ONVIF®

Analytics Engine Device Test Specification

Version 21.06

June 2021

www.onvif.org

REVISION HISTORY

Vers.	Date	Description
17.06	Mar 15, 2017	First Issue.
17.06	Apr 18, 2017	ANALYTICS-1-1-1 GET SUPPORTED RULES (MOTION REGION DETECTOR) added ANALYTICS-1-1-2 GET MOTION REGION DETECTOR RULE OPTIONS added
17.06	May 23, 2017	ANALYTICS-1-1-2 GET MOTION REGION DETECTOR RULE OPTIONS updated ANALYTICS-1-1-3 CREATE MOTION REGION DETECTOR RULE added
17.06	May 24, 2017	ANALYTICS-1-1-4 MODIFY MOTION REGION DETECTOR RULE added ANALYTICS-2-1-1 MOTION REGION DETECTOR EVENT added
17.12	Jul 24, 2017	The following test cases were changed according to #1444: ANALYTICS-1-1-2 GET MOTION REGION DETECTOR RULE OPTIONS ANALYTICS-1-1-4 MODIFY MOTION REGION DETECTOR RULE ANALYTICS-2-1-1 MOTION REGION DETECTOR EVENT
17.12	Jul 24, 2017	The following test cases were changed according to #1445: ANALYTICS-1-1-3 CREATE MOTION REGION DETECTOR RULE ANALYTICS-1-1-4 MODIFY MOTION REGION DETECTOR RULE ANALYTICS-2-1-1 MOTION REGION DETECTOR EVENT
17.12	Sept 18, 2017	The following test cases were added according to #1477: ANALYTICS-3-1-1 Get Services and Get Analytics Service Capabilities Consistency
17.12	Oct 16, 2017	Pre-Requisite of the following test cases were updated according to #1185: GET SUPPORTED RULES (MOTION REGION DETECTOR) GET MOTION REGION DETECTOR RULE OPTIONS CREATE MOTION REGION DETECTOR RULE MODIFY MOTION REGION DETECTOR RULE MOTION REGION DETECTOR EVENT
17.12	Nov 22, 2017	The following test case was updated according to #1184: ANALYTICS-2-1-1 MOTION REGION DETECTOR EVENT
17.12	Nov 28, 2017	The following test cases were updated according to #1398:

		ANALYTICS-1-1-3 CREATE MOTION REGION DETECTOR RULE
		ANALYTICS-1-1-4 MODIFY MOTION REGION DETECTOR RULE
		ANALYTICS-2-1-1 MOTION REGION DETECTOR EVENT
		The following annexes were added according to #1398:
		Annex A.8 Calculate Free Space for Rule
		Annex A.9 Delete Rule with Requested Type
		The following test cases were updated according to #1185:
		ANALYTICS-1-1-3 CREATE MOTION REGION DETECTOR RULE
		ANALYTICS-1-1-4 MODIFY MOTION REGION DETECTOR RULE
		ANALYTICS-2-1-1 MOTION REGION DETECTOR EVENT
18.06	Jan 22, 2018	The following test cases were updated according to #1557:
		ANALYTICS-1-1-3 CREATE MOTION REGION DETECTOR RULE
		ANALYTICS-1-1-4 MODIFY MOTION REGION DETECTOR RULE
		ANALYTICS-2-1-1 MOTION REGION DETECTOR EVENT
		The following annexes was added:
		Annex A.10 Create Motion Region Detector Rule
18.06	Jun 21, 2018	Reformatting document using new template
18.12	Aug 16, 2018	The following test cases were changed according to #1709:
		ANALYTICS-1-1-3 CREATE MOTION REGION DETECTOR RULE
		ANALYTICS-1-1-4 MODIFY MOTION REGION DETECTOR RULE
		ANALYTICS-2-1-1 MOTION REGION DETECTOR EVENT
		Annex Create Motion Region Detector Rule
18.12	Sept 05, 2018	The following test cases were changed according to #1733:
		ANALYTICS-1-1-3 CREATE MOTION REGION DETECTOR RULE (Note about comparison of Armed field changed)
		ANALYTICS-1-1-4 MODIFY MOTION REGION DETECTOR RULE (Note about comparison of Armed field changed)
18.12	Oct 08, 2018	The following test cases were changed according to #1665:
		ANALYTICS-1-1-1 GET SUPPORTED RULES (MOTION REGION DETECTOR) (step 4.3.1 added)
18.12	Dec 21, 2018	Switching Hub description in 'Network Configuration for DUT' section was updated according to #1737
18.12	Dec 27, 2018	Check of @Type in GetRuleOptionsResponse was changed in the following test cases according to #1774:
		GET MOTION REGION DETECTOR RULE OPTIONS
		CREATE MOTION REGION DETECTOR RULE

		MODIFY MOTION REGION DETECTOR RULE
		MOTION REGION DETECTOR EVENT
19.12	Apr 26, 2019	The following was changed according to #1833:
		Scope (Analytics Engine section was replaced with the following sections: Motion Region Detector, Events, Capabilities)
		Scope (Analytics Modules section was added)
		Test Policy (Analytics Modules section was added)
		Analytics Engine (Analytics Modules section was added)
		ANALYTICS-4-1-1 Get Supported Analytics Modules (added)
		ANALYTICS-4-1-2 Get Analytics Module Options (added)
		ANALYTICS-4-1-3 Get Analytics Modules (added)
		A.11 Topic Format Verification (added)
		A.12 Valid Topic Format (added)
19.12	Oct 18, 2019	The following was changed according to #1831:
		Introduction (Analytics Modules section was updated)
		Test Policy (Analytics Modules section was updated)
		ANALYTICS-4-1-4 Get Supported Metadata (added)
		ANALYTICS-3-1-1 Get Services and Get Analytics Service Capabilities Consistency (updated with adding of Note)
		ANALYTICS-3-1-2 Analytics Service Capabilities (added)
		Annex A.13 Get Installed Analytics Modules (added)
		Annex A.14 Get Supported Metadata (added)
19.12	Oct 23, 2019	The following was changed according to #1837:
		ANALYTICS-4-1-5 Object Class Descriptor (added)
20.06	Jan 16, 2020	The following was changed according to #1841:
		ANALYTICS-5-1-2 Vehicle Information Descriptor (added)
		The following was changed according to #1845:
		ANALYTICS-5-1-3 Geo Location Metadata (added)
20.06	Jan 17, 2020	The following was changed according to #1953:
		ANALYTICS-4-1-5 Create Analytics Modules (added)
		ANALYTICS-4-1-6 Delete Analytics Modules (added)
		ANALYTICS-4-1-7 Modify Analytics Modules (added)
20.06	Feb 10, 2020	The following was changed according to #1820:
		ANALYTICS-5-1-4 Human Face Descriptor (added)

		The following was changed according to #1843:
		ANALYTICS-5-1-5 License Plate Information Descriptor (added)
		The following was changed according to #1839:
		ANALYTICS-5-1-6 Human Body Descriptor (added)
20.06	Mar 17, 2020	The following was changed according to #1835:
		ANALYTICS-5-1-7 Image Data (added)
20.06	Apr 16, 2020	The following test cases were updated according to #2037:
		ANALYTICS-5-1-4 Human Face Descriptor
		ANALYTICS-5-1-5 License Plate Information Descriptor
		ANALYTICS-5-1-6 Human Body Descriptor
20.06	May 13, 2020	The following was changed according to #1999:
		ANALYTICS-2-1-1 MOTION REGION DETECTOR EVENT (step 21, 26 updated)
20.06	Jun 15, 2020	The following was changed according to #2018:
		ANALYTICS-4-1-4 GET SUPPORTED METADATA (step 7.2, 8.8 added)
20.12	Aug 05, 2020	The following was added according to #2015:
		Introduction\Rules
		Test Policy\Rules
		ANALYTICS-6-1-1 GET SUPPORTED RULES
		ANALYTICS-6-1-2 GET RULE OPTIONS
		ANALYTICS-6-1-3 GET RULES
		ANALYTICS-6-1-4 CREATE RULES
		ANALYTICS-6-1-5 DELETE RULES
		ANALYTICS-6-1-6 MODIFY RULES
		Annex A.27 Device Configuration For Create Rule
		Annex A.28 Get List of Non-Fixed Analytics Configurations With Supporting of Rules
		Annex A.29 Get Supported Rules
		Annex A.30 Prepare Free Space for Rule
		Annex A.31 Delete Rule
		Annex A.32 Create Rule
		Annex A.33 Select Existing Rule
20.12	Aug 24, 2020	The following test case was updated according to #2065:

		ANALYTICS-4-1-2 GET ANALYTICS MODULES OPTIONS (step with check that for each rule description there is corresponding option item was removed)
20.12	Aug 28, 2020	The following was added according to #1847: Introduction\Face Recognition Rule Test Policy\Face Recognition Rule ANALYTICS-7-1-1 GET SUPPORTED RULES (FACE RECOGNITION) ANALYTICS-7-1-2 GET FACE RECOGNITION RULE OPTIONS ANALYTICS-7-1-3 CREATE FACE RECOGNITION RULE ANALYTICS-7-1-4 MODIFY FACE RECOGNITION RULE ANALYTICS-2-1-2 FACE RECOGNITION EVENT A.34 Device Configuration For Create Specific Rule A.35 Select Value For Rule String Parameter Type
20.12	Oct 05, 2020	The following was changed according to #2102: ANALYTICS-4-1-3 GET ANALYTICS MODULES (4.6.2.4 and 4.6.2.5 steps were removed) ANALYTICS-6-1-3 GET RULES (4.6.2.4 and 4.6.2.5 steps were removed)
20.12	Nov 09, 2020	The following was changed according to #2020: ANALYTICS-6-1-1 GET SUPPORTED RULES (step 4.4.5 added) ANALYTICS-4-1-1 GET SUPPORTED ANALYTICS MODULES (step 4.3.5 added)
20.12	Dec 06, 2020	The following was added according to #2065: ANALYTICS-4-1-8 MODIFY ANALYTICS MODULES (PARAMETERS WITHOUT OPTIONS) A.37 Supported Parameter Types A.38 Select Analytics Module Description with Parameters without Options A.39 Get Analytics Module Options
20.12	Dec 06, 2020	The following was changed according to #2023: A.18 Prepare Free Space for Analytics Module (steps 5 - 7 added) A.10 Create Motion Region Detector Rule (steps 1, 2, 3, 6 removed) Steps with device configuration for adding a rule was replased with Annex A.34 Device Configuration For Create Specific Rule ANALYTICS-1-1-3 CREATE MOTION REGION DETECTOR RULE ANALYTICS-1-1-4 MODIFY MOTION REGION DETECTOR RULE ANALYTICS-2-1-1 MOTION REGION DETECTOR EVENT

