalid token.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetSpecialDayGroupInfoList
(out *specialDayGroupInfoCompleteList*)
Annex A.4
Retrieves a complete list of special day groups
Returns a complete list of special day groups
DeleteSpecialDayGroup
(Token := *invalidToken*)
Invoke deletion of special day group with invalid token
Send SOAP 1.2 fault message
SOAP 1.2 fault message
(env:Sender\ter:InvalidArgVal\ter:NotFound)
Receive and Validate SOAP 1.2 fault message

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of special day group information (out *specialDayGroupInfoCompleteList*) by following the procedure mentioned in Annex A.4.

4. Set the following:

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

5. ONVIF Client invokes **DeleteSpecialDayGroup** with parameters
    - Token := *invalidToken*
6. The DUT returns **env:Sender\ter:InvalidArgVal\ter:NotFound** SOAP 1.2 fault.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

Note: If the DUT sends other SOAP 1.2 fault message than specified, log WARNING message, and PASS the test.

4.6 *Schedule State*

4.6.1 GET SCHEDULE STATE

Test Label: Get Schedule State Verification

Test Case ID: SCHEDULE-6-1-1

ONVIF Core Specification Coverage: ScheduleState (ONVIF Schedule Service Specification), Get ScheduleState command (ONVIF Schedule Service Specification)

Command Under Test: GetScheduleState

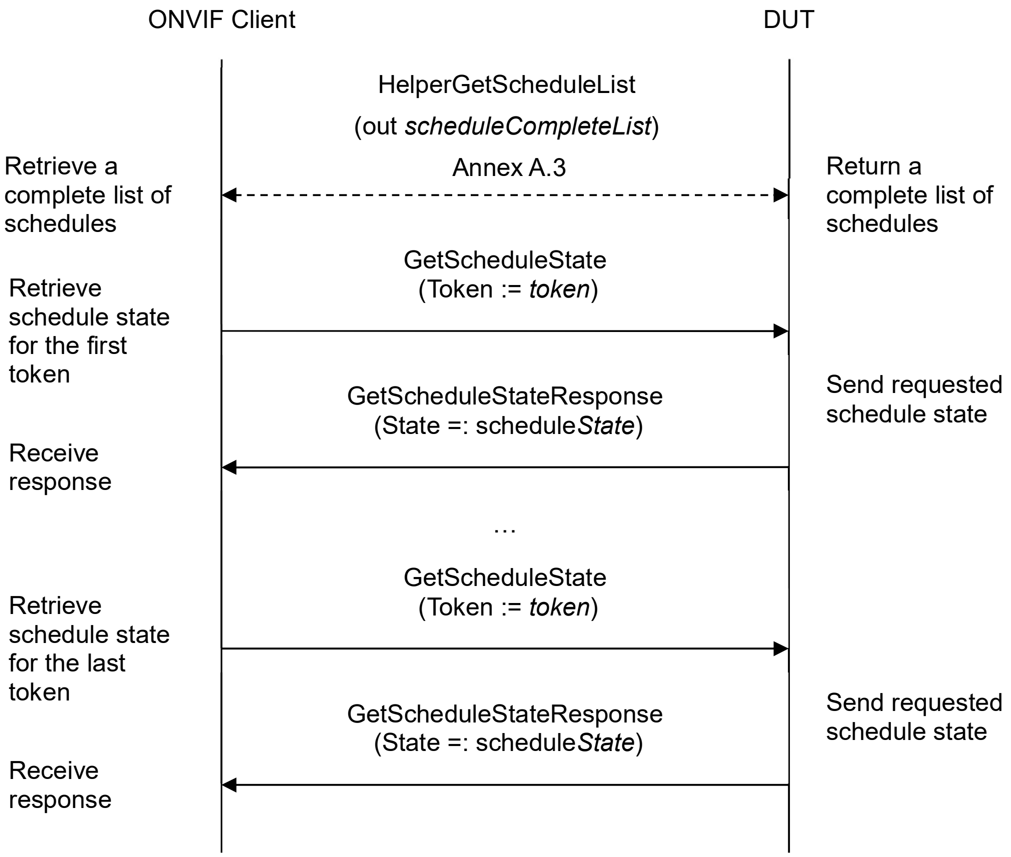
WSDL Reference: schedule.wsdl

Test Purpose: To verify Get Schedule State.

Pre-requisite: Schedule Service is received from the DUT. State Reporting is supported by the DUT as indicated by the Capabilities.StateReportingSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetScheduleList
(out *scheduleCompleteList*)
Annex A.3
Retrieve a complete list of schedules
Return a complete list of schedules
GetScheduleState
(Token := *token*)
Retrieve schedule state for the first token
GetScheduleStateResponse
(State =: schedule*State*)
Send requested schedule state
Receive response
…
GetScheduleState
(Token := *token*)
Retrieve schedule state for the last token
GetScheduleStateResponse
(State =: schedule*State*)
Send requested schedule state
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete Schedule list (out *scheduleCompleteList*) by following the procedure mentioned in Annex A.3.

4. If *scheduleCompleteList* is empty, skip other steps.

5. For each Schedule.token *scheduleToken* from *scheduleCompleteList* repeat the following steps:

   5.1. ONVIF client invokes **GetScheduleState** with parameters

   - Token := *scheduleToken*

   5.2. The DUT responds with **GetScheduleStateResponse** message with parameters

   - ScheduleState =: *scheduleState*

   5.3. If Schedule[token = *scheduleToken*] item from *scheduleCompleteList* contains at least one SpecialDays item and *scheduleState* does not contain SpecialDays item, FAIL the test.

   5.4. If Schedule[token = *scheduleToken*] item from *scheduleCompleteList* does not contain SpecialDays item and *scheduleState*.SpecialDays is equal to true, FAIL the test.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

4.6.2 CHANGE SCHEDULE STATE - CHANGE STANDARD

Test Label: Get Schedule State Verification

Test Case ID: SCHEDULE-6-1-2

ONVIF Core Specification Coverage: ScheduleState (ONVIF Schedule Service Specification), Get ScheduleState command (ONVIF Schedule Service Specification)

Command Under Test: GetScheduleState

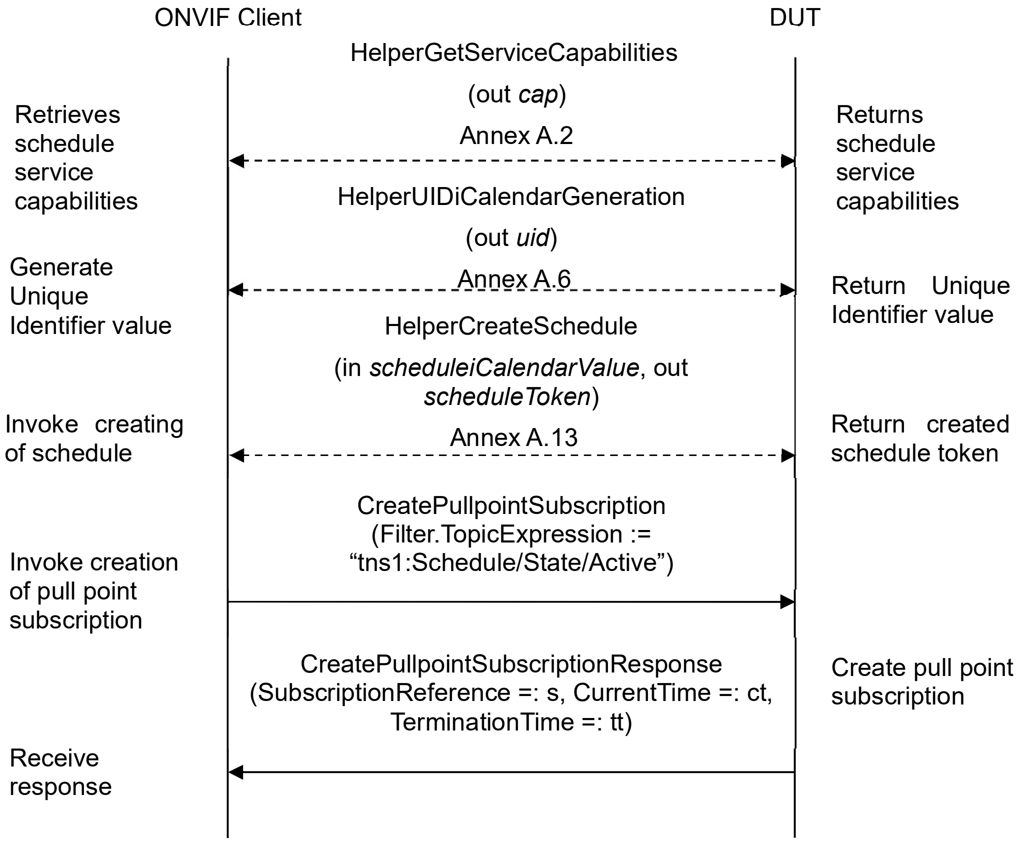
WSDL Reference: schedule.wsdl

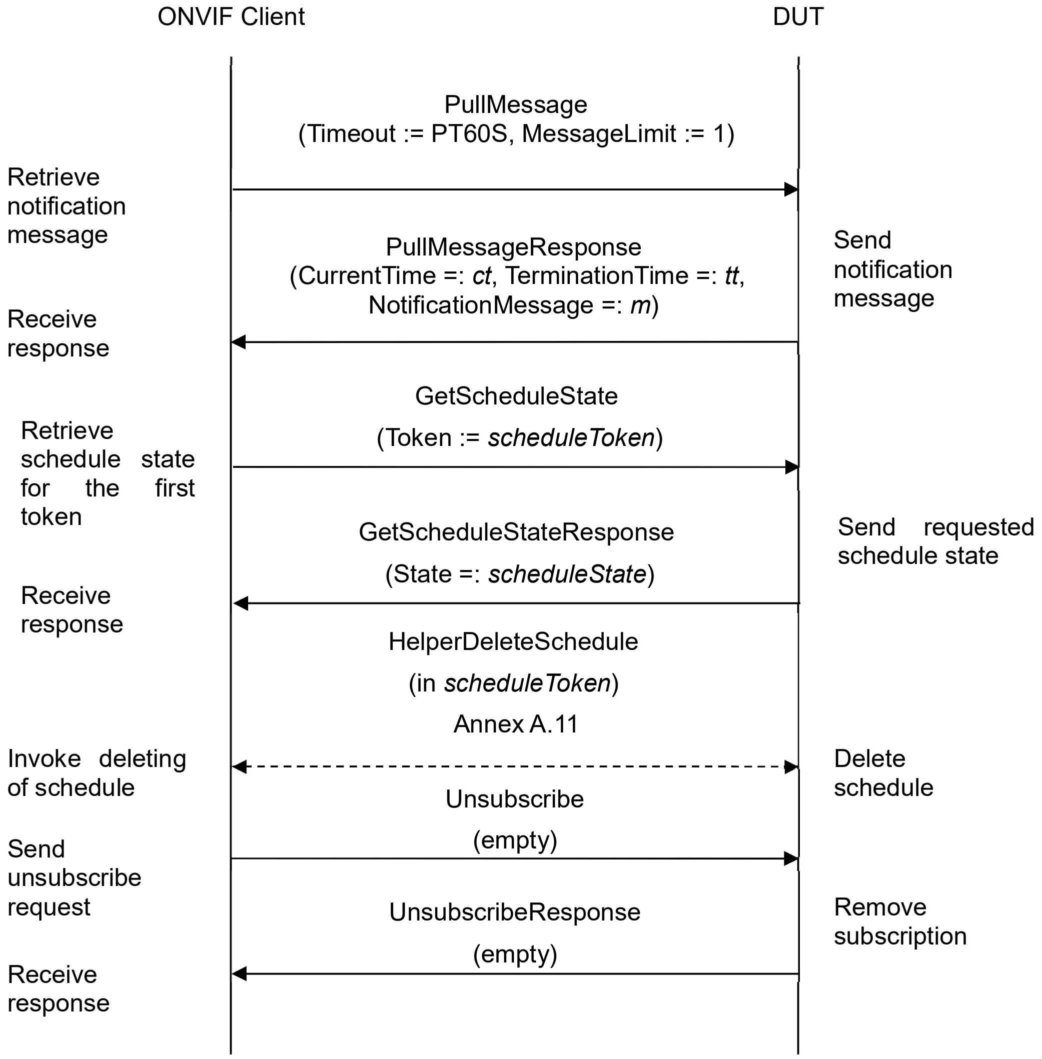
Test Purpose: To verify Get Schedule State and tns1:Schedule/State/Active changed event generation when current time is within the boundaries of the schedule period.

Pre-requisite: Schedule Service is received from the DUT. Event Service was received from the DUT. State Reporting is supported by the DUT as indicated by the Capabilities.StateReportingSupported. The DUT shall have enough free storage capacity for one additional Schedule.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieves schedule service capabilities
Returns schedule service capabilities
HelperUIDiCalendarGeneration
(out *uid*)
Annex A.6
Generate Unique Identifier value
Return Unique Identifier value
HelperCreateSchedule
(in *scheduleiCalendarValue*, out *scheduleToken*)
Annex A.13
Invoke creating of schedule
Return created schedule token
CreatePullpointSubscription (Filter.TopicExpression := "tns1:Schedule/State/Active")
Invoke creation of pull point subscription
CreatePullpointSubscriptionResponse (SubscriptionReference =: s, CurrentTime =: ct, TerminationTime =: tt)
Create pull point subscription
Receive response


ONVIF Client
DUT
PullMessage
(Timeout := PT60S, MessageLimit := 1)
Retrieve notification message
PullMessageResponse
(CurrentTime =: *ct*, TerminationTime =: *tt*, NotificationMessage =: *m*)
Send notification message
Receive response
GetScheduleState
(Token := *scheduleToken*)
Retrieve schedule state for the first token
GetScheduleStateResponse
(State =: *scheduleState*)
Send requested schedule state
Receive response
HelperDeleteSchedule
(in *scheduleToken*)
Annex A.11
Invoke deleting of schedule
Delete schedule
Unsubscribe
(empty)
Send unsubscribe request
UnsubscribeResponse
(empty)
Remove subscription
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

4. ONVIF Client generates Unique Identifier value for UID field in iCalendar (out *uid*) by following the procedure mentioned in Annex A.6.

5. Set the following:

    - *delta* := max{(Message Timeout value, 10 seconds}.

6. If *cap*.ExtendedRecurrenceSupported is equal to true, set the following:

    - *scheduleiCalendarValue* := "BEGIN:VCALENDAR

      BEGIN:VEVENT

      SUMMARY:Test summary

      DTSTART:<current year>< current month>< current day>T<current time + *delta*>

      DTEND:<year of (current day + one week)><month of (current day + one week)><day of (current day + one week)>T180000

      RRULE:FREQ=DAILY

      UID:*uid*

      END:VEVENT

      END:VCALENDAR"

7. If *cap*.ExtendedRecurrenceSupported is equal to false, set the following:

    - *scheduleiCalendarValue* := "BEGIN:VCALENDAR

      BEGIN:VEVENT

      SUMMARY: Test summary

      DTSTART:1970<current month><current day>T<current time + *delta*>

      DTEND:1970<month of (the current day + one week)><day of (the current day + one week)>T180000

      RRULE:FREQ=WEEKLY;BYDAY=<current week day>,<next week day>

      UID:*uid*

      END:VEVENT

      END:VCALENDAR"

8. ONVIF Client creates Schedule with Schedule token (out *scheduleToken*), with iCalendar value of the Schedule.Standard field (in *scheduleiCalendarValue*) and without SpecialDayGroup item by following the procedure mentioned in Annex A.13.