21.06	Feb 01, 2021	The following was changed according to #2065: ANALYTICS-4-1-7 MODIFY ANALYTICS MODULES was updated and renamed to ANALYTICS-4-1-7 MODIFY ANALYTICS MODULES (PARAMETERS WITH OPTIONS) A.40 Device Configuration For Modify Analytics Module (updated)
21.06	Feb 07, 2021	Profile T was added as Pre-Requisite according to #2128: ANALYTICS-1-1-3 CREATE MOTION REGION DETECTOR RULE ANALYTICS-1-1-4 MODIFY MOTION REGION DETECTOR RULE ANALYTICS-2-1-1 MOTION REGION DETECTOR EVENT
21.06	Mar 07, 2021	The following was changed according to #2070: ANALYTICS-6-1-6 MODIFY ANALYTICS MODULES was updated and renamed to ANALYTICS-6-1-6 MODIFY RULES (PARAMETERS WITH OPTIONS) ANALYTICS-6-1-7 MODIFY RULES (PARAMETERS WITHOUT OPTIONS) added A.41 Select Rule Description To Modify A.42 Device Configuration For Modify Rule (added)
21.06	Mar 29, 2021	The following test cases were added according to #1849: ANALYTICS-8-1-1 GET SUPPORTED RULES (LICENSE PLATE RECOGNITION) ANALYTICS-8-1-2 GET LICENSE PLATE RECOGNITION RULE OPTIONS ANALYTICS-2-1-3 LICENSE PLATE RECOGNITION EVENT
21.06	Mar 29, 2021	The following were updated according to #2023: A.27 Device Configuration For Create Rule (step 7 added) A.10 Create Motion Region Detector Rule (steps 1, 2, 3, 6 removed) Steps with device configuration for adding a rule was replased with Annex A.34 Device Configuration For Create Specific Rule ANALYTICS-1-1-3 CREATE MOTION REGION DETECTOR RULE ANALYTICS-1-1-4 MODIFY MOTION REGION DETECTOR RULE ANALYTICS-2-1-1 MOTION REGION DETECTOR EVENT
21.06	Apr 22, 2021	The following test cases and annexes were added according to #1853: ANALYTICS-2-1-4 LINE CROSSING COUNTER EVENT ANALYTICS-9-1-1 GET SUPPORTED RULES (LINE CROSSING COUNTER) ANALYTICS-9-1-2 GET LINE CROSSING COUNTER RULE OPTIONS A.43 Profile Configuration With Specific Rule Type

		A.44 Select Profile With Specific Rule Type
		A.45 Select Existing Rule With Specific Type
		A.46 Rule Engine Topic Format Verification
21.06	Apr 30, 2021	The following test cases were updated according to #2090:
		ANALYTICS-4-1-7 MODIFY ANALYTICS MODULES (PARAMETERS WITH OPTIONS) (step 6, 6.1 updated)
		ANALYTICS-4-1-8 MODIFY ANALYTICS MODULES (PARAMETERS WITHOUT OPTIONS) (step 6, 6.1 updated)
		ANALYTICS-6-1-6 MODIFY RULES (PARAMETERS WITH OPTIONS) (step 6, 6.1 updated)
		ANALYTICS-6-1-7 MODIFY RULES (PARAMETERS WITHOUT OPTIONS) (step 6, 6.1 updated)
21.06	May 13, 2021	The following test cases were updated according to #2200:
		ANALYTICS-4-1-4 GET SUPPORTED METADATA (step 6 added)
21.06	May 24, 2021	The following test cases were updated according to #2202:
		ANALYTICS-5-1-1 OBJECT CLASSIFICATION METADATA (steps 4.1.1 and 4.1.2 changed)
21.06	May 26, 2021	Profile M was added as Pre-Requisite according to #2193:
		ANALYTICS-2-1-4 LINE CROSSING COUNTER EVENT
		ANALYTICS-4-1-5 CREATE ANALYTICS MODULES
		ANALYTICS-4-1-6 DELETE ANALYTICS MODULES
		ANALYTICS-4-1-7 MODIFY ANALYTICS MODULES (PARAMETERS WITH OPTIONS)
		ANALYTICS-4-1-8 MODIFY ANALYTICS MODULES (PARAMETERS WITHOUT OPTIONS)
		ANALYTICS-6-1-4 CREATE RULES
		ANALYTICS-6-1-5 DELETE RULES
		ANALYTICS-6-1-6 MODIFY RULES (PARAMETERS WITH OPTIONS)
		ANALYTICS-6-1-7 MODIFY RULES (PARAMETERS WITHOUT OPTIONS)

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

1.1 Scope

This ONVIF Analytics Engine 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 to provide test cases to test individual requirements of ONVIF devices according to ONVIF Analytics service(s) which is defined in [ONVIF Network Interface Specs].

The principal intended purposes are:

• Provide self-assessment tool for implementations.

• Provide comprehensive test suite coverage for [ONVIF Network Interface Specs].

This specification **does not** address the following:

• Product use cases and non-functional (performance and regression) testing.

• SOAP Implementation Interoperability test i.e. Web Service Interoperability Basic Profile version 2.0 (WS-I BP 2.0).

• Network protocol implementation Conformance test for HTTP, HTTPS, RTP and RTSP protocol.

• Poor streaming performance test (audio/video distortions, missing audio/video frames, incorrect lib synchronization etc.).

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 Analytics Engine Test Specification covers Analytics 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.

1.2 Motion Region Detector

Motion Region Detector test cases cover verification of Rule interface for Motion Region Detector feature as mentioned in [ONVIF Analytics Spec]. It means that the following commands are covered by these test cases:

- GetSupportedRules (for Motion Region Detector Rule scope only);

- CreateRules (for Motion Region Detector Rule scope only);

- ModifyRules (for Motion Region Detector Rule scope only);

- DeleteRules (for Motion Region Detector Rule scope only);

- GetRules (for Motion Region Detector Rule scope only).

1.3 Events

Events test cases cover verification of property events defined in [ONVIF Analytics Spec]. Currently the following events are covered by these test cases:

- tns1:RuleEngine/MotionRegionDetector/Motion.

1.4 Capabilities

Capabilities test cases cover verification to get Analytics Service capabilities. It means that the following commands are covered by these test cases:

- GetServices (Analytics Service);

- GetServiceCapabilities.

1.5 Analytics Modules

Analytics Modules test cases cover verification of analytics modules configuration feature as mentioned in [ONVIF Analytics Spec]. It means that the following commands are covered by these test cases:

- GetSupportedAnalyticsModules;

- GetAnalyticsModules;

- GetAnalyticsModuleOptions;

- GetSupportedMetadata;

- CreateAnalyticsModules;

- DeleteAnalyticsModules;

- ModifyAnalyticsModules.

1.6 Scene Elements

Scene Elements test cases cover verification of scene elements in supported metadata as mentioned in [ONVIF Analytics Spec]. It means that the following features are covered by these test cases:

- Object Classification descriptor;

- Vehicle information descriptor;

- Human Face descriptor;

- Human Body descriptor;

- License plate information descriptor;

- Geo location metadata;

- GetSupportedMetadata;

- Image Data.

1.7 Rules

Rules test cases cover verification of rules configuration feature as mentioned in [ONVIF Analytics Spec]. It means that the following commands are covered by these test cases:

- GetSupportedRules;

- GetRules;

- GetRuleOptions;

- CreateRules;

- DeleteRules;

- ModifyRules.

1.8 Face Recognition Rule

Face Recognition Rule test cases cover verification of Rule interface for Face Recognition feature as mentioned in [ONVIF Analytics Spec]. It means that the following commands are covered by these test cases:

- GetSupportedRules (for Face Recognition Rule scope only);

- CreateRules (for Face Recognition Rule scope only);

- ModifyRules (for Face Recognition Rule scope only);

- DeleteRules (for Face Recognition Rule scope only);

- GetRules (for Face Recognition Rule scope only).