9. ONVIF Client invokes **CreatePullPointSubscription** with parameters

    - Filter.TopicExpression := "tns1:Schedule/State/Active"

    - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"

10. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters

    - SubscriptionReference =: *s*

    - CurrentTime =: *ct*

- TerminationTime =: *tt*

11. Set the following:

    - *timeout* := max{(Operation Delay value, *delta* + 5 seconds}.

12. Until *timeout* timeout expires, repeat the following steps:

    12.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.

    12.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters

    - Timeout := PT60S
    - MessageLimit := 1

    12.3. The DUT responds with **PullMessagesResponse** message with parameters

    - CurrentTime =: *ct*
    - TerminationTime =: *tt*
    - NotificationMessage =: *m*

    12.4. If the TopicExpression item in *m* is not equal to “tns1:Schedule/State/Active”, FAIL the test and go to the step 18.

    12.5. If *m* is not null and PropertyOperation item in *m* is equal to “Changed” check the following:

    12.5.1. If *m* does not contain Source.SimpleItem item with Name = "ScheduleToken" and Value with type is equal to pt:ReferenceToken, FAIL the test and go to the step 18.

    12.5.2. If value of Source.SimpleItem.ScheduleToken item in *m* is equal to *scheduleToken*, check the following:

    12.5.2.1. If *m* does not contain Source.SimpleItem item with Name = "Name" and Value with type is equal to xs:string, FAIL the test and go to the step 18.

    12.5.2.2. If value of Source.SimpleItem.Name item in *m* is not equal to “Test Name”, FAIL the test and go to the step 18.

    12.5.2.3. If *m* does not contain Data.SimpleItem item with Name = "Active" and Value with type is equal to xs:boolean, FAIL the test and go to the step 18.

    12.5.2.4. If value of Data.SimpleItem item with Name = "Active" is not equal to true, FAIL the test and go to the step 18.

    12.5.2.5. If *m* does not contain Data.SimpleItem item with Name = "SpecialDay" and Value with type is equal to xs:boolean, FAIL the test and go to the step 18.

    12.5.2.6. If value of Data.SimpleItem item with Name = "SpecialDay" is not equal to false, FAIL the test and go to the step 18.

    12.5.2.7. Go to step 14.

13. If *timeout* timeout expires for step 11 without Notification with TopicExpression = “tns1:Schedule/State/Active” and with *m* with Source.SimpleItem.ScheduleToken value is equal to *scheduleToken*, FAIL the test and go to the step 18.

14. ONVIF client invokes **GetScheduleState** with parameters

    - Token := *scheduleToken*

15. The DUT responds with **GetScheduleStateResponse** message with parameters

    - ScheduleState =: *scheduleState*

16. If *scheduleState*.Active value is not equal to true, FAIL the test and go to the step 18.

17. If *scheduleState* contains SpecialDays item and *scheduleState*.SpecialDays value is not equal to false, FAIL the test and go to the step 18.

18. ONVIF Client deletes the Schedule (in *scheduleToken*) by following the procedure mentioned in Annex A.11 to restore DUT configuration.

19. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

20. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

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

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

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

Note: If time to the end of the current day is less than *delta*, the test tool waits for the start of the next day before step 7.

Note: Message Timeout value for *delta* calculation will be taken from Message Timeout field of ONVIF Device Test Tool.

Note: Operation Delay value for *timeout* calculation will be taken from Operation Delay field of ONVIF Device Test Tool.

4.6.3 CHANGE SCHEDULE STATE - CHANGE SPECIAL DAYS

Test Label: Get Schedule State Verification

Test Case ID: SCHEDULE-6-1-3

ONVIF Core Specification Coverage: ScheduleState (ONVIF Schedule Service Specification), Get ScheduleState command (ONVIF Schedule Service Specification)

Command Under Test: GetScheduleState

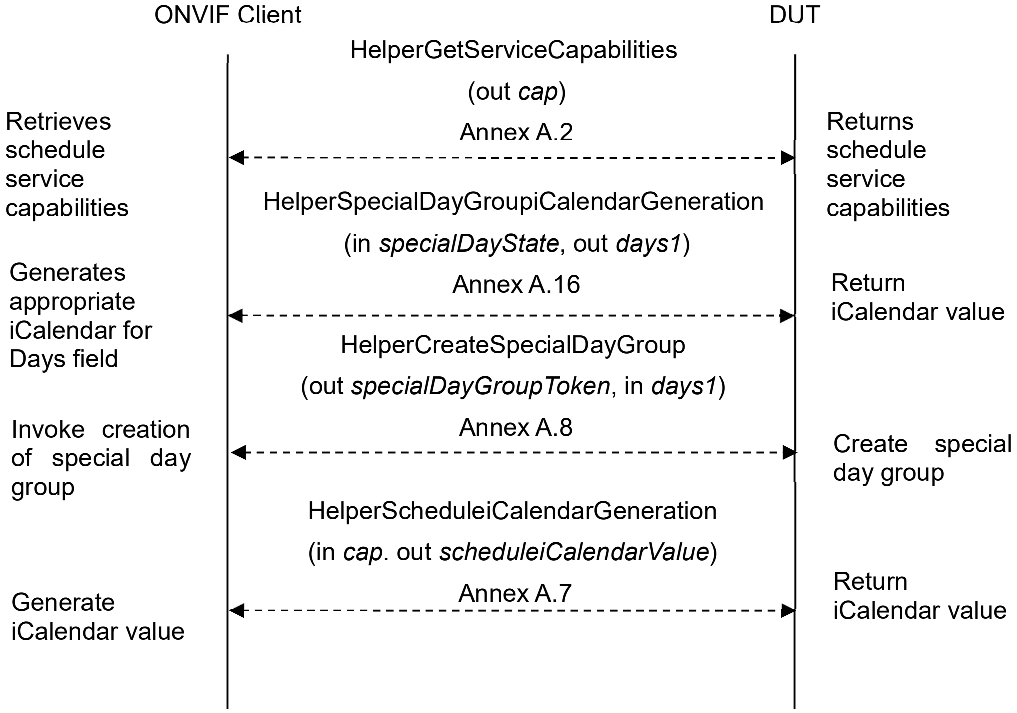
WSDL Reference: schedule.wsdl

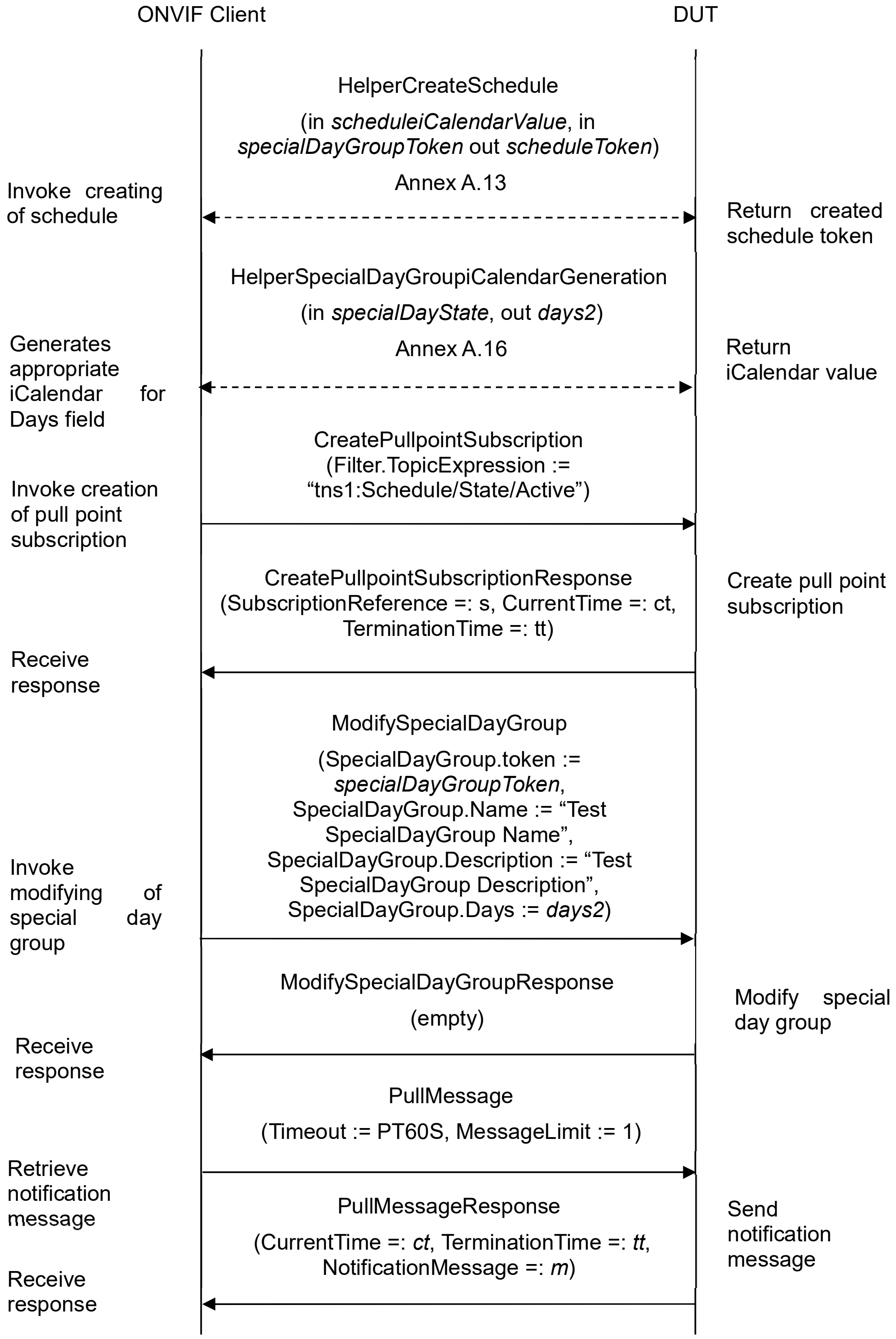
Test Purpose: To verify Get Schedule State and tns1:Schedule/State/Active changed event generation when current time is within the boundaries of the special day time period.

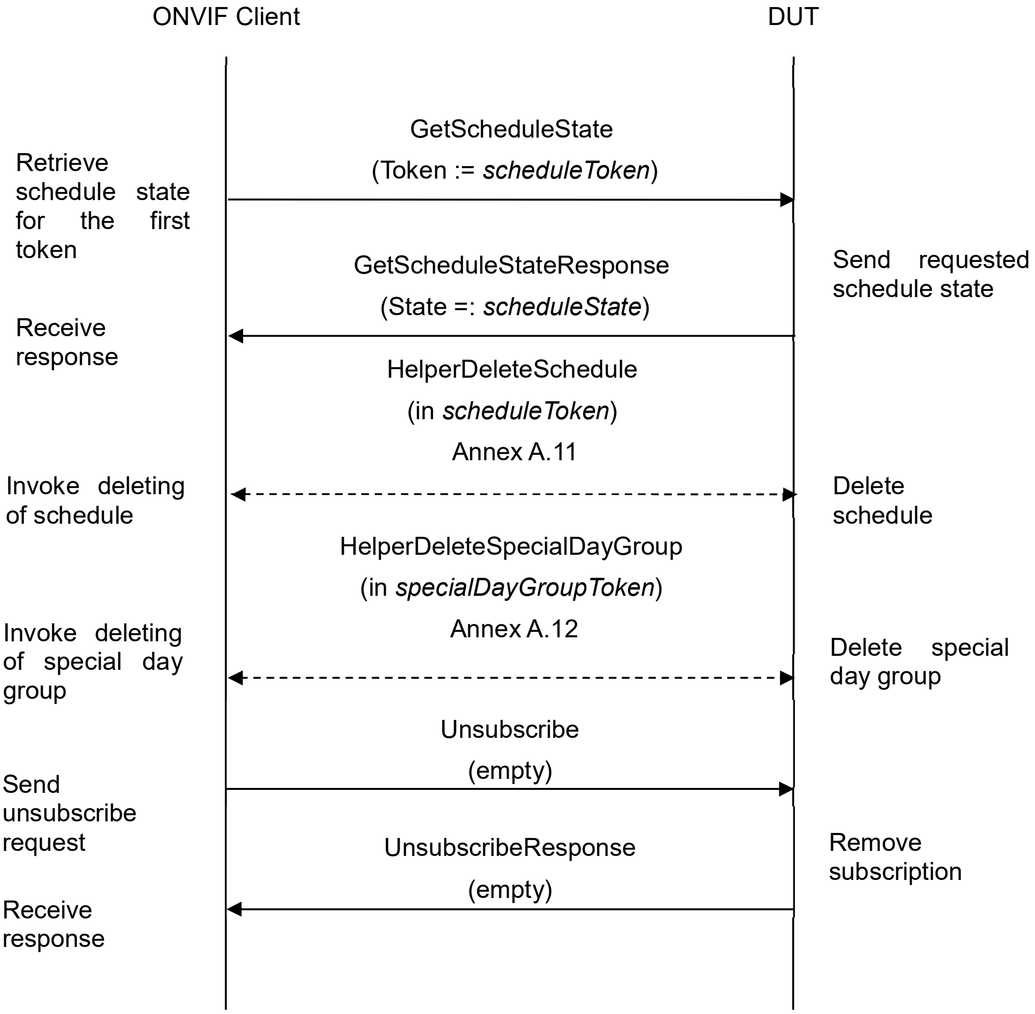
Pre-requisite: Schedule Service is received from the DUT. Event Service was received from the DUT. State Reporting is supported by the DUT as indicated by the Capabilities.StateReportingSupported. The DUT shall have enough free storage capacity for one additional Schedule. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported. The DUT shall have enough free storage capacity for one additional SpecialDayGroup.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieves schedule service capabilities
Returns schedule service capabilities
HelperSpecialDayGroupiCalendarGeneration
(in *specialDayState*, out *days1*)
Annex A.16
Generates appropriate iCalendar for Days field
Return iCalendar value
HelperCreateSpecialDayGroup
(out *specialDayGroupToken*, in *days1*)
Annex A.8
Invoke creation of special day group
Create special day group
HelperScheduleiCalendarGeneration
(in *cap*. out *scheduleiCalendarValue*)
Annex A.7
Generate iCalendar value
Return iCalendar value


ONVIF Client
DUT
HelperCreateSchedule
(in *scheduleiCalendarValue*, in *specialDayGroupToken* out *scheduleToken*)
Annex A.13
Invoke creating of schedule
Return created schedule token
HelperSpecialDayGroupiCalendarGeneration
(in *specialDayState*, out *days2*)
Annex A.16
Generates appropriate iCalendar for Days field
Return iCalendar value
CreatePullpointSubscription
(Filter.TopicExpression := “tns1:Schedule/State/Active”)
Invoke creation of pull point subscription
CreatePullpointSubscriptionResponse
(SubscriptionReference =: s, CurrentTime =: ct, TerminationTime =: tt)
Create pull point subscription
Receive response
ModifySpecialDayGroup
(SpecialDayGroup.token := *specialDayGroupToken*, SpecialDayGroup.Name := “Test SpecialDayGroup Name”, SpecialDayGroup.Description := “Test SpecialDayGroup Description”, SpecialDayGroup.Days := *days2*)
Invoke modifying of special day group
ModifySpecialDayGroupResponse
(empty)
Modify special day group
Receive response
PullMessage
(Timeout := PT60S, MessageLimit := 1)
Retrieve notification message
PullMessageResponse
(CurrentTime =: *ct*, TerminationTime =: *tt*, NotificationMessage =: *m*)
Send notification message
Receive response



Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.
4. Set the following:
   - *specialDayState* := false
5. ONVIF Client generates appropriate iCalendar for Days field in special day group (out *days1*) with boundaries of special days time period in iCalendar are not contained current time (in *specialDayState*) by following the procedure mentioned in Annex A.16.