2 Normative references

- [ONVIF Conformance] ONVIF Conformance Process Specification:

  https://www.onvif.org/profiles/conformance/

- [ONVIF Profile Policy] ONVIF Profile Policy:

  https://www.onvif.org/profiles/

- [ONVIF Network Interface Specs] ONVIF Network Interface Specification documents:

  https://www.onvif.org/profiles/specifications/

- [ONVIF Core Specs] ONVIF Core Specification:

  https://www.onvif.org/profiles/specifications/

- [ONVIF Media2 Spec] ONVIF Media 2 Specification:

  https://www.onvif.org/profiles/specifications/

- [ONVIF Analytics Spec] ONVIF Analytics Specification:

  https://www.onvif.org/profiles/specifications/

- [ONVIF Base Test] ONVIF Base Device Test Specification:

  https://www.onvif.org/profiles/conformance/device-test/

- [ISO/IEC Directives, Part 2] ISO/IEC Directives, Part 2, Annex H:

  http://www.iso.org/directives

- [ISO 16484-5] ISO 16484-5:2014-09 Annex P:

  https://www.iso.org/obp/ui/#!iso:std:63753:en

- [SOAP 1.2, Part 1] W3C SOAP 1.2, Part 1, Messaging Framework:

  http://www.w3.org/TR/soap12-part1/

- [XML-Schema, Part 1] W3C XML Schema Part 1: Structures Second Edition:

  http://www.w3.org/TR/xmlschema-1/

- [XML-Schema, Part 2] W3C XML Schema Part 2: Datatypes Second Edition:

  http://www.w3.org/TR/xmlschema-2/

- [WS-Security] "Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)", OASIS Standard, February 2006.:

  http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

3 Terms and Definitions

3.1 Conventions

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

3.2 Definitions

This section describes terms and definitions used in this document.

Profile	See ONVIF Profile Policy.
ONVIF Device	Computer appliance or software program that exposes one or multiple ONVIF Web Services.
ONVIF Client	Computer appliance or software program that uses ONVIF Web Services.
Media Profile	A media profile maps a video and/or audio source to a video and/or an audio encoder, PTZ and analytics configurations.
SOAP	SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols.
Device Test Tool	ONVIF Device Test Tool that tests ONVIF Device implementation towards the ONVIF Test Specification set.
Video Analytics	Algorithms used to evaluate video data for meaning of content.
Audio Analytics	Algorithms used to evaluate audio data for meaning of content.

3.3 Abbreviations

This section describes abbreviations used in this document.

HTTP Hyper Text Transport Protocol.

WSDL Web Services Description Language.

XML eXtensible Markup Language.

PTZ Pan/Tilt/Zoom.

4 Test Overview

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

4.1 Test Setup

4.1.1 Network Configuration for DUT

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

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

Figure 4.1. Test Configuration for DUT
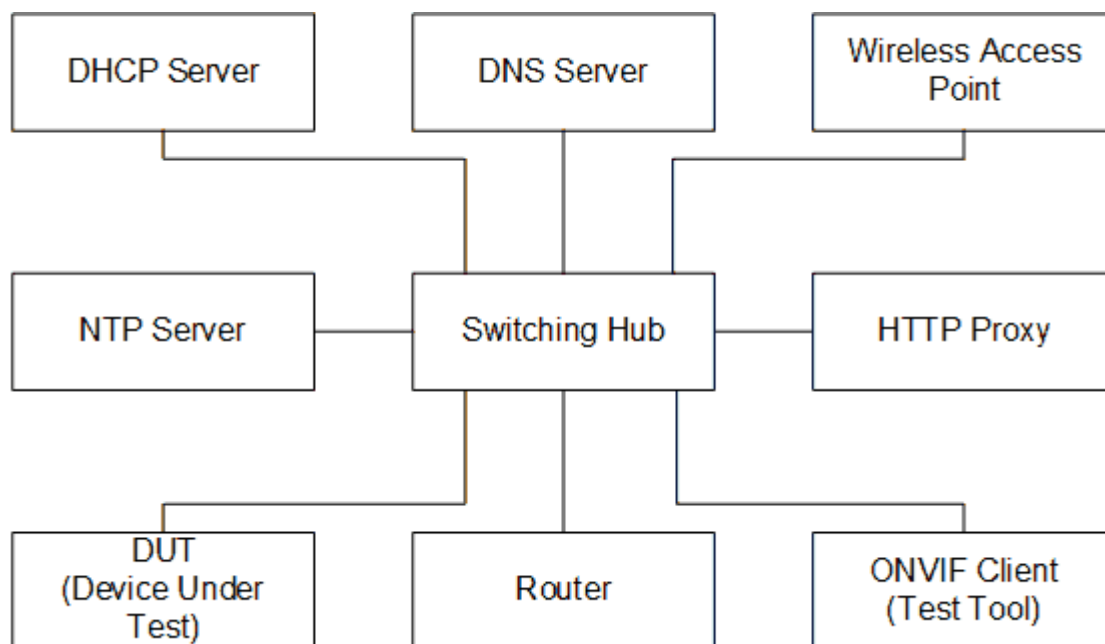
DUT: ONVIF device to be tested. Hereafter, this is referred to as DUT (Device Under Test).

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

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

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

www.onvif.org

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

DHCP Server: provides IPv4 Address to the connected devices.

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

Switching Hub: provides network connectivity among all the test equipments in the test environment. All devices should be connected to the Switching Hub. When running multiple test instances in parallel on the same network, the Switching Hub should be configured to use filtering in order to avoid multicast traffic being flooded to all ports, because this may affect test stability.

Router: provides router advertisements for IPv6 configuration.

4.2 Prerequisites

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

1. The DUT shall be configured with an IPv4 address.

2. The DUT shall be IP reachable [in the test configuration].

3. The DUT shall be able to be discovered by the Test Tool.

4. The DUT shall be configured with the time i.e. manual configuration of UTC time and if NTP is supported by the DUT, then NTP time shall be synchronized with NTP Server.

5. The DUT time and Test tool time shall be synchronized with each other either manually or by common NTP server

4.3 Test Policy

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

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

4.3.1 Motion Region Detector

The test policies specific to the test case execution of Motion Region Detector functional block:

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

• DUT shall give the Media2 Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.

• DUT shall provide Motion Region Detector rule, if DUT supports this rule. Otherwise, these test cases will be skipped.

- DUT shall support the following commands:

    - GetServiceCapabilities

- If DUT returns RuleSupport capability as supported, then DUT shall support commands listed below. Otherwise, these test cases will be skipped.

    - GetServiceCapabilities

    - GetSupportedRules

    - GetRules

    - CreateRules

    - ModifyRules

    - DeleteRules

- If DUT returns RuleOptionsSupported capability as supported, then DUT shall support GetRuleOptions command. Otherwise, the following test cases will be skipped:

    - GET MOTION REGION DETECTOR RULE OPTIONS

    - MODIFY MOTION REGION DETECTOR RULE

Please, refer to Section 5.1 for Motion Region Detector Test Cases.

4.3.2 Events

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

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

- DUT shall give the Media2 Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.

- DUT shall provide Motion Region Detector rule, if DUT supports this rule. Otherwise, these test cases will be skipped.

- DUT shall provide tns1:RuleEngine/MotionRegionDetector/Motion notification topic and Initialized event, if DUT supports Motion Region Detector rule. Otherwise, these test cases will be skipped.

- DUT shall support the following commands:

- GetServiceCapabilities

- GetEventProperties

- CreatePullPointSubscription

- PullMessages

- Unsubscribe

- If DUT returns RuleSupport capability as supported, then DUT shall support commands listed below. Otherwise, these test cases will be skipped.

  - GetServiceCapabilities

  - GetSupportedRules

  - CreateRules

  - DeleteRules

- If DUT returns RuleOptionsSupported capability as supported, then DUT shall support GetRuleOptions command. Otherwise, these test cases will be skipped.

Please, refer to Section 5.2 for Motion Region Detector Test Cases.

4.3.3 Capabilities

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

- DUT shall give the Analytics 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 5.3 for Capabilities Test Cases.

4.3.4 Analytics Modules

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

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

- DUT shall support the following commands:

  - GetSupportedAnalyticsModules

  - GetAnalyticsModules

  - GetAnalyticsModuleOptions

  - GetSupportedMetadata

  - CreateAnalyticsModules

  - DeleteAnalyticsModules

  - ModifyAnalyticsModules

- Additionally, DUT shall support the following commands which will be used as supplementary during the testing:

  - GetServices

- DUT shall return all supported analytics modules in GetSupportedAnalyticsModulesResponse response for the video analytics configuration specified in GetSupportedAnalyticsModules request.

- If DUT supports Supported Metadata as indicated by the Capabilities.SupportedMetadata, then DUT shall support the following commands

  - GetSupportedMetadata

- DUT shall indicate maximum number of analytics modules through the maxInstances attribute in GetSupportedAnalyticsModulesResponse.

- DUT shall return unique names of the parameters for each supported analytics modules in GetSupportedAnalyticsModulesResponse response.

- DUT shall return unique names of the messages for each supported analytics modules in GetSupportedAnalyticsModulesResponse response.

- DUT shall return valid parent topic value for each supported analytics modules in GetSupportedAnalyticsModulesResponse response.

- DUT shall return all configured analytics modules in GetAnalyticsModulesResponse response for the video analytics configuration specified in GetAnalyticsModules request.

- DUT shall return all analytics modules, which marked as fixed in GetSupportedAnalyticsModulesResponse response, in GetAnalyticsModulesResponse response for the video analytics configuration specified in GetAnalyticsModules request.

- If DUT shall return configured analytics modules in GetAnalyticsModulesResponse response whith the structure defined in GetSupportedAnalyticsModulesResponse response for the corresponding analytics module type.

- If DUT supports receiving of analitics module options as indicated by AnalyticsModuleOptionsSupported capability:

  - DUT shall return options for all parameters of all supported analytics modules in GetAnalyticsModuleOptionsResponse response for the video analytics configuration specified in GetAnalyticsModuleOptions request, if Type is skipped.

  - DUT shall return options for all parameters in GetAnalyticsModuleOptionsResponse response for the video analytics configuration and supported analytics module specified in GetAnalyticsModuleOptions request.

  - DUT shall not return RuleType for any option in GetAnalyticsModuleOptionsResponse response.

- The following tests are performed:

  - Receiving of supported analytics modules for each video analytics configuration.