6. ONVIF Client creates SpecialDayGroup (out *specialDayGroupToken*) with required iCalendar value (in *days1*) by following the procedure mentioned in Annex A.8.
7. ONVIF Client generates appropriate iCalendar value for the Schedule.Standard field (in *cap*, out *scheduleiCalendarValue*) by following the procedure mentioned in Annex A.7.
8. ONVIF Client creates Schedule with Schedule token (out *scheduleToken*), with iCalendar value of the Schedule.Standard field (in *scheduleiCalendarValue*) and with SpecialDayGroupToken (in *specialDayGroupToken*) by following the procedure mentioned in Annex A.13.
9. Set the following:
   - *specialDayState* := true
10. ONVIF Client generates appropriate iCalendar for Days field in special day group (out *days2*) with boundaries of special days time period in iCalendar are contained current time (in *specialDayState*) by following the procedure mentioned in Annex A.16.
11. ONVIF Client invokes **CreatePullPointSubscription** with parameters
    - Filter.TopicExpression := “tns1:Schedule/State/Active”
    - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
12. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
    - SubscriptionReference =: *s*
    - CurrentTime =: *ct*
    - TerminationTime =: *tt*
13. ONVIF client invokes **ModifySpecialDayGroup** with parameters
    - SpecialDayGroup.token := “*specialDayGroupToken*”
    - SpecialDayGroup.Name := “Test SpecialDayGroup Name”
    - SpecialDayGroup.Description := “Test SpecialDayGroup Description”
    - SpecialDayGroup.Days := *days2*
14. The DUT responds with empty **ModifySpecialDayGroupResponse** message.
15. Until *timeout* timeout expires, repeat the following steps:

    15.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.

    15.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters

    - Timeout := PT60S
    - MessageLimit := 1

    15.3. The DUT responds with **PullMessagesResponse** message with parameters

    - CurrentTime =: *ct*

- TerminationTime =: *tt*
- NotificationMessage =: *m*

15.4. If the TopicExpression item in *m* is not equal to “tns1:Schedule/State/Active”, FAIL the test and go to the step 22.

15.5. If *m* is not null and PropertyOperation item in *m* is equal to “Changed” check the following:

15.5.1. If *m* does not contain Source.SimpleItem item with Name = "ScheduleToken" and Value with type is equal to pt:ReferenceToken, FAIL the test and go to the step 22.

15.5.2. If value of Source.SimpleItem.ScheduleToken item in *m* is equal to *scheduleToken*, check the following:

15.5.2.1. If *m* does not contain Source.SimpleItem item with Name = "Name" and Value with type is equal to xs:string, FAIL the test and go to the step 22.

15.5.2.2. If value of Source.SimpleItem.Name item in *m* is not equal to “Test Name”, FAIL the test and go to the step 22.

15.5.2.3. If *m* does not contain Data.SimpleItem item with Name = "Active" and Value with type is equal to xs:boolean, FAIL the test and go to the step 22.

15.5.2.4. If value of Data.SimpleItem item with Name = "Active" is not equal to true, FAIL the test and go to the step 22.

15.5.2.5. If *m* does not contain Data.SimpleItem item with Name = "SpecialDay" and Value with type is equal to xs:boolean, FAIL the test and go to the step 22.

15.5.2.6. If value of Data.SimpleItem item with Name = "SpecialDay" is not equal to true, FAIL the test and go to the step 22.

15.5.2.7. Go to step 17.

16. If *timeout* timeout expires for step 15 without Notification with TopicExpression = “tns1:Schedule/State/Active” and with *m* with Source.SimpleItem.ScheduleToken value is equal to *scheduleToken*, FAIL the test and go to the step 22.

17. ONVIF client invokes **GetScheduleState** with parameters

    - Token := *scheduleToken*

18. The DUT responds with **GetScheduleStateResponse** message with parameters

    - ScheduleState =: *scheduleState*

19. If *scheduleState*.Active value is not equal to true, FAIL the test and go to the step 22.

20. If *scheduleState* does not contain SpecialDays item, FAIL the test and go to the step 22.

21. If *scheduleState*.SpecialDays value is not equal to true, FAIL the test and go to the step 22.

22. ONVIF Client deletes the Schedule (in *scheduleToken*) by following the procedure mentioned in Annex A.11 to restore DUT configuration.

23. ONVIF Client deletes the SpecialDayGroup (in *specialDayGroupToken*) by following the procedure mentioned in Annex A.12 to restore DUT configuration.

24. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

25. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

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

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

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

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

4.6.4 GET SCHEDULE STATE WITH INVALID TOKEN

Test Label: Get Schedule State with Invalid Token Verification

Test Case ID: SCHEDULE-6-1-4

ONVIF Core Specification Coverage: ScheduleState (ONVIF Schedule Service Specification), Get ScheduleState command (ONVIF Schedule Service Specification)

Command Under Test: GetScheduleState

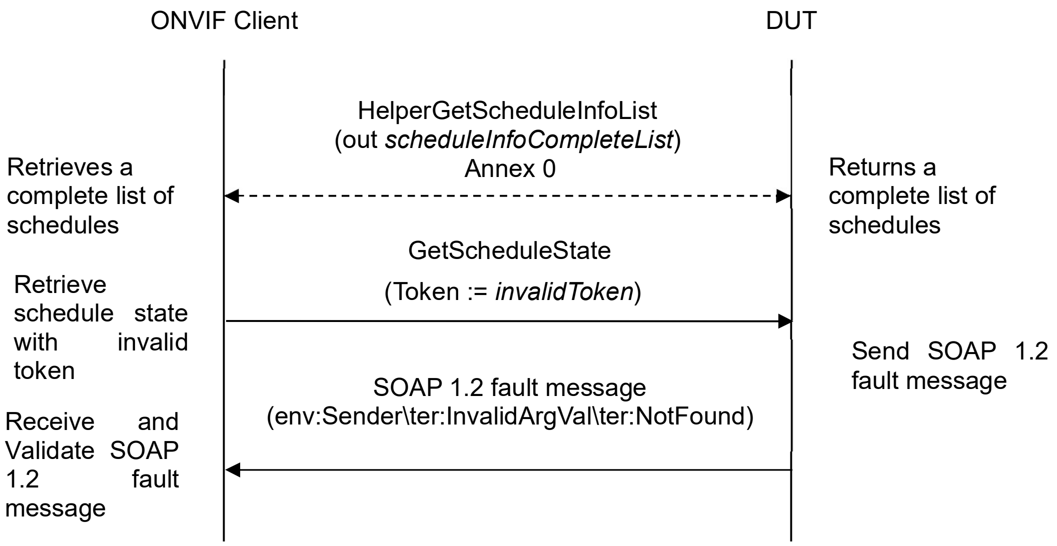
WSDL Reference: schedule.wsdl

Test Purpose: To verify Get Schedule State with invalid token.

Pre-requisite: Schedule Service is received from the DUT. State Reporting is supported by the DUT as indicated by the Capabilities.StateReportingSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetScheduleInfoList
(out *scheduleInfoCompleteList*)
Annex 0
Retrieves a complete list of schedules
Returns a complete list of schedules
GetScheduleState
(Token := *invalidToken*)
Retrieve schedule state with invalid token
Send SOAP 1.2 fault message
SOAP 1.2 fault message
(env:Sender\ter:InvalidArgVal\ter:NotFound)
Receive and Validate SOAP 1.2 fault message

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of schedule info (out *scheduleInfoCompleteList*) by following the procedure mentioned in Annex 0.

4. Set the following:

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

5. ONVIF client invokes **GetScheduleState** with parameters

    - Token := *invalidToken*

6. The DUT returns **env:Sender\ter:InvalidArgVal\ter:NotFound** SOAP 1.2 fault.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

Note: If the DUT sends other SOAP 1.2 fault message than specified, log WARNING message, and PASS the test.

4.7 Schedule Events

4.7.1 SCHEDULE STATE ACTIVE EVENT (PROPERTY EVENT)

Test Label: State Active Event Verification

Test Case ID: SCHEDULE-7-1-1

ONVIF Core Specification Coverage: Schedule State (ONVIF Schedule Service Specification), Notification topics (ONVIF Schedule Service Specification), Get event properties (ONVIF Core specification).

Command Under Test: GetEventProperties

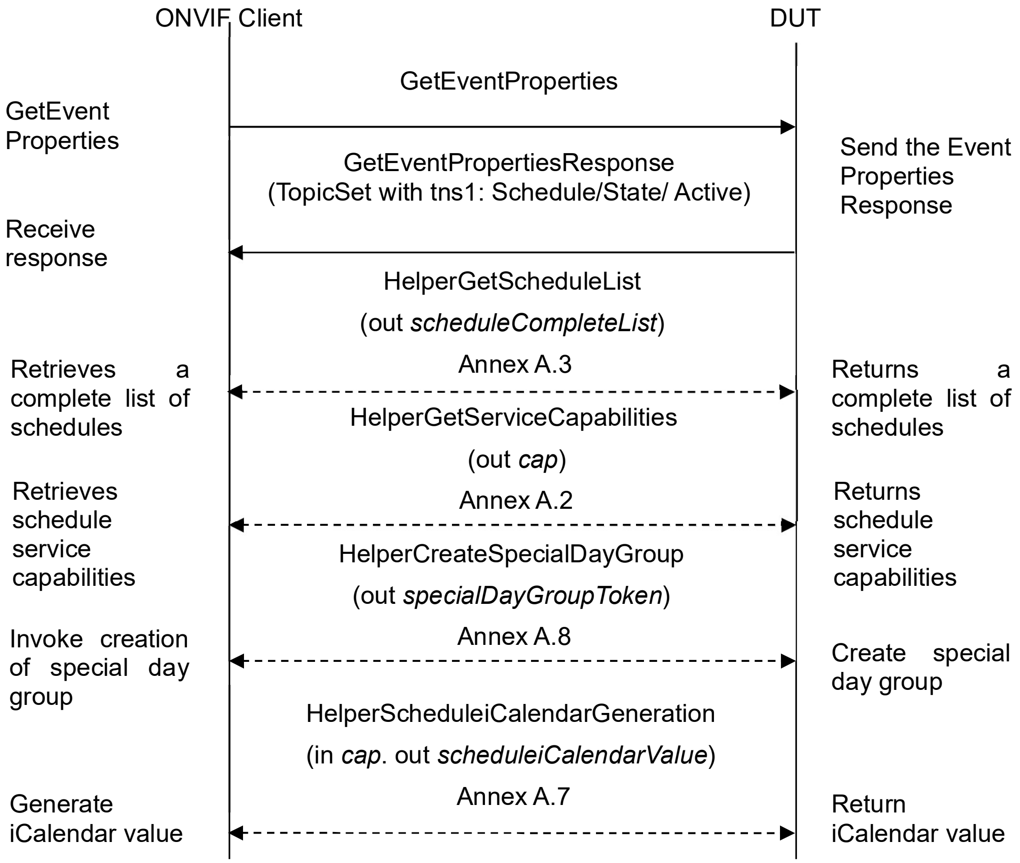
WSDL Reference: schedule.wsdl and event.wsdl

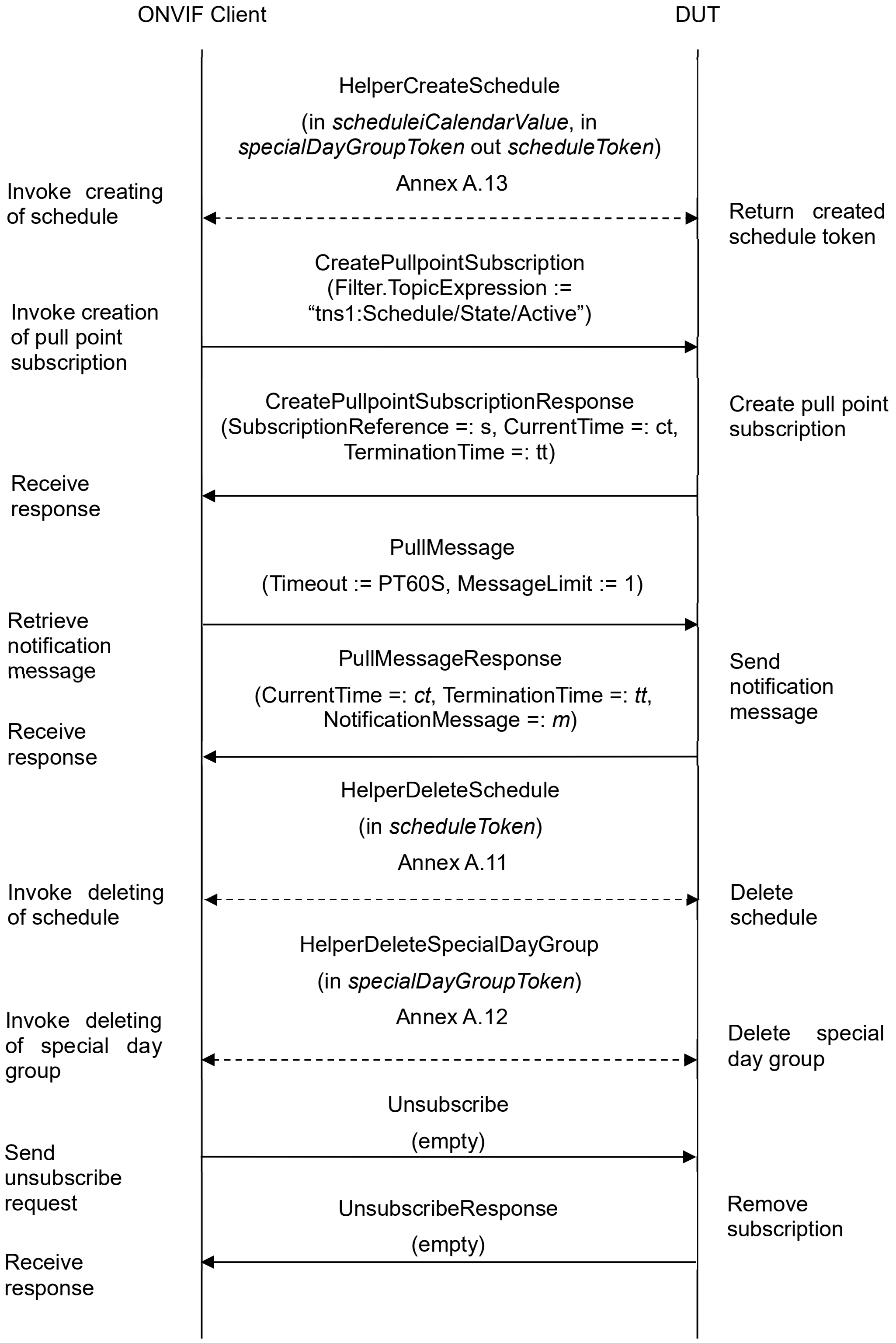
Test Purpose: To verify tns1:Schedule/State/Active event format and and initialize event generation.

Pre-requisite: Schedule Service is received from the DUT. Event Service is received from the DUT. State Reporting is supported by the DUT as indicated by the Capabilities.StateReportingSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
GetEventProperties
GetEvent Properties
GetEventPropertiesResponse (TopicSet with tns1: Schedule/State/ Active)
Send the Event Properties Response
Receive response
HelperGetScheduleList
(out *scheduleCompleteList*)
Annex A.3
Retrieves a complete list of schedules
Returns a complete list of schedules
HelperGetServiceCapabilities
(out *cap*)
Annex A.2
Retrieves schedule service capabilities
Returns schedule service capabilities
HelperCreateSpecialDayGroup
(out *specialDayGroupToken*)
Annex A.8
Invoke creation of special day group
Create special day group
HelperScheduleiCalendarGeneration
(in *cap*. out *scheduleiCalendarValue*)
Annex A.7
Generate iCalendar value
Return iCalendar value


ONVIF Client
DUT
HelperCreateSchedule
(in *scheduleiCalendarValue*, in *specialDayGroupToken* out *scheduleToken*)
Annex A.13
Invoke creating of schedule
Return created schedule token
CreatePullpointSubscription
(Filter.TopicExpression := “tns1:Schedule/State/Active”)
Invoke creation of pull point subscription
CreatePullpointSubscriptionResponse
(SubscriptionReference =: s, CurrentTime =: ct, TerminationTime =: tt)
Create pull point subscription
Receive response
PullMessage
(Timeout := PT60S, MessageLimit := 1)
Retrieve notification message
PullMessageResponse
(CurrentTime =: *ct*, TerminationTime =: *tt*, NotificationMessage =: *m*)
Send notification message
Receive response
HelperDeleteSchedule
(in *scheduleToken*)
Annex A.11
Invoke deleting of schedule
Delete schedule
HelperDeleteSpecialDayGroup
(in *specialDayGroupToken*)
Annex A.12
Invoke deleting of special day group
Delete special day group
Unsubscribe
(empty)
Send unsubscribe request
UnsubscribeResponse
(empty)
Remove subscription
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client invokes **GetEventProperties**.

4. The DUT responds with a **GetEventPropertiesResponse** message with parameters

    - TopicNamespaceLocation list

    - FixedTopicSet

    - TopicSet =: *topicSet*

    - TopicExpressionDialect list

    - MessageContentFilterDialect list

    - MessageContentSchemaLocation list

5. If *topicSet* does not contain tns1:Schedule/State/Active topic, FAIL the test and skip other steps.

6. ONVIF Client verifies tns1:Schedule/State/Active topic (*stateActiveTopic*) from *topicSet*:

    6.1. If *stateActiveTopic*.MessageDescription.IsProperty equals to false or skipped, FAIL the test and skip other steps.

    6.2. If *stateActiveTopic* does not contain MessageDescription.Source.SimpleItemDescription item with Name =: “ScheduleToken”, FAIL the test and skip other steps.

    6.3. If *stateActiveTopic*.MessageDescription.Source.SimpleItemDescription with Name =: “ScheduleToken” does not have Type =: “pt:ReferenceToken”, FAIL the test and skip other steps.

    6.4. If *stateActiveTopic* does not contain MessageDescription.Source.SimpleItemDescription item with Name =: “Name”, FAIL the test and skip other steps.

    6.5. If *stateActiveTopic*.MessageDescription.Source.SimpleItemDescription with Name =: “Name” does not have Type =: “xs:string”, FAIL the test and skip other steps.

    6.6. If *stateActiveTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name =: “Active”, FAIL the test and skip other steps.

    6.7. If *stateActiveTopic.*MessageDescription.Data.SimpleItemDescription with Name =: “Active” does not have Type =: “xs:boolean”, FAIL the test and skip other steps.

    6.8. If *stateActiveTopic* does not contain MessageDescription.Data.SimpleItemDescription item with Name =: “SpecialDay”, FAIL the test and skip other steps.

    6.9. If *stateActiveTopic*.MessageDescription.Data.SimpleItemDescription with Name =: “SpecialDay” does not have Type =: “xs:boolean”, FAIL the test and skip other steps.

7. ONVIF Client retrieves a complete list of Schedules (out *scheduleCompleteList*) by following the procedure mentioned in Annex A.3.

8. If scheduleCompleteList is not empty, go to step 14.

9. ONVIF Client gets the service capabilities (out *cap*) by following the procedure mentioned in Annex A.2.

10. If *cap*.SpecialDaysSupported is equal to true, ONVIF Client creates SpecialDayGroup (out *specialDayGroupToken*) by following the procedure mentioned in Annex A.8.

11. ONVIF Client generates appropriate iCalendar value for the Schedule.Standard field (in *cap*, out *scheduleiCalendarValue*) by following the procedure mentioned in Annex A.7.

12. ONVIF Client creates Schedule with Schedule token (out *scheduleToken*), with iCalendar value of the Schedule.Standard field (in *scheduleiCalendarValue*) and with SpecialDayGroupToken (in *specialDayGroupToken*) if Special Days is supported by the DUT by following the procedure mentioned in Annex A.13.

13. Set:

    - *scheduleCompleteList* := *scheduleToken*

14. ONVIF Client invokes **CreatePullPointSubscription** with parameters

    - Filter.TopicExpression := “tns1:Schedule/State/Active”
    - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"

15. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters

    - SubscriptionReference =: *s*
    - CurrentTime =: *ct*
    - TerminationTime =: *tt*

16. Until *timeout1* timeout expires, repeat the following steps:

    16.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.

    16.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters

    - Timeout := PT60S
    - MessageLimit := 1

    16.3. The DUT responds with **PullMessagesResponse** message with parameters

    - CurrentTime =: *ct*
    - TerminationTime =: *tt*
    - NotificationMessage =: *m*

    16.4. If *m* is not null and the TopicExpression item in *m* is not equal to “tns1:Schedule/State/Active”, FAIL the test and go to the step 19.

    16.5. If *m* is not null and PropertyOperation item in *m* is equal to “Initialized” check the following:

    16.5.1. If *m* does not contain Source.SimpleItem item with Name = "ScheduleToken" and Value with type is equal to pt:ReferenceToken, FAIL the test and go to the step 19.

16.5.2. If value of ScheduleToken item (*scheduleToken*) in *m* is not equal to any Schedule token from *scheduleCompleteList*, FAIL the test and go to the step 19.

16.5.3. If *m* does not contain Source.SimpleItem item with Name = "Name" and Value with type is equal to xs:string, FAIL the test and go to the step 19.

16.5.4. If value of Name item in *m* is not equal to value of Name of Schedule[token = *scheduleToken*] from *scheduleCompleteList*, FAIL the test and go to the step 19.

16.5.5. If *m* does not contain Data.SimpleItem item with Name = "Active" and Value with type is equal to xs:boolean, FAIL the test and go to the step 19.

16.5.6. If *m* does not contain Data.SimpleItem item with Name = "SpecialDay" and Value with type is equal to xs:boolean, FAIL the test and go to the step 19.

16.5.7. If notifications for all schedule tokens from *scheduleCompleteList* are received, go to step 18.

17. If *timeout1* timeout expires for step 16 without Notification with TopicExpression := “tns1:Schedule/State/Active”, FAIL the test and go to step 19.

18. If the DUT sent Notifications not for all schedule tokens from *scheduleCompleteList*, FAIL the test and go to step 19.

19. If Schedule was created at step 12, ONVIF Client deletes the Schedule (in *scheduleToken*) by following the procedure mentioned in Annex A.11 to restore DUT configuration.

20. If SpecialDayGroup was created at step 10, ONVIF Client deletes the SpecialDayGroup (in *specialDayGroupToken*) by following the procedure mentioned in Annex A.12 to restore DUT configuration.

21. ONVIF Client sends an **Unsubscribe** to the subscription endpoint *s*.

22. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

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

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

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

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

4.7.2 SCHEDULE CHANGED EVENT

Test Label: Schedule Changed Event Verification

Test Case ID: SCHEDULE-7-1-2

ONVIF Core Specification Coverage: Schedule State (ONVIF Schedule Service Specification), Notification topics (ONVIF Schedule Service Specification), Get event properties (ONVIF Core specification).

Command Under Test: GetEventProperties

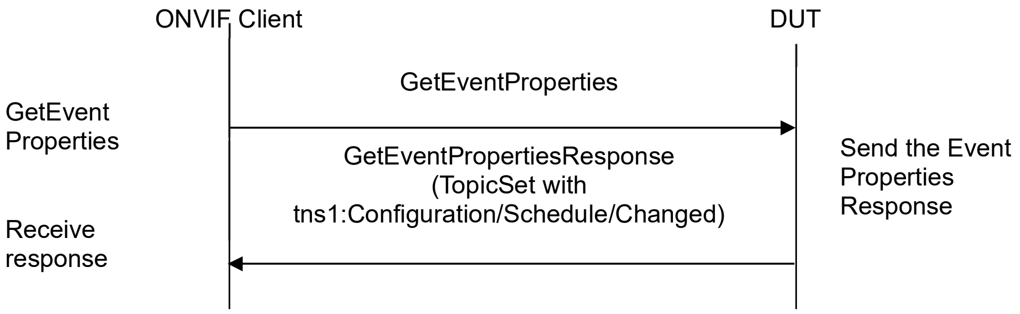
WSDL Reference: event.wsdl

Test Purpose: To verify tns1:Configuration/Schedule/Changed event format.

Pre-requisite: Schedule Service is supported by the DUT. Event Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
GetEventProperties
GetEvent Properties
GetEventPropertiesResponse (TopicSet with tns1:Configuration/Schedule/Changed)
Send the Event Properties Response
Receive response

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
    - TopicNamespaceLocation list
    - FixedTopicSet
    - TopicSet =: *topicSet*
    - TopicExpressionDialect list
    - MessageContentFilterDialect list
    - MessageContentSchemaLocation list

5. If *topicSet* does not contain tns1:Configuration/Schedule/Changed topic, FAIL the test and skip other steps.

6. ONVIF Client verifies tns1:Configuration/Schedule/Changed topic (*scheduleChangedTopic*) from *topicSet*:

   6.1. If *scheduleChangedTopic*.MessageDescription.IsProperty equals to true, FAIL the test and skip other steps.

   6.2. If *scheduleChangedTopic* does not contain MessageDescription.Source.SimpleItemDescription item with Name =: “ScheduleToken”, FAIL the test and skip other steps.

   6.3. If *scheduleChangedTopic.*MessageDescription.Source.SimpleItemDescription with Name =: “ScheduleToken” does not have Type =: “pt:ReferenceToken”, FAIL the test.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

4.7.3 SCHEDULE REMOVED EVENT

Test Label: Schedule Removed Event Verification

Test Case ID: SCHEDULE-7-1-3

ONVIF Core Specification Coverage: Notification topics (ONVIF Schedule Service Specification), Get event properties (ONVIF Core specification).

Command Under Test: GetEventProperties

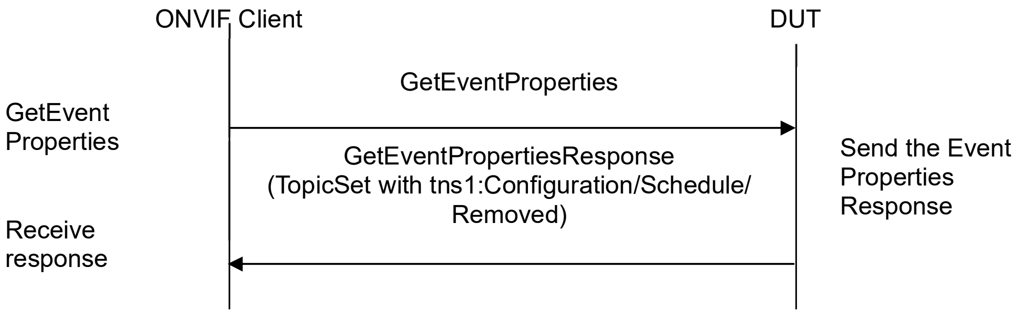
WSDL Reference: event.wsdl

Test Purpose: To verify tns1:Configuration/Schedule/Removed event format.

Pre-requisite: Schedule Service is supported by the DUT. Event Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
GetEventProperties
GetEvent Properties
GetEventPropertiesResponse (TopicSet with tns1:Configuration/Schedule/ Removed)
Send the Event Properties Response
Receive response

Test Procedure:

7. Start an ONVIF Client.

8. Start the DUT.

9. ONVIF Client invokes **GetEventProperties**.

10. The DUT responds with a **GetEventPropertiesResponse** message with parameters

    - TopicNamespaceLocation list
    - FixedTopicSet
    - TopicSet =: *topicSet*
    - TopicExpressionDialect list
    - MessageContentFilterDialect list
    - MessageContentSchemaLocation list

11. If *topicSet* does not contain tns1:Configuration/Schedule/Removed topic, FAIL the test and skip

other steps.

12. ONVIF Client verifies tns1:Configuration/Schedule/Removed topic (*scheduleRemovedTopic*) from *topicSet*:

    12.1. If *scheduleRemovedTopic*.MessageDescription.IsProperty equals to true, FAIL the test and skip other steps.

    12.2. If *scheduleRemovedTopic* does not contain MessageDescription.Source.SimpleItemDescription item with Name =: "ScheduleToken", FAIL the test and skip other steps.

    12.3. If *scheduleRemovedTopic.*MessageDescription.Source.SimpleItemDescription with Name =: "ScheduleToken" does not have Type =: "pt:ReferenceToken", FAIL the test.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

4.7.4 SPECIAL DAYS CHANGED EVENT

Test Label: Special Days Changed Event Verification

Test Case ID: SCHEDULE-7-1-4

ONVIF Core Specification Coverage: Notification topics (ONVIF Schedule Service Specification), Get event properties (ONVIF Core specification).

Command Under Test: GetEventProperties

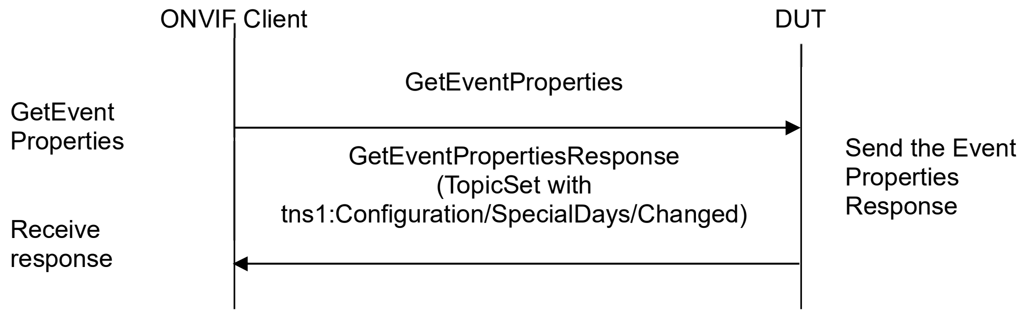
WSDL Reference: event.wsdl

Test Purpose: To verify tns1:Configuration/SpecialDays/Changed event format.

Pre-requisite: Schedule Service is supported by the DUT. Event Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
GetEventProperties
GetEvent Properties
GetEventPropertiesResponse (TopicSet with tns1:Configuration/SpecialDays/Changed)
Send the Event Properties Response
Receive response

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties**.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list

5. If *topicSet* does not contain tns1:Configuration/SpecialDays/Changed topic, FAIL the test and skip other steps.

6. ONVIF Client verifies tns1:Configuration/SpecialDays/Changed topic (*specialDaysChangedTopic*) from *topicSet*:

   6.1. If *specialDaysChangedTopic*.MessageDescription.IsProperty equals to true, FAIL the test and skip other steps.

   6.2. If *specialDaysChangedTopic* does not contain MessageDescription.Source.SimpleItemDescription item with Name =: “SpecialDaysToken”, FAIL the test and skip other steps.