  - Receiving of all configured analytics modules for each video analytics configuration.

  - Verifying of consistency between configured analytics modules and supported analytics modules description.

  - If DUT supports receiving of analitics module options as indicated by AnalyticsModuleOptionsSupported capability:

    - Receiving of all analytics module options for each video analytics configuration.

    - Receiving of analytics module options for specified analytics module type for each video analytics configuration.

    - Verifying of consistency between analytics module options and supported analytics modules description.

- If DUT supports receiving of supported metadata as indicated by SupportedMetadata capability:

    - Receiving of supported metadata for each installed module type.

Please refer to Section 5.4 for Analytics Modules Test Cases.

4.3.5 Scene Elements

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

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

- If DUT supports Supported Metadata as indicated by the Capabilities.SupportedMetadata, then DUT shall support the following commands

    - GetSupportedMetadata

- The following tests are performed

    - Getting supported metadata with GetSupportedMetadata command;

    - Check that sum of Likelihood of Object is not greater than 1;

    - Check Vehicle Information Descriptor in Supported Metadata;

    - Check Geo Location Metadata in Supported Metadata;

    - Check Human Face Descriptor in Supported Metadata;

    - Check Human Body Descriptor in Supported Metadata;

    - Check License Plate Information Descriptor in Supported Metadata;

    - Check Image Data in Supported Metadata;

Please refer to Section 5.5 for Scene Elements Test Cases.

4.3.6 Rules

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

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

- DUT shall give the Media2 Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.

- DUT shall support Rule Engine as defined by RuleSupport capability. Otherwise, these test cases will be skipped.

- DUT shall support Rule Options as defined by RuleOptionsSupported capability. Otherwise, th following test cases will be skipped:

  - ANALYTICS-6-1-2 (see Section 5.6.2)

- DUT shall supports Profile M. Otherwise, th following test cases will be skipped:

  - ANALYTICS-6-1-4 (see Section 5.6.4)

  - ANALYTICS-6-1-5 (see Section 5.6.5)

  - Section 5.6.6, "MODIFY RULES (PARAMETERS WITH OPTIONS)"

- DUT shall support the following commands:

  - GetSupportedRules

  - GetRules

  - GetRuleOptions

  - CreateRules

  - DeleteRules

  - ModifyRules

- DUT shall support the following commands if DUT supports Rule Options:

  - GetSupportedRules

  - GetRuleOptions

- Additionally, DUT shall support the following commands which will be used as supplementary during the testing:

  - GetServices

  - GetAnalyticsConfigurations

- DUT shall return all supported rules in GetSupportedRulesResponse response for the video analytics configuration specified in GetSupportedRules request.

- DUT shall indicate maximum number of rules with the same type through the maxInstances attribute in GetSupportedRulesResponse.

- DUT shall indicate maximum number of rules through the Limit attribute in GetSupportedRulesResponse if there is such limitation.

- DUT shall return unique names of the parameters for each supported rules in GetSupportedRulesResponse response.

- DUT shall return unique names of the messages for each supported rules in GetSupportedRulesResponse response.

- DUT shall return valid parent topic value for each supported rules in GetSupportedRulesResponse response.

- DUT shall return all configured rules in GetRulesResponse response for the video analytics configuration specified in GetRules request.

- DUT shall return all rules, which marked as fixed in GetSupportedRulesResponse response, in GetRulesResponse response for the video analytics configuration specified in GetRules request.

- DUT shall return configured rules in GetRulesResponse response whith the structure defined in GetSupportedRulesResponse response for the corresponding rule type.

- If DUT supports receiving of rule options as indicated by RuleOptionsSupported capability:

  - DUT shall return options for all parameters, for which options is defined, of all supported rules in GetRuleOptionsResponse response for the video analytics configuration specified in GetRuleOptions request, if Type is skipped.

  - DUT shall return options for all parameters, for which options is defined, in GetRuleOptionsResponse response for the video analytics configuration and supported rule specified in GetRuleOptions request.

  - DUT shall not return AnalyticsModule for any option in GetRuleOptionsResponse response.

- The DUT shall support creating of a rule if at least one non-fixed rule is supported.

- The DUT shall support deleting of a rule if at least one non-fixed rule is supported.

- The DUT shall support modifying of a rule.

- If DUT supports Profile M:

www.onvif.org

- DUT shall support creating of a rule with default parameters values, if at least one non-fixed rule is supported, if empty Parameters list was specified in CreateRules request.

- DUT shall support modifying of a rule with default parameters values, if empty Parameters list was specified in ModifyRules request.

- The following tests are performed:

  - Receiving of supported rules for each video analytics configuration.

  - Receiving of all configured rules for each video analytics configuration.

  - Verifying of consistency between configured rules and supported rules description.

  - If DUT supports receiving of rule options as indicated by RuleOptionsSupported capability:

    - Receiving of all rule options for each video analytics configuration.

    - Receiving of rule options for specified rule type for each video analytics configuration.

    - Verifying of consistency between rule options and supported rules description.

  - If DUT supports Profile M:

    - Creating a rule with CreateRules command with empty Parameters list.

    - Modifying a rule with ModifyRules command with empty Parameters list.

    - Deleting a rule with DeleteRules command.

Please refer to Section 5.6 for Rules Test Cases.

4.3.7 Face Recognition Rule

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

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

- DUT shall give the Media2 Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.

- DUT shall provide Face Recognition rule, if DUT supports this rule. Otherwise, these test cases will be skipped.

- DUT shall support the following commands:

- GetServiceCapabilities

- If DUT returns RuleSupport capability as supported, then DUT shall support commands listed below. Otherwise, these test cases will be skipped.

    - GetServiceCapabilities

    - GetSupportedRules

- If DUT returns RuleOptionsSupported capability as supported, then DUT shall support GetRuleOptions.

- The following tests are performed:

    - Receiving of Face Recognition rule description for each video analytics configuration that supports this rule.

    - If DUT supports receiving of rule options as indicated by RuleOptionsSupported capability:

        - Receiving of Face Recognition rule options for each video analytics configuration that supports this rule.

Please, refer to Section 5.7 for Face Recognition Rule Test Cases.

4.3.8 License Plate Recognition Rule

The test policies specific to the test case execution of License Plate Recognition functional block:

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

- DUT shall give the Media2 Service entry point by GetServices command, if DUT supports this service. Otherwise, these test cases will be skipped.

- DUT shall provide License Plate Recognition rule, if DUT supports this rule. Otherwise, these test cases will be skipped.

- DUT shall support the following commands:

    - GetServiceCapabilities

- If DUT returns RuleSupport capability as supported, then DUT shall support commands listed below. Otherwise, these test cases will be skipped.

    - GetServiceCapabilities

• GetSupportedRules

• If DUT returns RuleOptionsSupported capability as supported, then DUT shall support GetRuleOptions.

• The following tests are performed:

• Receiving of License Plate Recognition rule description for each video analytics configuration that supports this rule.

• If DUT supports receiving of rule options as indicated by RuleOptionsSupported capability:

• Receiving of License Plate Recognition rule options for each video analytics configuration that supports this rule.

Please, refer to Section 5.8 for License Plate Recognition Rule Test Cases.

5 Analytics Engine

5.1 Motion Region Detector

5.1.1 GET SUPPORTED RULES (MOTION REGION DETECTOR)

Test Case ID: ANALYTICS-1-1-1

Specification Coverage: Get Supported rules (ONVIF Analytics Service Spec), Motion Region Detector (ONVIF Analytics Service Spec)

Feature Under Test: GetSupportedRules, RuleDescription for tt:MotionRegionDetector

WSDL Reference: analytics.wsdl, media2.wsdl

Test Purpose: To verify that device includes tt:MotionRegionDetector in GetSupportedRulesResponse. To verify structure of Motion Region Detector.

Pre-Requisite: Analytics Service is received from the DUT. Media2 Service is received from the DUT. Rule Engine is supported by the Device as indicated by the RuleSupport capabilities. Motion Region Detector Rule is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

   • out *analyticsConfList* - a list of Analytics configurations

4. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

   4.1. ONVIF Client invokes **GetSupportedRules** request with parameters

      • ConfigurationToken := *analyticsConf*.@token

   4.2. DUT responds with **GetSupportedRulesResponse** message with parameters

      • SupportedRules =: *supportedRules*

4.3. If *supportedRules* contains RuleDescription element (*motionRegionDetectorRuleDescription*) with Name value is equal to **tt:MotionRegionDetector**:

4.3.1. If *motionRegionDetectorRuleDescription* does not have maxInstances attribute, FAIL the test and skip other steps.

4.3.2. If *motionRegionDetectorRuleDescription* does not have Parameters.ElementItemDescription element with Name attribute value is equal to "MotionRegion", FAIL the test and skip other steps.

4.3.3. If Type attribute value is not equal to "axt:MotionRegionConfig" for *motionRegionDetectorRuleDescription*.Parameters.ElementItemDescription with Name attribute value is equal to "MotionRegion", FAIL the test and skip other steps.

4.3.4. If *motionRegionDetectorRuleDescription* does not have Messages.Source.SimpleItemDescription element with Name attribute value is equal to "VideoSource", FAIL the test and skip other steps.

4.3.5. If Type attribute value is not equal to "tt:ReferenceToken" for *motionRegionDetectorRuleDescription*.Messages.Source.SimpleItemDescription with Name attribute value is equal to "VideoSource", FAIL the test and skip other steps.

4.3.6. If *motionRegionDetectorRuleDescription* does not have Messages.Source.SimpleItemDescription element with Name attribute value is equal to "RuleName", FAIL the test and skip other steps.

4.3.7. If Type attribute value is not equal to "xs:string" for *motionRegionDetectorRuleDescription*.Messages.Source.SimpleItemDescription with Name attribute value is equal to "RuleName", FAIL the test and skip other steps.

4.3.8. If *motionRegionDetectorRuleDescription* does not have Messages.Data.SimpleItemDescription element with Name attribute value is equal to "State", FAIL the test and skip other steps.

4.3.9. If Type attribute value is not equal to "xs:boolean" for *motionRegionDetectorRuleDescription*.Messages.Data.SimpleItemDescription with Name attribute value is equal to "State", FAIL the test and skip other steps.

4.3.10. If Messages.ParentTopic value is not equal to "tns1:RuleEngine/MotionRegionDetector/Motion" for Messages with Source.SimpleItemDescription.Name value is equal to VideoSource and with Source.SimpleItemDescription.Name value is equal to RuleName, FAIL the test and skip other steps.

5. If there was no RuleDescription element with Name value is equal to **tt:MotionRegionDetector** in at least one *supportedRules* at step 4.2 [34], FAIL the test.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.1.2 GET MOTION REGION DETECTOR RULE OPTIONS

Test Case ID: ANALYTICS-1-1-2

Specification Coverage: Get Rule Options (ONVIF Analytics Service Spec), Motion Region Detector (ONVIF Analytics Service Spec)

Feature Under Test: GetRuleOptions, MotionRegionConfigOptions

WSDL Reference: analytics.wsdl, media2.wsdl

Test Purpose: To verify retrieving of MotionRegionConfigOptions by GetRuleOptions operation.

Pre-Requisite: Analytics Service is received from the DUT. Media2 Service is received from the DUT. Rule Engine is supported by the Device as indicated by the RuleSupport capabilities, Rule Options is supported by the Device as indicated by the RuleOptionsSupported capabilities. Motion Region Detector Rule is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of Analytics Configurations that supports tt:MotionRegionDetector type by following the procedure mentioned in Annex A.2 with the following input and output parameters

- in **tt:MotionRegionDetector** - rule type

- out *analyticsConfListWithSupportingOfMotionRegionDetector* - a list of Analytics configurations

4. If *analyticsConfListWithSupportingOfMotionRegionDetector* is empty, FAIL the test and skip other steps.

5. For each Analytics Configuration *analyticsConf* in *analyticsConfListWithSupportingOfMotionRegionDetector* repeat the following steps:

   5.1. ONVIF Client invokes **GetRuleOptions** request with parameters

      - RuleType := tt:MotionRegionDetector

      - ConfigurationToken := *analyticsConf.*@token

   5.2. DUT responds with **GetRuleOptionsResponse** message with parameters

      - RuleOptions list =: *ruleOptionsList*

   5.3. If *ruleOptionsList* does not contain RuleOption with @Name = **MotionRegion** and with @Type = **axt:MotionRegionConfigOptions** (if @Type is present), FAIL the test and skip other steps.

   5.4. If RuleOption element with @Name = **MotionRegion** and with @Type = **axt:MotionRegionConfigOptions** (if @Type is present) does not contain **MotionRegionConfigOptions** element, FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.1.3 CREATE MOTION REGION DETECTOR RULE

Test Case ID: ANALYTICS-1-1-3

Specification Coverage: Create Rules (ONVIF Analytics Service Spec)

Feature Under Test: Create Rules

WSDL Reference: analytics.wsdl, media2.wsdl

Test Purpose: To verify adding of Motion Region Detector Rule to an AnalyticsConfiguration by CreateRules operation.

Pre-Requisite: Analytics Service is received from the DUT. Media2 Service is received from the DUT. Rule Engine is supported by the DUT as indicated by the RuleSupport capabilities. Motion Region Detector Rule is supported by the DUT. Profile T is supported.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client configures device for adding of new Motion Region Detector rule by following the procedure mentioned in Annex A.34 with the following input and output parameters

    • in "tt:MotionRegionDetector" - Rule type

    • out *nonFixedRuleSupported* - Indication of non-fixed FaceRecognition Rule support

    • out *analyticsToken* - a token of Analytics Configuration

    • out *ruleDescription* - a rule description

    • out *ruleToRestore* - a rule to restore, if it was removed

    • out *profile* - media profile with Analytics Configuration

4. If *nonFixedRuleSupported* = false, FAIL the test and skip other steps*.*

5. ONVIF Client retrieves Rule Options of tt:MotionRegionDetector type by following the procedure mentioned in Annex A.3 with the following input and output parameters

    • in **tt:MotionRegionDetector** - Rule type

    • in *analyticsToken* - Analytics Configuration token

    • out *ruleOptions* - Rule Options

6. If *ruleOptionsList* does not contain RuleOption with @Name = **MotionRegion** and with @Type = **axt:MotionRegionConfigOptions** (if @Type is present), FAIL the test and skip other steps.

7. If RuleOption element with @Name = **MotionRegion** and @Type = **axt:MotionRegionConfigOptions** (if @Type is present) does not contain **MotionRegionConfigOptions** element, FAIL the test and skip other steps.

8. Set *motionRegionConfigOptions* := RuleOption[0].MotionRegionConfigOptions, where RuleOption[0] is element with @Name = **MotionRegion** and with @Type = **axt:MotionRegionConfigOptions** (if @Type is present).

9. ONVIF Client invokes **CreateRules** request with parameters

   • ConfigurationToken := *profile*.Configurations.Analytics.@token

   • Rule[0].@Name := TestMotionRegion

   • Rule[0].@Type := tt:MotionRegionDetector

   • Rule[0].Parameters.ElementItem[0].@Name := "MotionRegion"

   • Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[0].@x := *profile*.Configurations.VideoSource.Bounds.@x

   • Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[0].@y := *profile*.Configurations.VideoSource.Bounds.@y

   • Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[1].@x := *profile*.Configurations.VideoSource.Bounds.@x

   • Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[1].@y := *profile*.Configurations.VideoSource.Bounds.@y + *profile*.Configurations.VideoSource.Bounds.@height - 1

   • Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[2].@x := *profile*.Configurations.VideoSource.Bounds.@x + *profile*.Configurations.VideoSource.Bounds.@width - 1

   • Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[2].@y := *profile*.Configurations.VideoSource.Bounds.@y + *profile*.Configurations.VideoSource.Bounds.@height - 1

   • Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[3].@x := *profile*.Configurations.VideoSource.Bounds.@x + *profile*.Configurations.VideoSource.Bounds.@width - 1

   • Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[3].@y := *profile*.Configurations.VideoSource.Bounds.@y

- Rule[0].Parameters.ElementItem[0].MotionRegion.@Armed := true if *motionRegionConfigOptions*.DisarmSupport = true, otherwise skipped.

- Rule[0].Parameters.ElementItem[0].MotionRegion.@Sensitivity := 1

10. The DUT responds with **CreateRulesResponse**.

11. ONVIF Client retrieves updated Rule list by following the procedure mentioned in Annex A.5 with the following input and output parameters

    - in *profile*.Configurations.Analytics.@token - Analytics configuration token

    - out *updatedRuleList* - Rule list.

12. If *updatedRuleList* does not contain Rule with @Name = TestMotionRegion and with Type = tt:MotionRegionDetector, FAIL the test and skip other steps.

13. Set *rule* := Rule with @Name = TestMotionRegion and with Type = tt:MotionRegionDetector from *updatedRuleList*.

14. If *rule*.Parameters does not contain ElementItem with @Name = "MotionRegion", FAIL the test and skip other steps.

15. If *rule*.Parameters.ElementItem with @Name = "MotionRegion" is not equal to Parameters.ElementItem[0] element from step 9 [39], FAIL the test and skip other steps.

16. ONVIF Client invokes **DeleteRules** request with parameters

    - ConfigurationToken := *profile*.Configurations.Analytics.@token

    - RuleName := TestMotionRegion

17. The DUT responds with **DeleteRulesResponse**.

18. ONVIF Client retrieves updated Rule list by following the procedure mentioned in Annex A.5 with the following input and output parameters

    - in *profile*.Configurations.Analytics.@token - Analytics configuration token

    - out *updatedRuleList* - Rule list.

19. If *updatedRuleList* contains Rule with @Name = TestMotionRegion and with Type = tt:MotionRegionDetector, FAIL the test and skip other steps.

20. ONVIF Client restores *ruleToRestore* if any.

21. ONVIF Client restores media profile if it was changed at step 3.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

Note: The following fields are compared at step 15 [40]:

- MotionRegion.Polygon.Point[0].@x

- MotionRegion.Polygon.Point[0].@y

- MotionRegion.Polygon.Point[1].@x

- MotionRegion.Polygon.Point[1].@y

- MotionRegion.Polygon.Point[2].@x

- MotionRegion.Polygon.Point[2].@y

- MotionRegion.Polygon.Point[3].@x

- MotionRegion.Polygon.Point[3].@y

  - If **CreateRules** request contained Armed filed:

    - MotionRegion.@Armed

- MotionRegion.@Sensitivity

5.1.4 MODIFY MOTION REGION DETECTOR RULE

Test Case ID: ANALYTICS-1-1-4

Specification Coverage: Get Rule Options (ONVIF Analytics Service Spec), Modify Rules (ONVIF Analytics Service Spec)

Feature Under Test: Modify Rules

WSDL Reference: analytics.wsdl, media2.wsdl

Test Purpose: To verify modifying of Motion Region Detector Rule by ModifyRules operation.