   6.3. If *specialDaysChangedTopic.*MessageDescription.Source.SimpleItemDescription with Name =: “SpecialDaysToken” does not have Type =: “pt:ReferenceToken”, FAIL the test.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

4.7.5 SPECIAL DAYS REMOVED EVENT

Test Label: Special Days Removed Event Verification

Test Case ID: SCHEDULE-7-1-5

ONVIF Core Specification Coverage: Notification topics (ONVIF Schedule Service Specification), Get event properties (ONVIF Core specification).

Command Under Test: GetEventProperties

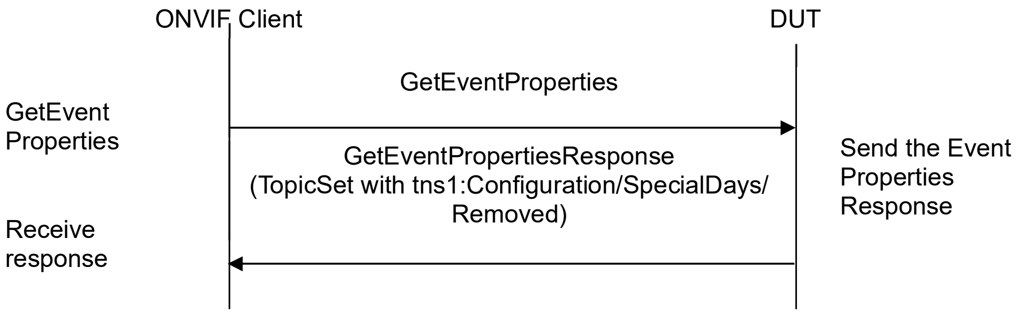
WSDL Reference: event.wsdl

Test Purpose: To verify tns1:Configuration/SpecialDays/Removed event format.

Pre-requisite: Schedule Service is supported by the DUT. Event Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
GetEventProperties
GetEvent Properties
GetEventPropertiesResponse (TopicSet with tns1:Configuration/SpecialDays/ Removed)
Send the Event Properties Response
Receive response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client invokes **GetEventProperties**.

4. The DUT responds with a **GetEventPropertiesResponse** message with parameters

   - TopicNamespaceLocation list

   - FixedTopicSet

   - TopicSet =: *topicSet*

   - TopicExpressionDialect list

   - MessageContentFilterDialect list

   - MessageContentSchemaLocation list

5. If *topicSet* does not contain tns1:Configuration/SpecialDays/Removed topic, FAIL the test and skip other steps.

6. ONVIF Client verifies tns1:Configuration/SpecialDays/Removed topic (*specialDaysRemovedTopic*) from *topicSet*:

    6.1. If *specialDaysRemovedTopic*.MessageDescription.IsProperty equals to true, FAIL the test and skip other steps.

    6.2. If *specialDaysRemovedTopic* does not contain MessageDescription.Source.SimpleItemDescription item with Name =: “SpecialDaysToken”, FAIL the test and skip other steps.

    6.3. If *specialDaysRemovedTopic.*MessageDescription.Source.SimpleItemDescription with Name =: “SpecialDaysToken” does not have Type =: “pt:ReferenceToken”, FAIL the test.

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

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

4.8 Consistency

4.8.1 GET SCHEDULE AND GET SPECIAL DAY GROUP INFO LIST CONSISTENCY

Test Label: Get Schedules List and Special Day Group Info List Consistency Verification

Test Case ID: SCHEDULE-8-1-1

ONVIF Core Specification Coverage: Schedule (ONVIF Schedule Service Specification), Special day group (ONVIF Schedule Specification).

Command Under Test: GetScheduleList, GetSpecialDayGroupInfoList

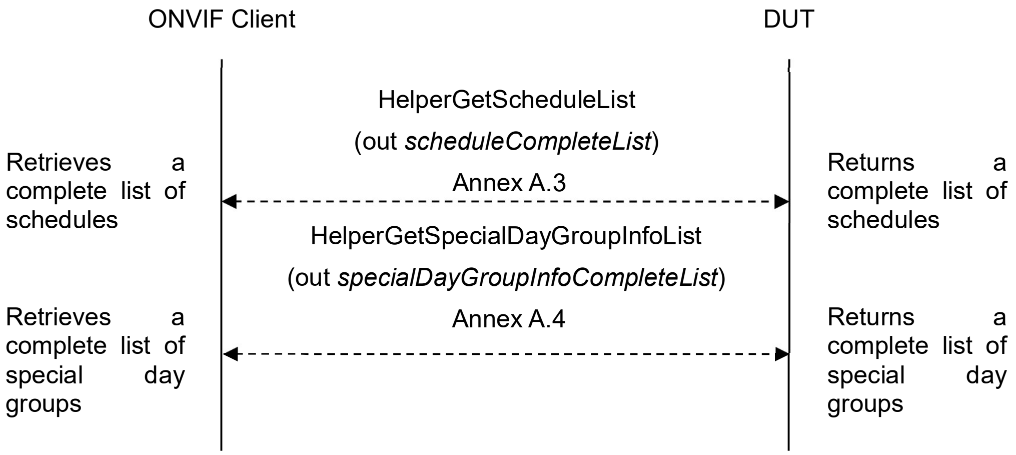
WSDL Reference: schedule.wsdl

Test Purpose: To verify that all Special Day Group Tokens from GetScheduleResponses could be listed through GetSpecialDayGroupInfoList command.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
HelperGetScheduleList
(out *scheduleCompleteList*)
Annex A.3
Retrieves a complete list of schedules
Returns a complete list of schedules
HelperGetSpecialDayGroupInfoList
(out *specialDayGroupInfoCompleteList*)
Annex A.4
Retrieves a complete list of special day groups
Returns a complete list of special day groups

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a complete list of schedules (out *scheduleCompleteList*) by following the procedure mentioned in Annex A.3.

4. ONVIF Client retrieves a complete list of special day group information (out *specialDayGroupInfoCompleteList*) by following the procedure mentioned in Annex A.4.

5. For each Schedule.SpecialDays.GroupToken (*specialDaysGroupToken*) from

scheduleCompleteList repeat the following steps:

5.1. If *specialDaysGroupToken* is not listed in *specialDayGroupInfoCompleteList*, FAIL the test and skip other steps.

Test Result:

PASS –

The DUT passed all assertions.

Annex A

This section describes the meaning of the following definitions. These definitions are used in the test case description.

A.1 Get schedules information list

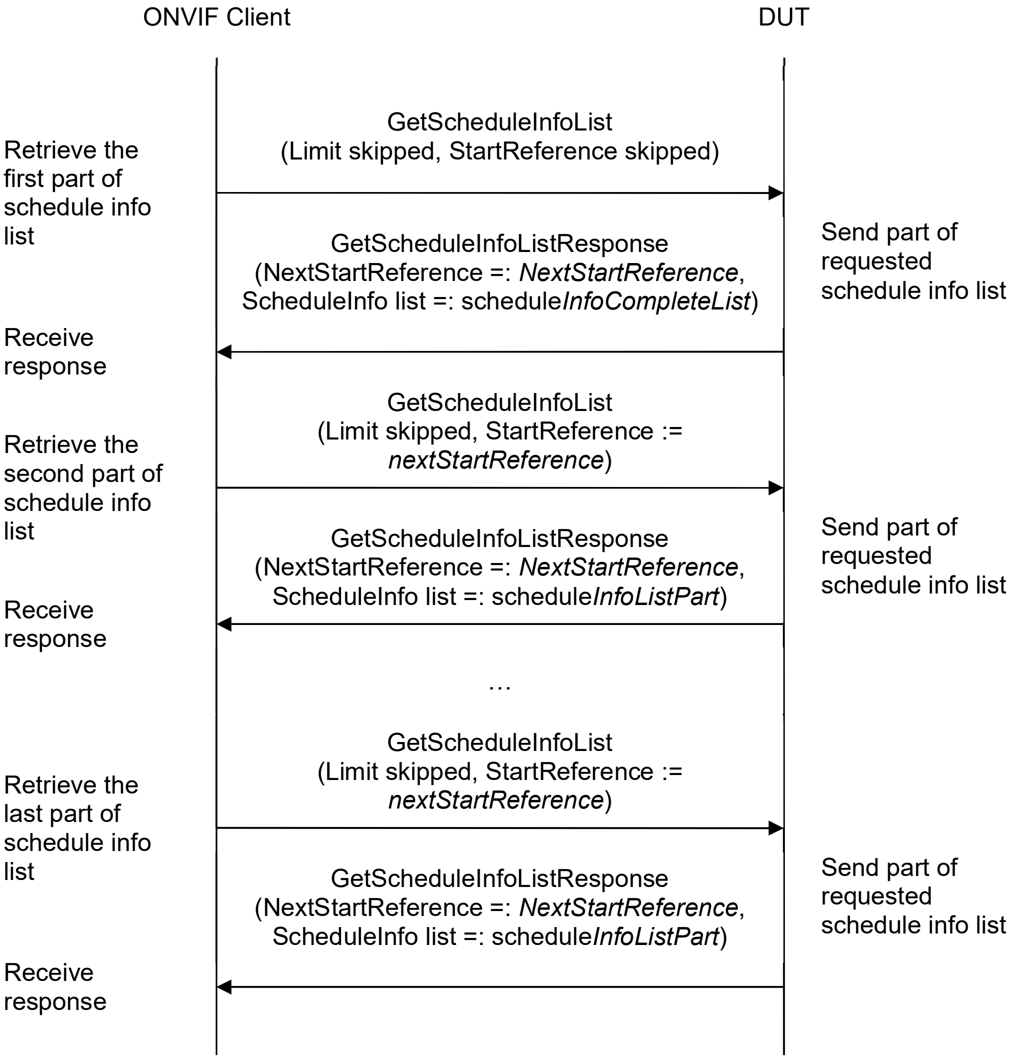
Name: HelperGetScheduleInfoList

Procedure Purpose: Helper procedure to get complete schedules information list.

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

Input: None.

Returns: The complete list of schedules information (*scheduleInfoCompleteList*).


ONVIF Client
DUT
Retrieve the first part of schedule info list
GetScheduleInfoList (Limit skipped, StartReference skipped)
GetScheduleInfoListResponse (NextStartReference =: *NextStartReference*, ScheduleInfo list =: schedule*InfoCompleteList*)
Send part of requested schedule info list
Receive response
Retrieve the second part of schedule info list
GetScheduleInfoList (Limit skipped, StartReference := *nextStartReference*)
GetScheduleInfoListResponse (NextStartReference =: *NextStartReference*, ScheduleInfo list =: schedule*InfoListPart*)
Send part of requested schedule info list
Receive response
…
Retrieve the last part of schedule info list
GetScheduleInfoList (Limit skipped, StartReference := *nextStartReference*)
GetScheduleInfoListResponse (NextStartReference =: *NextStartReference*, ScheduleInfo list =: schedule*InfoListPart*)
Send part of requested schedule info list
Receive response

Procedure:

1. ONVIF client invokes **GetScheduleInfoList** with parameters
   - Limit skipped
   - StartReference skipped
2. The DUT responds with **GetScheduleInfoListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - ScheduleInfo list =: *scheduleInfoCompleteList*
3. Until *nextStartReference* is not null, repeat the following steps:

3.1. ONVIF client invokes **GetScheduleInfoList** with parameters

- Limit skipped
- StartReference := *nextStartReference*

3.2. The DUT responds with **GetScheduleInfoListResponse** message with parameters

- NextStartReference =: *nextStartReference*
- ScheduleInfo list =: *scheduleInfoListPart*

3.3. Set the following:

- *scheduleInfoCompleteList* := *scheduleInfoCompleteList* + *scheduleInfoListPart*

Procedure Result:

PASS –

The DUT passed all assertions.

FAIL –

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

A.2 Get service capabilities

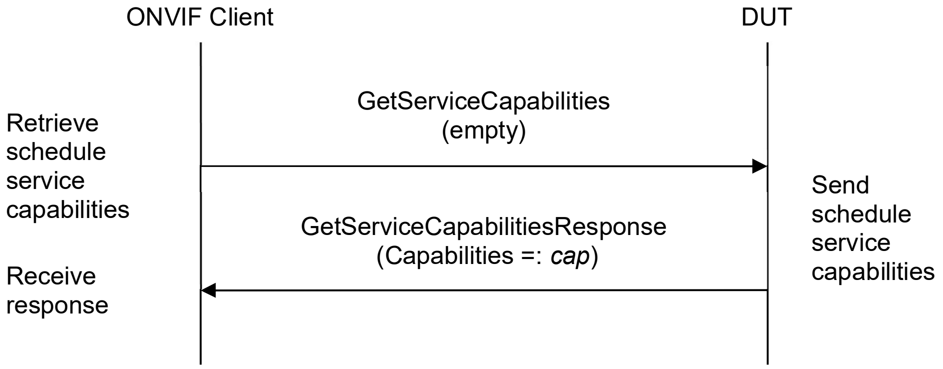
Name: HelperGetServiceCapabilities

Procedure Purpose: Helper procedure to get service capabilities.

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

Input: None

Returns: The service capabilities (*cap*).


ONVIF Client
DUT
GetServiceCapabilities
(empty)
Retrieve schedule service capabilities
Send schedule service capabilities
GetServiceCapabilitiesResponse
(Capabilities =: *cap*)
Receive response

Procedure:

1. ONVIF Client invokes **GetServiceCapabilities**.
2. The DUT responds with a **GetServiceCapabilitiesResponse** message with parameters
    - Capabilities =: *cap*

Procedure Result:

PASS –

The DUT passed all assertions.