Pre-Requisite: Analytics Service is received from the DUT. Media2 Service is received from the DUT. Rule Engine is supported by the Device as indicated by the RuleSupport capability. Rule Options is supported by the Device as indicated by the RuleOptionsSupported capability. Motion Region Detector Rule is supported by the DUT. Profile T is supported.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client configures device for adding of new Motion Region Detector rule by following the procedure mentioned in Annex A.34 with the following input and output parameters

   • in "tt:MotionRegionDetector" - Rule type

   • out *nonFixedRuleSupported* - Indication of non-fixed FaceRecognition Rule support

   • out *analyticsToken* - a token of Analytics Configuration

   • out *ruleDescription* - a rule description

   • out *ruleToRestore* - a rule to restore, if it was removed

   • out *profile* - media profile with Analytics Configuration

4. If *nonFixedRuleSupported* = false, FAIL the test and skip other steps.

5. ONVIF Client retrieves Rule Options of tt:MotionRegionDetector type by following the procedure mentioned in Annex A.3 with the following input and output parameters

   • in **tt:MotionRegionDetector** - Rule type

   • in *profile*.Configurations.Analytics.token - Analytics Configuration token

   • out *ruleOptions* - Rule Options

6. If *ruleOptionsList* does not contain RuleOption with @Name = **MotionRegion** and with @Type = **axt:MotionRegionConfigOptions** (if @Type is present), FAIL the test and skip other steps.

7. If RuleOption element with @Name = **MotionRegion** and @Type = **axt:MotionRegionConfigOptions** (if @Type is present) does not contain **MotionRegionConfigOptions** element, FAIL the test and skip other steps.

8. Set *motionRegionConfigOptions* := RuleOption[0].MotionRegionConfigOptions, where RuleOption[0] is element with @Name = **MotionRegion** and with @Type = **axt:MotionRegionConfigOptions** (if @Type is present).

9. ONVIF Client creates Motion Region Detector Rule by following the procedure mentioned in Annex A.10 with the following input parameter

   • in *analyticsToken* - Analytics Configuration Token.

   • in *motionRegionConfigOptions* - motion region configuration option.

10. ONVIF Client invokes **ModifyRules** request with parameters

   • ConfigurationToken := *profile*.Configurations.Analytics.@token

   • Rule[0].@Name := TestMotionRegion

   • Rule[0].@Type := tt:MotionRegionDetector

   • Rule[0].Parameters.ElementItem[0].@Name := "MotionRegion"

   • Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[0].@x := *profile*.Configurations.VideoSource.Bounds.@x

   • Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[0].@y := *profile*.Configurations.VideoSource.Bounds.@y

   • Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[1].@x := *profile*.Configurations.VideoSource.Bounds.@x

   • Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[1].@y := [(*profile*.Configurations.VideoSource.Bounds.@y + *profile*.Configurations.VideoSource.Bounds.@height - 1)/2]

   • Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[2].@x := [(*profile*.Configurations.VideoSource.Bounds.@x + *profile*.Configurations.VideoSource.Bounds.@width - 1)/2]

   • Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[2].@y := [(*profile*.Configurations.VideoSource.Bounds.@y + *profile*.Configurations.VideoSource.Bounds.@height - 1)/2]

   • Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[3].@x := [(*profile*.Configurations.VideoSource.Bounds.@x + *profile*.Configurations.VideoSource.Bounds.@width - 1)/2]

- Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[3].@y :=
  profile.Configurations.VideoSource.Bounds.@y

- Rule[0].Parameters.ElementItem[0].MotionRegion.@Armed := false if
  motionRegionConfigOptions.DisarmSupport = true, otherwise skipped.

- Rule[0].Parameters.ElementItem[0].MotionRegion.@Sensitivity := 0

11. The DUT responds with **ModifyRulesResponse**.

12. ONVIF Client retrieves updated Rule list by following the procedure mentioned in Annex A.5
    with the following input and output parameters

    - in *profile*.Configurations.Analytics.@token - Analytics configuration token

    - out *updatedRuleList* - Rule list.

13. If *updatedRuleList* does not contain Rule with @Name = **"TestMotionRegion"** and with
    Type = **tt:MotionRegionDetector**, FAIL the test and skip other steps.

14. Set *rule* := Rule with @Name = "TestMotionRegion" and with Type =
    tt:MotionRegionDetector from *updatedRuleList*.

15. If *rule*.Parameters does not contain ElementItem with @Name = "MotionRegion", FAIL the
    test and skip other steps.

16. If *rule*.Parameters.ElementItem with @Name = **"MotionRegion"** is not equal to
    Parameters.ElementItem[0] element from step 11 [43], FAIL the test and skip other steps.

17. ONVIF Client invokes **DeleteRules** request with parameters

    - ConfigurationToken := *profile*.Configurations.Analytics.@token

    - RuleName := TestMotionRegion

18. The DUT responds with **DeleteRulesResponse**.

19. ONVIF Client restores *ruleToRestore* if any.

20. ONVIF Client restores media profile if it was changed at step 3.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

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

Note: Symbol [] at step 10 [43] means integer part of value (floor function).

Note: The following fields are compared at step 16 [44]:

- MotionRegion.Polygon.Point[0].@x

- MotionRegion.Polygon.Point[0].@y

- MotionRegion.Polygon.Point[1].@x

- MotionRegion.Polygon.Point[1].@y

- MotionRegion.Polygon.Point[2].@x

- MotionRegion.Polygon.Point[2].@y

- MotionRegion.Polygon.Point[3].@x

- MotionRegion.Polygon.Point[3].@y

- If **ModifyRules** request contained Armed filed:

    - MotionRegion.@Armed

- MotionRegion.@Sensitivity

5.2 Events

5.2.1 MOTION REGION DETECTOR EVENT

Test Case ID: ANALYTICS-2-1-1

Specification Coverage: Motion Region Detector (ONVIF Analytics Service Spec)

Feature Under Test: tns1:RuleEngine/MotionRegionDetector/Motion

WSDL Reference: analytics.wsdl, media2.wsdl

Test Purpose: To verify tns1:RuleEngine/MotionRegionDetector/Motion event format. To verify event generation for tns1:RuleEngine/MotionRegionDetector/Motion.

Pre-Requisite: Analytics Service is received from the DUT. Media2 Service is received from the DUT. Rule Engine is supported by the Device as indicated by the RuleSupport capability. Rule Options is supported by the Device as indicated by the RuleOptionsSupported capability. Motion Region Detector Rule is supported by the DUT. Profile T is supported.

Test Configuration: ONVIF Client and DUT.

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client configures device for adding of new Motion Region Detector rule by following the procedure mentioned in Annex A.34 with the following input and output parameters

    - in "tt:MotionRegionDetector" - Rule type

    - out *nonFixedRuleSupported* - Indication of non-fixed FaceRecognition Rule support

    - out *analyticsToken* - a token of Analytics Configuration

    - out *ruleDescription* - a rule description

    - out *ruleToRestore* - a rule to restore, if it was removed

4. If *nonFixedRuleSupported* = false, FAIL the test and skip other steps.

5. ONVIF Client retrieves Rule Options of tt:MotionRegionDetector type by following the procedure mentioned in Annex A.3 with the following input and output parameters

    - in **tt:MotionRegionDetector** - Rule type

    - in *profile*.Configurations.Analytics.token - Analytics Configuration token

    - out *ruleOptions* - Rule Options

6. If *ruleOptionsList* does not contain RuleOption with @Name = **MotionRegion** and with @Type = **axt:MotionRegionConfigOptions** (if @Type is present), FAIL the test and skip other steps.

7. If RuleOption element with @Name = **MotionRegion** and @Type = **axt:MotionRegionConfigOptions** (if @Type is present) does not contain **MotionRegionConfigOptions** element, FAIL the test and skip other steps.

8. Set *motionRegionConfigOptions* := RuleOption[0].MotionRegionConfigOptions, where RuleOption[0] is element with @Name = **MotionRegion** and with @Type = **axt:MotionRegionConfigOptions** (if @Type is present).

9. ONVIF Client creates Motion Region Detector Rule by following the procedure mentioned in Annex A.10 with the following input parameter

    • in *analyticsToken* - Analytics Configuration Token.

    • in *motionRegionConfigOptions* - motion region configuration option.

10. ONVIF Client invokes **GetEventProperties**.

11. The DUT responds with **GetEventPropertiesResponse** with parameters

    • TopicNamespaceLocation list

    • FixedTopicSet

    • TopicSet =: *topicSet*

    • TopicExpressionDialect list

    • MessageContentFilterDialect list

    • MessageContentSchemaLocation list

12. If *topicSet* does not contain **tns1:RuleEngine/MotionRegionDetector/Motion** topic, FAIL the test and skip other steps.

13. Set *topic* := tns1:RuleEngine/MotionRegionDetector/Motion topic from *topicSet*.

14. If *topic*.MessageDescription.IsProperty is not equal to true, FAIL the test and skip other steps.

15. If *topic* does not contain MessageDescription.Source.SimpleItemDescription item with Name = "VideoSource", FAIL the test and skip other steps.

16. If *topic*.MessageDescription.Source.SimpleItemDescription with Name = "VideoSource" does not have Type = "tt:ReferenceToken", FAIL the test and skip other steps.

17. If *motionRegionConfigOptions*.MotionRegionConfigOptions.RuleNotification = true:

    17.1. If *topic* does not contain MessageDescription.Source.SimpleItemDescription item with Name = "RuleName", FAIL the test and skip other steps.

    17.2. If *topic*.MessageDescription.Source.SimpleItemDescription with Name = "RuleName" does not have Type = "xs:string", FAIL the test and skip other steps.

18. If *topic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "State", FAIL the test and skip other steps.

19. If *topic*.MessageDescription.Data.SimpleItemDescription with Name = "State" does not have Type = "xs:boolean", FAIL the test and skip other steps.