FAIL –

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

A.3 Get schedules list

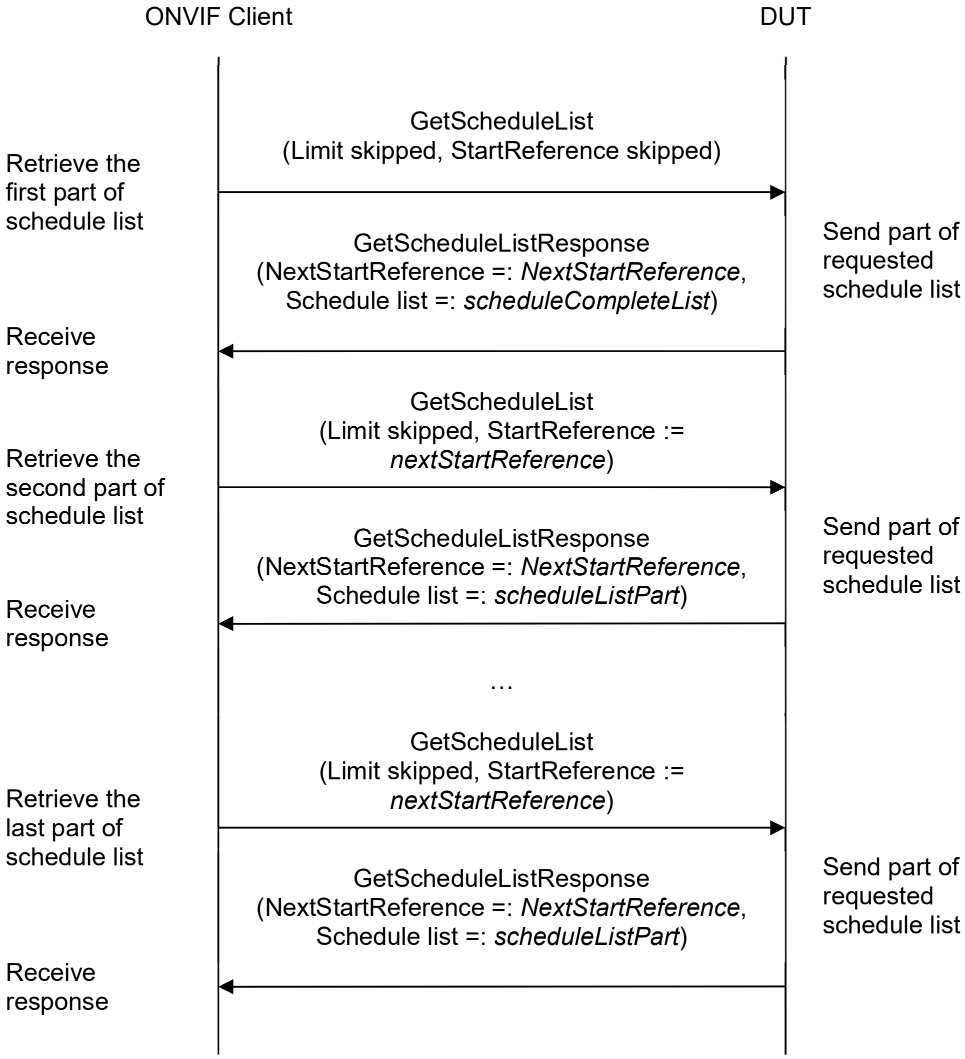
Name: HelperGetScheduleList

Procedure Purpose: Helper procedure to get complete schedules list with.

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

Input: None.

Returns: The complete list of schedules (*scheduleCompleteList*).


ONVIF Client
DUT
GetScheduleList
(Limit skipped, StartReference skipped)
Retrieve the first part of schedule list
GetScheduleListResponse
(NextStartReference =: *NextStartReference*,
Schedule list =: *scheduleCompleteList*)
Send part of requested schedule list
Receive response
GetScheduleList
(Limit skipped, StartReference :=
nextStartReference)
Retrieve the second part of schedule list
GetScheduleListResponse
(NextStartReference =: *NextStartReference*,
Schedule list =: *scheduleListPart*)
Send part of requested schedule list
Receive response
…
GetScheduleList
(Limit skipped, StartReference :=
nextStartReference)
Retrieve the last part of schedule list
GetScheduleListResponse
(NextStartReference =: *NextStartReference*,
Schedule list =: *scheduleListPart*)
Send part of requested schedule list
Receive response

Procedure:

1. ONVIF client invokes **GetScheduleList** with parameters

- Limit skipped
- StartReference skipped

2. The DUT responds with **GetScheduleListResponse** message with parameters
    - NextStartReference =: *nextStartReference*
    - Schedule list =: *scheduleCompleteList*
3. Until *nextStartReference* is not null, repeat the following steps:
    3.1. ONVIF client invokes **GetScheduleList** with parameters
    - Limit skipped
    - StartReference := *nextStartReference*

    3.2. The DUT responds with **GetScheduleListResponse** message with parameters
    - NextStartReference =: *nextStartReference*
    - Schedule list =: *scheduleListPart*

    3.3. Set the following:
    - *scheduleCompleteList* := *scheduleCompleteList* + *scheduleListPart*

Procedure Result:

PASS –

The DUT passed all assertions.

FAIL –

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

A.4 Get special day group information list

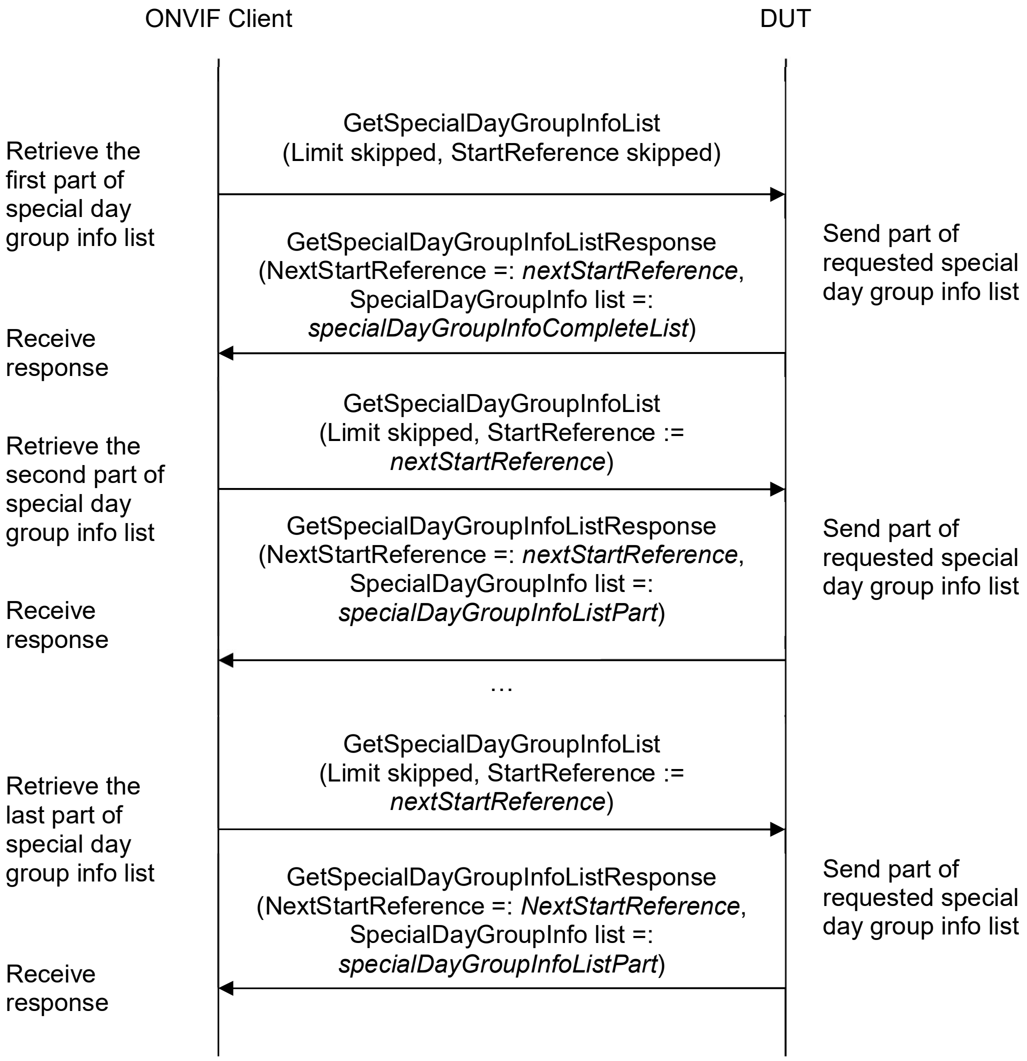
Name: HelperGetSpecialDayGroupInfoList

Procedure Purpose: Helper procedure to get complete SpecialDayGroup information list.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Input: None.

Returns: The complete list of special day group information (*specialDayGroupInfoCompleteList*).


ONVIF Client
DUT
GetSpecialDayGroupInfoList (Limit skipped, StartReference skipped)
Retrieve the first part of special day group info list
GetSpecialDayGroupInfoListResponse (NextStartReference =: nextStartReference, SpecialDayGroupInfo list =: specialDayGroupInfoCompleteList)
Send part of requested special day group info list
Receive response
GetSpecialDayGroupInfoList (Limit skipped, StartReference := nextStartReference)
Retrieve the second part of special day group info list
GetSpecialDayGroupInfoListResponse (NextStartReference =: nextStartReference, SpecialDayGroupInfo list =: specialDayGroupInfoListPart)
Send part of requested special day group info list
Receive response
…
GetSpecialDayGroupInfoList (Limit skipped, StartReference := nextStartReference)
Retrieve the last part of special day group info list
GetSpecialDayGroupInfoListResponse (NextStartReference =: NextStartReference, SpecialDayGroupInfo list =: specialDayGroupInfoListPart)
Send part of requested special day group info list
Receive response

Procedure:

1. ONVIF client invokes **GetSpecialDayGroupInfoList** with parameters

- Limit skipped
- StartReference skipped

2. The DUT responds with **GetSpecialDayGroupInfoListResponse** message with parameters
    - NextStartReference =: *nextStartReference*
    - SpecialDayGroupInfo list =: *specialDayGroupInfoCompleteList*
3. Until *nextStartReference* is not null, repeat the following steps:

3.1. ONVIF client invokes **GetSpecialDayGroupInfoList** with parameters

- Limit skipped
- StartReference := *nextStartReference*

3.2. The DUT responds with **GetSpecialDayGroupInfoListResponse** message with parameters

- NextStartReference =: *nextStartReference*
- SpecialDayGroupInfo list =: *specialDayGroupInfoListPart*

3.3. Set the following:

- *specialDayGroupInfoCompleteList* := *specialDayGroupInfoCompleteList* + *specialDayGroupInfoListPart*

Procedure Result:

PASS –

The DUT passed all assertions.

FAIL –

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

A.5 Get special day group list

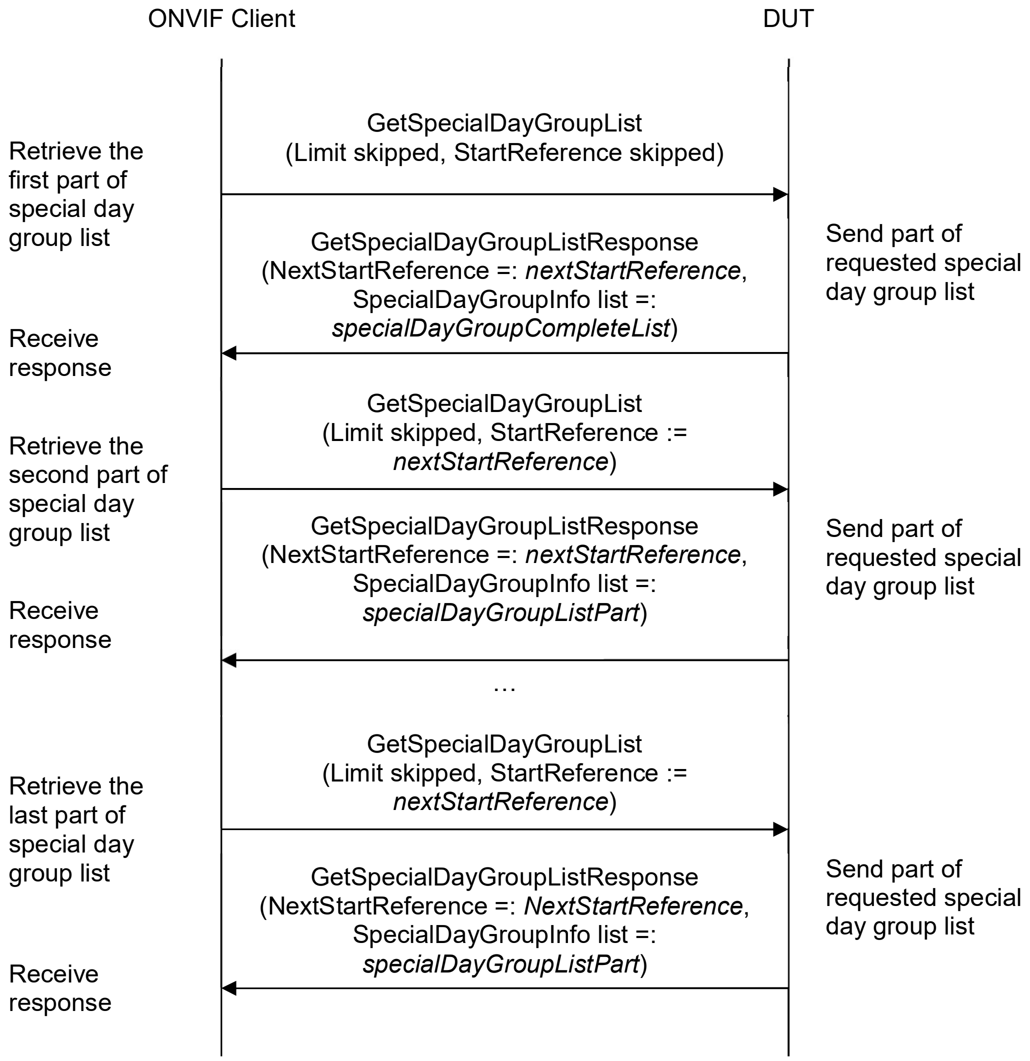
Name: HelperSpecialDayGroupList

Procedure Purpose: Helper procedure to get complete Special Day Group list.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Input: None.

Returns: The complete list of Special Day Group (*specialDayGroupCompleteList*).


ONVIF Client
DUT
GetSpecialDayGroupList (Limit skipped, StartReference skipped)
Retrieve the first part of special day group list
GetSpecialDayGroupListResponse (NextStartReference =: *nextStartReference*, SpecialDayGroupInfo list =: *specialDayGroupCompleteList*)
Send part of requested special day group list
Receive response
GetSpecialDayGroupList (Limit skipped, StartReference := *nextStartReference*)
Retrieve the second part of special day group list
GetSpecialDayGroupListResponse (NextStartReference =: *nextStartReference*, SpecialDayGroupInfo list =: *specialDayGroupListPart*)
Send part of requested special day group list
Receive response
…
GetSpecialDayGroupList (Limit skipped, StartReference := *nextStartReference*)
Retrieve the last part of special day group list
GetSpecialDayGroupListResponse (NextStartReference =: *NextStartReference*, SpecialDayGroupInfo list =: *specialDayGroupListPart*)
Send part of requested special day group list
Receive response

Procedure:

1. ONVIF client invokes **GetSpecialDayGroupList** with parameters

- Limit skipped
- StartReference skipped

2. The DUT responds with **GetSpecialDayGroupListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - SpecialDayGroup list =: *specialDayGroupsCompleteList*
3. Until *nextStartReference* is not null, repeat the following steps:

   3.1. ONVIF client invokes **GetSpecialDayGroupList** with parameters
   - Limit skipped
   - StartReference := *nextStartReference*

   3.2. The DUT responds with **GetSpecialDayGroupListResponse** message with parameters
   - NextStartReference =: *nextStartReference*
   - SpecialDayGroup list =: *specialDayGroupsListPart*

   3.3. Set the following:
   - *specialDayGroupsCompleteList* := *specialDayGroupsCompleteList* + *specialDayGroupsListPart*

Procedure Result:

PASS –

The DUT passed all assertions.

FAIL –

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

A.6 Generate UID value for iCalendar

Name: HelperUIDiCalendarGeneration

Procedure Purpose: Helper procedure to generate Unique Identifier value for UID field in iCalendar.

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

Input: None.

Returns: Unique Identifier value for UID field in iCalendar (*uid*) that is compliant to [RFC2445].