20. If DUT supports Pull-Point Notification feature

    20.1. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.6 with the following input and output parameters

- in **"tns1:RuleEngine/MotionRegionDetector/Motion"** - Notification Topic

- out *s* - Subscription reference

- out *currentTime* - current time for the DUT

- out *terminationTime* - Subscription termination time

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

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

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

- Timeout := PT60S

- MessageLimit := 1

    20.2.3 The DUT responds with **PullMessagesResponse** message with parameters

- CurrentTime =: *ct*

- TerminationTime =: *tt*

- NotificationMessage list =: *notificationMessageList*

    20.2.4 If *notificationMessageList* contains more than one notification, FAIL the test and skip other steps.

    20.2.5 If *notificationMessageList* is not empty and *notificationMessageList*[0].Topic is not equal to **"tns1:RuleEngine/MotionRegionDetector/Motion"**, FAIL the test and skip other steps.

    20.2.6 If *notificationMessageList* is not empty and *notificationMessageList*[0].PropertyOperation = "Initialized" and *notificationMessageList*[0] has Source.SimpleItem with Name = "VideoSource" and with Value = *profile*.Configurations.VideoSource.SourceToken:

20.2.6.1 If
motionRegionConfigOptions.MotionRegionConfigOptions.RuleNotification
is not equal to true, go to step 21.3 [49].

20.2.6.2 If *notificationMessageList*[0] has Source.SimpleItem with Name =
"RuleName" and with Value = "TestMotionRegion", go to step
21.3 [49].

20.2.7 If *timeout1* timeout expires for step 21.2 without Notification corresponds to step
21.2.6 [48], FAIL the test and skip other steps.

20.3. If *notificationMessageList*[0] does not have Data.SimpleItem with Name = "State" and
with Value with type = "xs:boolean", FAIL the test and skip other steps.

21. ONVIF Client invokes **DeleteRules** request with parameters

- ConfigurationToken := *profile*.Configurations.Analytics.@token

- RuleName := TestMotionRegion

22. The DUT responds with **DeleteRulesResponse**.

23. ONVIF Client restores rule if it was deleted at step 10 [47].

24. ONVIF Client restores *ruleToRestore* if any.

25. ONVIF Client restores media profile if it was changed at step 3.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

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

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

5.2.2 FACE RECOGNITION EVENT

Test Case ID: ANALYTICS-2-1-2

Specification Coverage: Recognition rule engines (ONVIF Analytics Service Spec), Face Recognition (ONVIF Analytics Service Spec)

Feature Under Test: tns1:RuleEngine/Recognition/Face

WSDL Reference: event.wsdl

Test Purpose: To verify tns1:RuleEngine/Recognition/Face topic format.

Pre-Requisite: Event Service is received from the DUT. Face Recognition Rule is supported by the DUT.

Test Configuration: ONVIF Client and DUT.

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client invokes **GetEventProperties**.

4. The DUT responds with **GetEventPropertiesResponse** with parameters

    • TopicNamespaceLocation list

    • FixedTopicSet

    • TopicSet =: *topicSet*

    • TopicExpressionDialect list

    • MessageContentFilterDialect list

    • MessageContentSchemaLocation list

5. If *topicSet* does not contain **tns1:RuleEngine/Recognition/Face** topic, FAIL the test and skip other steps.

6. Set *topic* := tns1:RuleEngine/Recognition/Face topic from *topicSet*.

7. If *topic*.MessageDescription contains @IsProperty = true, FAIL the test and skip other steps.

8. ONVIF Client verifies rule engine message format by following the procedure mentioned in Annex A.46 with the following input and output parameters

    • in *topic*.MessageDescription - message description to be verified

9. If *topic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "Likelihood", FAIL the test and skip other steps.

10. If *topic*.MessageDescription.Data.SimpleItemDescription with Name = "Likelihood" does not have Type = "xs:float", FAIL the test and skip other steps.

11. If *topic* contains MessageDescription.Data.SimpleItemDescription item with Name = "Label" and its Type != "xs:string", FAIL the test and skip other steps.

12. If *topic* contains MessageDescription.Data.SimpleItemDescription item with Name = "ImageUri" and its Type != "xs:anyURI", FAIL the test and skip other steps.

13. If *topic* contains MessageDescription.Data.SimpleItemDescription item with Name = "EnrollmentID" and its Type != "xs:string", FAIL the test and skip other steps.

14. If *topic* contains MessageDescription.Data.SimpleItemDescription item with Name = "RefImageUri" and its Type != "xs:anyURI", FAIL the test and skip other steps.

15. If *topic* contains MessageDescription.Data.ElementItemDescription item with Name = "Image" and its Type != "xs:base64Binary", FAIL the test and skip other steps.

16. If *topic* contains MessageDescription.Data.ElementItemDescription item with Name = "BoundingBox" and its Type != "tt:Rectangle", FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.3 LICENSE PLATE RECOGNITION EVENT

Test Case ID: ANALYTICS-2-1-3

Specification Coverage: Recognition rule engines (ONVIF Analytics Service Spec), License Plate Recognition (ONVIF Analytics Service Spec)

Feature Under Test: tns1:RuleEngine/Recognition/LicensePlate

WSDL Reference: event.wsdl

Test Purpose: To verify tns1:RuleEngine/Recognition/LicensePlate topic format.

Pre-Requisite: Event Service is received from the DUT. License Plate Recognition Rule is supported by the DUT.

Test Configuration: ONVIF Client and DUT.

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client invokes **GetEventProperties**.

4. The DUT responds with **GetEventPropertiesResponse** with parameters

    • TopicNamespaceLocation list

    • FixedTopicSet

    • TopicSet =: *topicSet*

    • TopicExpressionDialect list

    • MessageContentFilterDialect list

    • MessageContentSchemaLocation list

5. If *topicSet* does not contain **tns1:RuleEngine/Recognition/LicensePlate** topic, FAIL the test and skip other steps.

6. Set *topic* := tns1:RuleEngine/Recognition/LicensePlate topic from *topicSet*.

7. If *topic*.MessageDescription contains @IsProperty = true, FAIL the test and skip other steps.

8. ONVIF Client verifies rule engine message format by following the procedure mentioned in Annex A.46 with the following input and output parameters

    • in *topic*.MessageDescription - message description to be verified

9. If *topic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "Likelihood", FAIL the test and skip other steps.

10. If *topic*.MessageDescription.Data.SimpleItemDescription with Name = "Likelihood" does not have Type = "xs:float", FAIL the test and skip other steps.

11. If *topic* contains MessageDescription.Data.SimpleItemDescription item with Name = "Label" and its Type != "xs:string", FAIL the test and skip other steps.

12. If *topic* contains MessageDescription.Data.SimpleItemDescription item with Name = "ImageUri" and its Type != "xs:anyURI", FAIL the test and skip other steps.

13. If *topic* contains MessageDescription.Data.SimpleItemDescription item with Name = "VehicleImageURI" and its Type != "xs:anyURI", FAIL the test and skip other steps.

14. If *topic* contains MessageDescription.Data.ElementItemDescription item with Name = "BoundingBox" and its Type != "tt:Rectangle", FAIL the test and skip other steps.

15. If *topic* contains MessageDescription.Data.ElementItemDescription item with Name = "Image" and its Type != "xs:base64Binary", FAIL the test and skip other steps.

16. If *topic* does not contain MessageDescription.Data.ElementItemDescription item with Name = "LicensePlateInfo", FAIL the test and skip other steps.

17. If *topic*.MessageDescription.Data.ElementItemDescription with Name = "LicensePlateInfo" does not have Type = "tt:LicensePlateInfo", FAIL the test and skip other steps.

18. If *topic* contains MessageDescription.Data.ElementItemDescription item with Name = "VehicleInfo" and its Type != "tt:VehicleInfo", FAIL the test and skip other steps.

19. If *topic* contains MessageDescription.Data.ElementItemDescription item with Name = "VehicleImage" and its Type != "xs:base64Binary", FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.2.4 LINE CROSSING COUNTER EVENT

Test Case ID: ANALYTICS-2-1-4

Specification Coverage: Line crossing counting (ONVIF Analytics Service Spec)

Feature Under Test: tns1:RuleEngine/CountAggregation/Counter

WSDL Reference: event.wsdl

Test Purpose: To verify tns1:RuleEngine/CountAggregation/Counter topic format. To verify tns1:RuleEngine/CountAggregation/Counter initialize event generation.

Pre-Requisite: Event Service is supported by the DUT. Line crossing counting Rule is supported by the DUT. Profile M is supported.

Test Configuration: ONVIF Client and DUT.

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client invokes **GetEventProperties**.

4. The DUT responds with **GetEventPropertiesResponse** with parameters

   • TopicNamespaceLocation list

   • FixedTopicSet

   • TopicSet =: *topicSet*

   • TopicExpressionDialect list

   • MessageContentFilterDialect list

   • MessageContentSchemaLocation list

5. If *topicSet* does not contain **tns1:RuleEngine/CountAggregation/Counter** topic, FAIL the test and skip other steps.

6. Set *topic* := tns1:RuleEngine/CountAggregation/Counter topic from *topicSet*.

7. If *topic*.MessageDescription.@IsProperty = !true, FAIL the test and skip other steps.

8. ONVIF Client verifies rule engine message format by following the procedure mentioned in Annex A.46 with the following input and output parameters

   • in *topic*.MessageDescription - message description to be verified

9. If *topic* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "Count", FAIL the test and skip other steps.

10. If *topic*.MessageDescription.Data.SimpleItemDescription with Name = "Count" does not have Type = "xs:int", FAIL the test and skip other steps.