Procedure:

1. ONVIF Client generates Globally Unique Identifier value.

A.7 Generate iCalendar value for Schedule

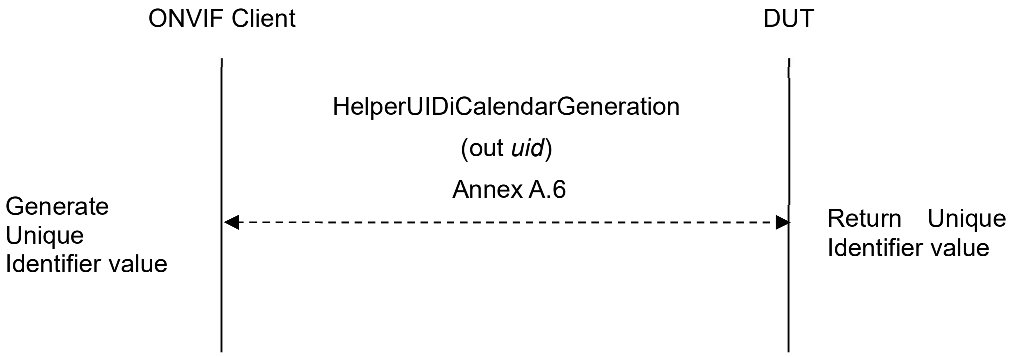
Name: HelperScheduleiCalendarGeneration

Procedure Purpose: Helper procedure to generate iCalendar value for Schedule.Standard field.

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

Input: The Schedule service capabilities (*cap*).

Returns: iCalendar value for Standard field (*scheduleiCalendarValue*) that is compliant to [RFC2445].


ONVIF Client
DUT
HelperUIDiCalendarGeneration
(out *uid*)
Annex A.6
Generate Unique Identifier value
Return Unique Identifier value

Procedure:

1. ONVIF Client generates Unique Identifier value for UID field in iCalendar (out *uid*) by following the procedure mentioned in Annex A.6.

2. If *cap*.ExtendedRecurrenceSupported is equal to true, set the following:

    - *scheduleiCalendarValue* := “BEGIN:VCALENDAR

      BEGIN:VEVENT

SUMMARY:Access from 9 AM to 6 PM

DTSTART:<current year>< current month>< current day>T090000

DTEND:<year of (current day + one week)><month of (current day + one week)><day of (current day + one week)>T180000

RRULE:FREQ=DAILY

UID:*uid*

END:VEVENT

END:VCALENDAR”

3. If *cap*.ExtendedRecurrenceSupported is equal to false, set the following:

    - *scheduleiCalendarValue* := “BEGIN:VCALENDAR

      BEGIN:VEVENT

      SUMMARY:Access on weekdays from 9 AM to 6 PM for employees

      DTSTART:1970<current month><current day>T090000

      DTEND:1970<current month><current day>T180000

      RRULE:FREQ=WEEKLY;BYDAY=MO,TU,WE,TH,FR

      UID:*uid*

      END:VEVENT

      END:VCALENDAR”

A.8 Create special day group

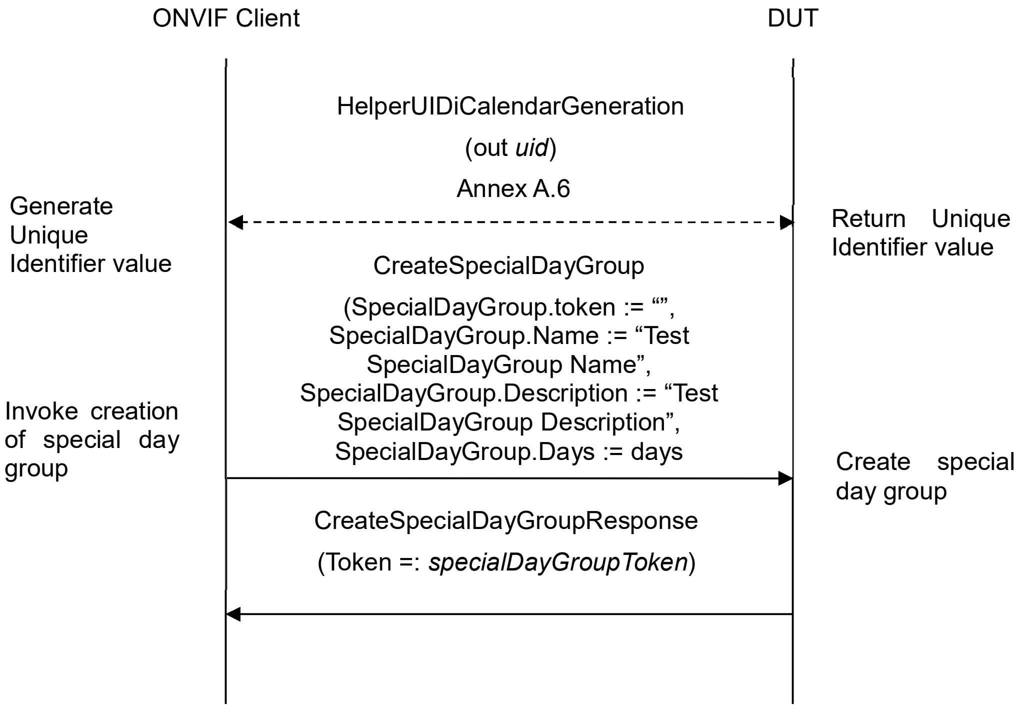
Name: HelperCreateSpecialDayGroup

Procedure Purpose: Helper procedure to create SpecialDayGroup with Days field value that is compliant to [RFC2445].

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported. The DUT shall have enough free storage capacity for one additional SpecialDayGroup.

Input: iCalendar value of the SpecialDayGroup.Days field (*days)* that is compliant to [RFC2445].

Returns: Special day group token (*specialDayGroupToken*), iCalendar value of Days field (*days*) that is compliant to [RFC2445].


ONVIF Client
DUT
HelperUIDiCalendarGeneration
(out *uid*)
Annex A.6
Generate Unique Identifier value
Return Unique Identifier value
CreateSpecialDayGroup
(SpecialDayGroup.token := “”, SpecialDayGroup.Name := “Test SpecialDayGroup Name”, SpecialDayGroup.Description := “Test SpecialDayGroup Description”, SpecialDayGroup.Days := days
Invoke creation of special day group
Create special day group
CreateSpecialDayGroupResponse
(Token =: *specialDayGroupToken*)

Procedure:

1. ONVIF Client generates Unique Identifier value for UID field in iCalendar (out *uid*) by following the procedure mentioned in Annex A.6.

2. If *days* is empty, set the following:

    - *days* := BEGIN:VCALENDAR

      BEGIN:VEVENT

      SUMMARY:Test special days

DTSTART:<current year><current month><current day>T000000

DTEND:<year of the next day><month of the next day><day of the next day>T000000

UID:*uid*

END:VEVENT

END:VCALENDAR

3. ONVIF client invokes **CreateSpecialDayGroup** with parameters
    - SpecialDayGroup.token := “”
    - SpecialDayGroup.Name := “Test SpecialDayGroup Name”
    - SpecialDayGroup.Description := “Test SpecialDayGroup Description”
    - SpecialDayGroup.Days := *days*
4. The DUT responds with **CreateSpecialDayGroupResponse** message with parameters
    - Token =: *specialDayGroupToken*

Procedure Result:

PASS –

The DUT passed all assertions.

FAIL –

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

A.9 Get schedule

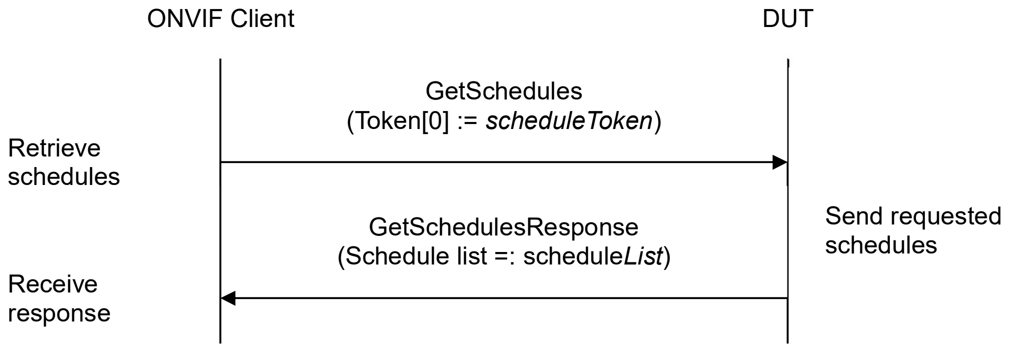
Name: HelperGetSchedule

Procedure Purpose: Helper procedure to get schedule.

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

Input: Schedule Token (*scheduleToken*).

Returns: Schedule List (*scheduleList*).



Procedure:

1. ONVIF client invokes **GetSchedules** with parameters
    - Token[0] := *scheduleToken*
2. The DUT responds with **GetSchedulesResponse** message with parameters
    - Schedule list =: *scheduleList*

PASS –

The DUT passed all assertions.

FAIL –

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

A.10 Get schedule info

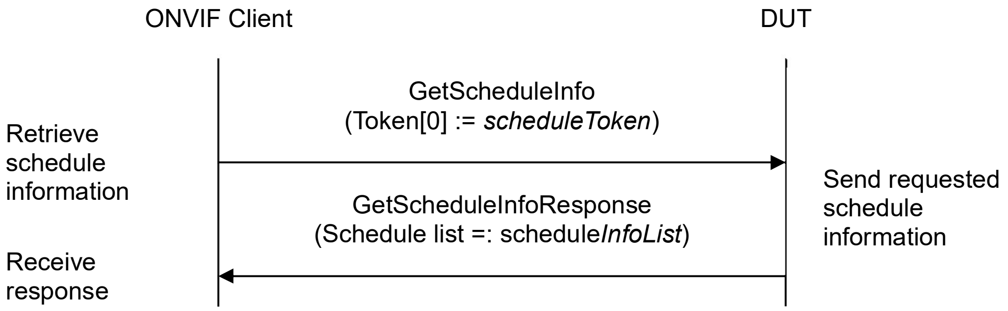
Name: HelperGetScheduleInfo

Procedure Purpose: Helper procedure to get schedule info.

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

Input: Schedule Token (*scheduleToken*).

Returns: Schedule Info List (*scheduleInfoList*).


ONVIF Client
DUT
GetScheduleInfo
(Token[0] := *scheduleToken*)
Retrieve schedule information
Send requested schedule information
GetScheduleInfoResponse
(Schedule list =: schedule*InfoList*)
Receive response

Procedure:

1. ONVIF client invokes **GetScheduleInfo** with parameters
   - Token[0] := *scheduleToken*
2. The DUT responds with **GetScheduleInfoResponse** message with parameters
   - ScheduleInfo =: *scheduleInfoList*

PASS –

The DUT passed all assertions.

FAIL –

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

A.11 Delete schedule

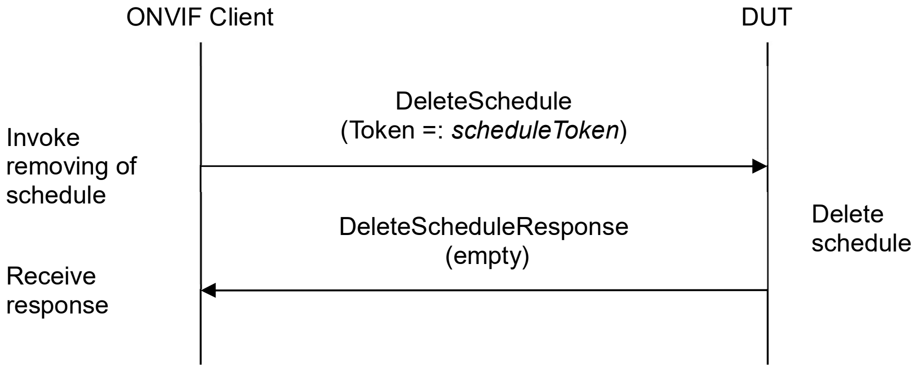
Name: HelperDeleteSchedule

Procedure Purpose: Helper procedure to delete schedule.

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

Input: Schedule Token (*scheduleToken*).

Returns: None.



Procedure:

1. ONVIF client invokes **DeleteSchedule** with parameters
    - Token =: *scheduleToken*
2. The DUT responds with empty **DeleteScheduleResponse** message.

PASS –

The DUT passed all assertions.

FAIL –

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

A.12 Delete special day group

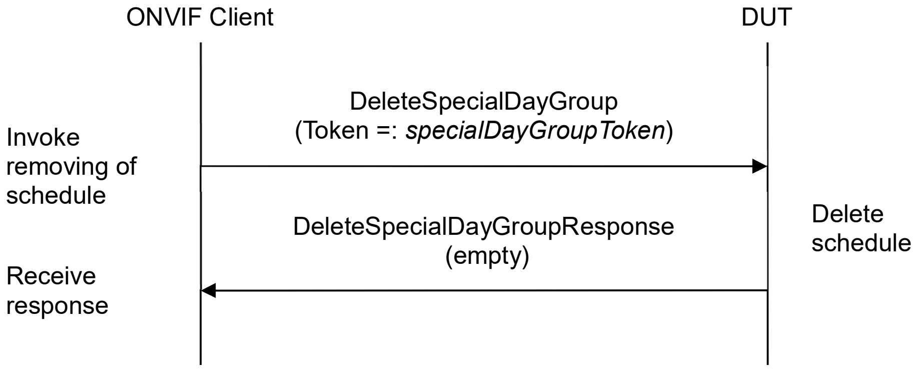
Name: HelperDeleteSpecialDayGroup

Procedure Purpose: Helper procedure to delete SpecialDayGroup.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Input: SpecialDayGroup Token (*specialDayGroupToken*).

Returns: None.


ONVIF Client
DUT
DeleteSpecialDayGroup
(Token =: *specialDayGroupToken*)
Invoke removing of schedule
DeleteSpecialDayGroupResponse
(empty)
Delete schedule
Receive response

Procedure:

1. ONVIF client invokes **DeleteSpecialDayGroup** with parameters
    - Token =: *specialDayGroupToken*
2. The DUT responds with empty **DeleteSpecialDayGroupResponse** message.

PASS –

The DUT passed all assertions.

FAIL –

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

A.13 Create Schedule

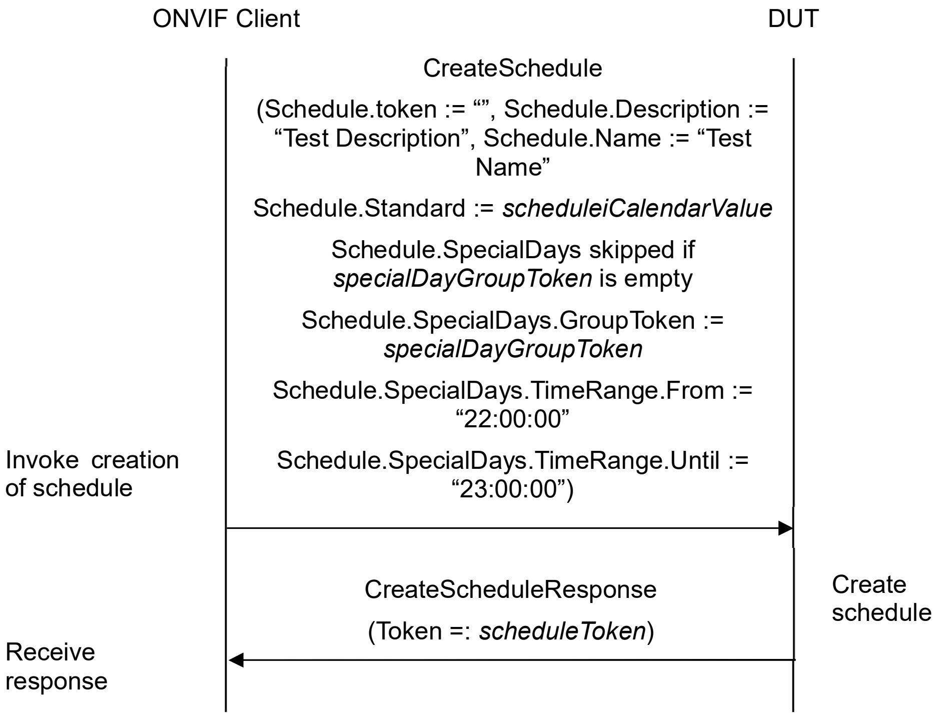
Name: HelperCreateSchedule

Procedure Purpose: Helper procedure to create Schedule with Standard field value that is compliant to [RFC2445].

Pre-requisite: Schedule Service is received from the DUT. The DUT shall have enough free storage capacity for one additional Schedule.

Input: iCalendar value of the Schedule.Standard field (*scheduleiCalendarValue*), SpecialDayGroupToken (*specialDayGroupToken*).

Returns: Schedule token (*scheduleToken*).


ONVIF Client
DUT
CreateSchedule
(Schedule.token := “”, Schedule.Description := “Test Description”, Schedule.Name := “Test Name”
Schedule.Standard := *scheduleiCalendarValue*
Schedule.SpecialDays skipped if *specialDayGroupToken* is empty
Schedule.SpecialDays.GroupToken := *specialDayGroupToken*
Schedule.SpecialDays.TimeRange.From := “22:00:00”
Schedule.SpecialDays.TimeRange.Until := “23:00:00”)
Invoke creation of schedule
CreateScheduleResponse
(Token =: *scheduleToken*)
Create schedule
Receive response

Procedure:

1. ONVIF client invokes **CreateSchedule** with parameters

   - Schedule.token := “”
   - Schedule.Description := “Test Description”
   - Schedule.Name := “Test Name”
   - Schedule.Standard := *scheduleiCalendarValue*
   - Schedule.SpecialDays skipped if *specialDayGroupToken* is empty

- Schedule.SpecialDays.GroupToken := *specialDayGroupToken*
- Schedule.SpecialDays.TimeRange.From := “22:00:00”
- Schedule.SpecialDays.TimeRange.Until := “23:00:00”

2. The DUT responds with **CreateScheduleResponse** message with parameters

   - Token =: *scheduleToken*

Procedure Result:

PASS –

The DUT passed all assertions.

FAIL –

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

A.14 Get special day group

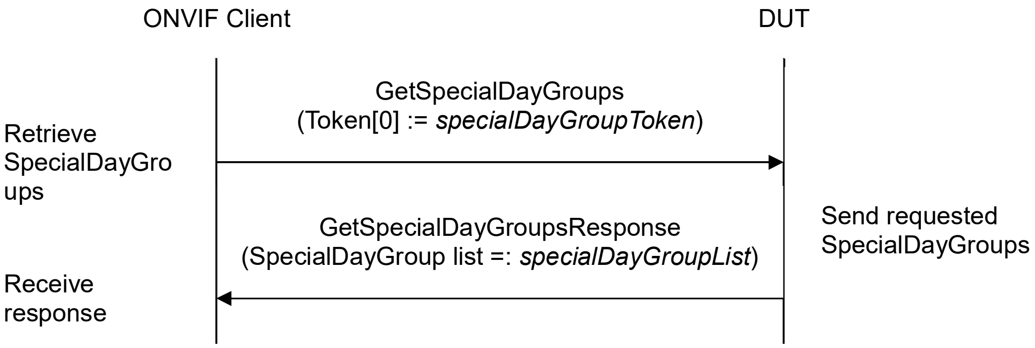
Name: HelperGetSpecialDayGroup

Procedure Purpose: Helper procedure to get SpecialDayGroup.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Input: SpecialDayGroup Token (*specialDayGroupToken*).

Returns: SpecialDayGroup List (*specialDayGroupList*).


ONVIF Client
DUT
GetSpecialDayGroups
(Token[0] := specialDayGroupToken)
Retrieve SpecialDayGroups
GetSpecialDayGroupsResponse
(SpecialDayGroup list =: specialDayGroupList)
Send requested SpecialDayGroups
Receive response

Procedure:

1. ONVIF client invokes **GetSpecialDayGroups** with parameters
   - Token[0] := *specialDayGroupToken*
2. The DUT responds with **GetSpecialDayGroupsResponse** message with parameters
   - SpecialDayGroup list =: *specialDayGroupList*

PASS –

The DUT passed all assertions.

FAIL –

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

A.15 Get special day group info

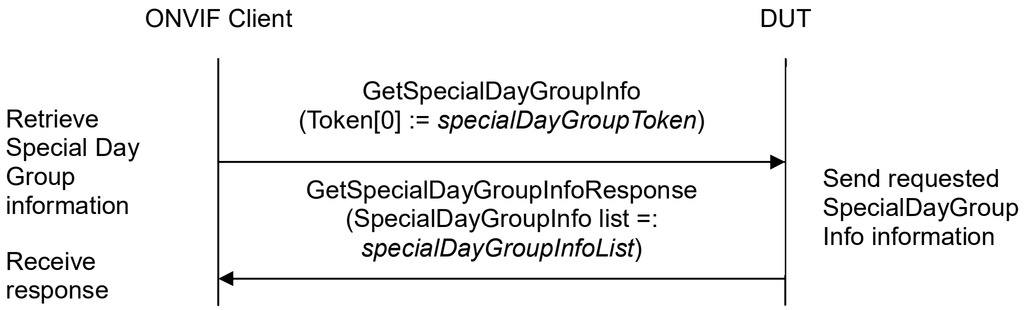
Name: HelperGetSpecialDayGroupInfo

Procedure Purpose: Helper procedure to get special day group info.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Input: SpecialDayGroup Token (*specialDayGroupToken*).

Returns: SpecialDayGroup Info List (*specialDayGroupInfoList*).


ONVIF Client
DUT
Retrieve Special Day Group information
GetSpecialDayGroupInfo (Token[0] := specialDayGroupToken)
Send requested SpecialDayGroup Info information
GetSpecialDayGroupInfoResponse (SpecialDayGroupInfo list =: specialDayGroupInfoList)
Receive response

Procedure:

1. ONVIF client invokes **GetSpecialDayGroupInfo** with parameters
   - Token[0] := *specialDayGroupToken*
2. The DUT responds with **GetSpecialDayGroupInfoResponse** message with parameters
   - SpecialDayGroupInfo =: *specialDayGroupInfoList*

PASS –

The DUT passed all assertions.

FAIL –

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

A.16 Generate iCalenrad value for Special Day Group with required Special Day state

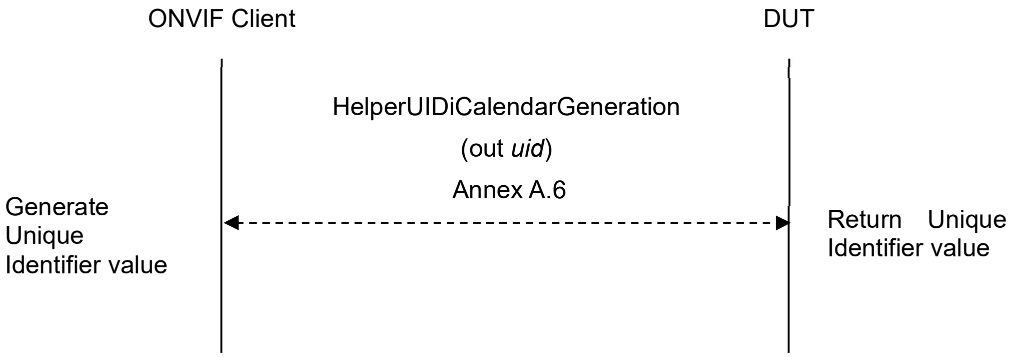
Name: HelperSpecialDayGroupiCalendarGeneration

Procedure Purpose: Helper procedure to generate iCalendar value for Day field in special day group with required special day state.

Pre-requisite: Schedule Service is received from the DUT. Special Days is supported by the DUT as indicated by the Capabilities.SpecialDaysSupported.

Input: special day state (*specialDayState*).

Returns: iCalendar value of the SpecialDayGroup.Days field (*days)* that is compliant to [RFC2445].


ONVIF Client
DUT
HelperUIDiCalendarGeneration
(out *uid*)
Annex A.6
Generate Unique Identifier value
Return Unique Identifier value

Procedure:

1. ONVIF Client generates Unique Identifier value for UID field in iCalendar (out *uid*) by following the procedure mentioned in Annex A.6.

2. If *specialDayState* is equal to *true*, do the following steps:

   2.1. If time to the end of the current day is less than Operation delay time or equals to Operation delay time, the test tool waits for the start of the next day.

   2.2. Set the following:

   - *days* := BEGIN:VCALENDAR

     BEGIN:VEVENT

     SUMMARY:Test special days

     DTSTART:<current year><current month><current day>T000000

     DTEND:<year of the next day><month of the next day><day of the next day>T000000

     UID:*uid*

     END:VEVENT

     END:VCALENDAR

3. If *specialDayState* is equal to *false*, do the following steps:

   3.1. *days* := BEGIN:VCALENDAR

   BEGIN:VEVENT

   SUMMARY:Test special days

   DTSTART:<year of the day that was two days before the current day><month of the day that was two days before the current day><day that was two days before the current day>T000000

   DTEND:<year of the previous day><month of the previous day><day of the previous day>T000000

   UID:*uid*

   END:VEVENT

   END:VCALENDAR

A.17 Generate iCalendar value for Schedule with required number of time periods per day

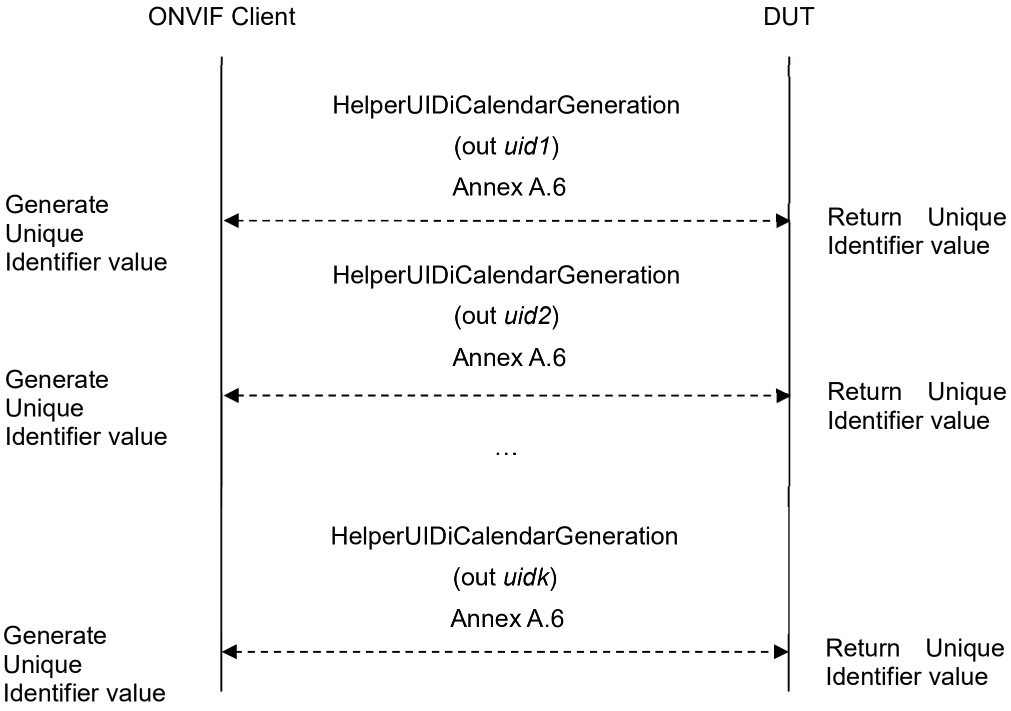
Name: HelperScheduleiCalTimePeriodsGeneration

Procedure Purpose: Helper procedure to generate iCalendar value for Schedule.Standard field with required number of time periods per day.

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

Input: The Schedule service capabilities (*cap*), the number of time periods per day (*numberPeriodsPerDay*).

Returns: iCalendar value for Standard field (*scheduleiCalendarValue*) that is compliant to [RFC2445].


ONVIF Client
DUT
HelperUIDiCalendarGeneration
(out *uid1*)
Annex A.6
Generate Unique Identifier value
Return Unique Identifier value
HelperUIDiCalendarGeneration
(out *uid2*)
Annex A.6
Generate Unique Identifier value
Return Unique Identifier value
…
HelperUIDiCalendarGeneration
(out *uidk*)
Annex A.6
Generate Unique Identifier value
Return Unique Identifier value

Procedure:

1. Set the following:
   - *k* := 1
2. For *k* = 1 to *numberPeriodsPerDay:*

   2.1. ONVIF Client generates Unique Identifier value for UID field in iCalendar (out *uidk*) by following the procedure mentioned in Annex A.6.

   2.2. *k* := *k*+1
3. If *cap*.ExtendedRecurrenceSupported is equal to true, set the following:
   - *scheduleiCalendarValue* := “BEGIN:VCALENDAR

     BEGIN:VEVENT

     SUMMARY:required number of time periods per day

     DTSTART:<current year><current month><current day>T000100

     DTEND:<current year><current month><current day>T000200

     RRULE:FREQ=DAILY

UID:*uid1*

END:VEVENT

BEGIN:VEVENT

SUMMARY:required number of time periods per day

DTSTART:<current year><current month><current day>T000300

DTEND:<current year><current month><current day>T000400

RRULE:FREQ=DAILY

UID:*uid2*

END:VEVENT

…

BEGIN:VEVENT

SUMMARY:required number of time periods per day

DTSTART:<current year><current month><current day>T<time of DTEND in the previous VEVENT + 1min>

DTEND:<current year><current month><current day>T<time of current DTSTART + 1min>

RRULE:FREQ=DAILY

UID:*uidk*

END:VEVENT

END:VCALENDAR"

4. If *cap*.ExtendedRecurrenceSupported is equal to false, set the following:

   - *scheduleiCalendarValue* := "BEGIN:VCALENDAR

     BEGIN:VEVENT

     SUMMARY: required number of time periods per day

     DTSTART:1970<current month><current day>T000100

     DTEND:1970<current month><current day>T000200

     RRULE:FREQ=WEEKLY;BYDAY=MO,TU,WE,TH,FR

     UID:*uid1*

     END:VEVENT

     BEGIN:VEVENT

     SUMMARY: required number of time periods per day

DTSTART:1970<current month><current day>T000300

DTEND:1970<current month><current day>T000400

RRULE:FREQ=WEEKLY;BYDAY=MO,TU,WE,TH,FR

UID:*uid2*

END:VEVENT

…

BEGIN:VEVENT

SUMMARY: required number of time priods per day

DTSTART:1970<current month><current day>T<time of DTEND in the previous VEVENT + 1min>

DTEND:1970<current month><current day>T<time of current DTSTART + 1min>

RRULE:FREQ=WEEKLY;BYDAY=MO,TU,WE,TH,FR

UID:*uidk*

END:VEVENT

END:VCALENDAR"

Note: If *numberPeriodsPerDay* value does not allow creating of *numberPeriodsPerDay* VEVENT items with different periods (value is more than 720), then the number of time periods will be restricted.