11. If DUT supports Pull-Point Notification feature

    11.1. ONVIF Client configures profile with Line crossing counting rule by following the procedure mentioned in Annex A.43 with the following input and output parameters

       • in "tt:LineCounting" - Rule type

       • out *videoSourceToken* - token of the video source present in the media profile

       • out *ruleName* - name of the rule present in the media profile

       • out *analyticsToken* - a token of Analytics Configuration present in the media profile

- out *ruleToRestore* - a rule to restore, if it was removed

- out *ruleToDelete* - a rule to delete, if it was created

11.2. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.6 with the following input and output parameters

- in **"tns1:RuleEngine/CountAggregation/Counter"** - Notification Topic

- out *s* - Subscription reference

- out *currentTime* - current time for the DUT

- out *terminationTime* - Subscription termination time

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

11.3.1 ONVIF Client waits for time $t := \min\{(tt\text{-}ct)/2, 1 \text{ second}\}$.

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

- Timeout := PT60S

- MessageLimit := 1

11.3.3 The DUT responds with **PullMessagesResponse** message with parameters

- CurrentTime =: *ct*

- TerminationTime =: *tt*

- NotificationMessage list =: *notificationMessageList*

11.3.4 If *notificationMessageList* contains more than one notification, FAIL the test and skip other steps.

11.3.5 If *notificationMessageList* is not empty and corresponds the following parameters:

- *notificationMessageList*[0].Topic = **"tns1:RuleEngine/CountAggregation/Counter"**,

- *notificationMessageList*[0].PropertyOperation = "Initialized",

- *notificationMessageList*[0] has Source.SimpleItem with Name = "Rule" and with Value = *ruleName*,

- *notificationMessageList*[0] has Source.SimpleItem with Name = "VideoSource" and with Value = *videoSourceToken*,

- if *notificationMessageList*[0] has Source.SimpleItem with Name = "AnalyticsConfiguration" then its Value = *analyticsToken*, then the ONVIF Client checks notification:

  11.3.5.1 If *notificationMessageList*[0] does not contain Data.SimpleItem with Name = "Count", FAIL the test, restore DUT settings and skip other steps.

  11.3.5.2 If Data.SimpleItem with Name = "Count" has Value type differ from integer, FAIL the test, restore DUT settings and skip other steps.

  11.3.5.3 Go to step 11.4.

  11.3.6 If *timeout1* timeout expires for step 11.3 without Notification corresponds to parameters in step 11.3.6 [55], FAIL the test and skip other steps.

11.4. ONVIF Client deletes *ruleToDelete* if any to restore DUT settings.

11.5. ONVIF Client restores media profile if it was changed at step 11.1.

11.6. ONVIF Client creates *ruleToRestore* if any to restore DUT settings.

11.7. ONVIF Client deletes PullPoint subscription by following the procedure mentioned in Annex A.7 with the following input and output parameters

- in *s* - Subscription reference

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

5.3 Capabilities

5.3.1 GET SERVICES AND GET ANALYTICS SERVICES CAPABILITIES CONSISTENCY

Test Case ID: ANALYTICS-3-1-1

Specification Coverage: Capability exchange (ONVIF Core Specification), Capabilities (ONVIF Analytics Service Spec)

Feature under test: GetServices, GetServiceCapabilities (Analytics)

WSDL Reference: devicemgmt.wsdl, analytics.wsdl

Test Purpose: To verify getting Analytics Service using GetServices request. To verify Get Services and Analytics Service Capabilities consistency.

Pre-Requisite: Analytics Service was received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client invokes **GetServices** message with parameters:

   • IncludeCapability := false

4. The DUT responds with a **GetServicesResponse** message with parameters:

   • Service list =: *listOfServicesWithoutCapabilities*

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

6. Set *analyticsServ* := item from *listOfServicesWithoutCapabilities* list with Namespace = "http://www.onvif.org/ver20/analytics/wsdl".

7. If *analyticsServ*.Capabilities is specified, FAIL the test and skip other steps.

8. ONVIF Client invokes **GetServices** message with parameters:

- IncludeCapability := true

9. The DUT responds with a **GetServicesResponse** message with parameters:

   - Service list =: *listOfServicesWithCapabilities*

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

11. Set *analyticsServ* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver20/analytics/wsdl".

12. If *analyticsServ*.Capabilities is not specified, FAIL the test and skip other steps.

13. If *analyticsServ*.Capabilities does not contain valid Capabilities element for Analytics service from "http://www.onvif.org/ver20/analytics/wsdl" namespace, FAIL the test and skip other steps.

14. ONVIF Client invokes **GetServiceCapabilities** (Analytics) request.

15. The DUT responds with **GetServiceCapabilitiesResponse** message with parameters

   - Capabilities =: *cap*

16. If *cap* differs from *analyticsServ*.Capabilities.Capabilities (see Note at the end of the test), FAIL the test.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

Note: The following fields are compared at step 16:

- RuleSupport

- AnalyticsModuleSupport

- RuleOptionsSupported

- AnalyticsModuleOptionsSupported

- SupportedMetadata

5.3.2 ANALYTICS SERVICE CAPABILITIES

Test Case ID: ANALYTICS-3-1-2

Specification Coverage: Capabilities (ONVIF Analytics Service Spec)

Feature under test: GetServiceCapabilities (Analytics Service)

WSDL Reference: analytics.wsdl

Test Purpose: To verify Analytics Service Capabilities.

Pre-Requisite: Analytics Service was received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client invokes **GetServiceCapabilities** request.

4. The DUT responds with **GetServiceCapabilitiesResponse** message with parameters

    - Capabilities =: *cap*

5. If *cap*.SupportedMetadata = true and *cap*.AnalyticsModuleSupport = false or skipped, FAIL the test.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

5.4 Analytics Modules

5.4.1 GET SUPPORTED ANALYTICS MODULES

Test Case ID: ANALYTICS-4-1-1

Specification Coverage: GetSupportedAnalyticsModules (ONVIF Analytics Service Spec)

Feature under test: GetSupportedAnalyticsModules (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify getting supported analytics modules using GetSupportedAnalyticsModules request.

Pre-Requisite: Analytics Service was received from the DUT. Media2 Service was received from the DUT. Analytics Modules is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

    • out *analyticsConfList* - a list of Analytics configurations

4. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

    4.1. ONVIF Client invokes **GetSupportedAnalyticsModules** request with parameters

        • ConfigurationToken := *analyticsConf*.@token

    4.2. DUT responds with **GetSupportedAnalyticsModulesResponse** message with parameters

        • SupportedAnalyticsModules =: *supportedModules*

    4.3. For each Analytics Module Description *analyticsModuleDescription* in *supportedModules*.AnalyticsModuleDescription list repeat the following steps:

4.3.1. If *analyticsModuleDescription* does not have maxInstances attribute, FAIL the test, restore the DUT state, and skip other steps.

4.3.2. If at least one item in *analyticsModuleDescription*.Parameters list (SimpleItemDescription or ElementItemDescription item) has the same Name value with other item from the same list, FAIL the test, restore the DUT state, and skip other steps.

4.3.3. If at least one item in *analyticsModuleDescription*.Messages list (Source, or Key, or Data; SimpleItemDescription or ElementItemDescription item) has the same Name value with other item from the same list, FAIL the test, restore the DUT state, and skip other steps.

4.3.4. If *analyticsModuleDescription*.ParentTopic is not valid topic (see Annex A.11), FAIL the test, restore the DUT state, and skip other steps.

4.3.5. If *analyticsModuleDescription*.fixed = true:

4.3.5.1. If *analyticsConf* does not contain at least one AnalyticsEngineConfiguration.AnalyticsModule with Type = *analyticsModuleDescription*.Name, FAIL the test, restore the DUT state, and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **GetSupportedAnalyticsModulesResponse** messages.

5.4.2 GET ANALYTICS MODULES OPTIONS

Test Case ID: ANALYTICS-4-1-2

Specification Coverage: GetAnalyticsModuleOptions (ONVIF Analytics Service Spec)

Feature under test: GetAnalyticsModuleOptions (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify getting supported analytics module options using GetAnalyticsModuleOptions request.

Pre-Requisite: Analytics Service was received from the DUT. Media2 Service was received from the DUT. Analytics Modules is supported by the DUT. Analytics Module Options is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

    • out *analyticsConfList* - a list of Analytics configurations

4. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

    4.1. ONVIF Client invokes **GetSupportedAnalyticsModules** request with parameters

        • ConfigurationToken := *analyticsConf*.@token

    4.2. DUT responds with **GetSupportedAnalyticsModulesResponse** message with parameters

        • SupportedAnalyticsModules =: *supportedModules*

    4.3. ONVIF Client invokes **GetAnalyticsModuleOptions** request with parameters

        • Type is skipped

        • ConfigurationToken := *analyticsConf*.@token

    4.4. DUT responds with **GetAnalyticsModuleOptionsResponse** message with parameters

        • Options list =: *analyticsModuleOptionsFullList*

    4.5. If at least one item in *analyticsModuleOptionsFullList* list contains RuleType, FAIL the test, restore the DUT state, and skip other steps.

    4.6. For each Analytics Module Description *analyticsModuleDescription* in *supportedModules*.AnalyticsModuleDescription list repeat the following steps:

        4.6.1. ONVIF Client invokes **GetAnalyticsModuleOptions** request with parameters

- Type := *analyticsModuleDescription*.Name

- ConfigurationToken := *analyticsConf*.@token

4.6.2. DUT responds with **GetAnalyticsModuleOptionsResponse** message with parameters

- Options list =: *analyticsModuleOptionsList*

4.6.3. If at least one item in *analyticsModuleOptionsList* list contains RuleType, FAIL the test, restore the DUT state, and skip other steps.

4.6.4. If at least one item in *analyticsModuleOptionsList* list contains AnalyticsModule which is not equal to *analyticsModuleDescription*.Name, FAIL the test, restore the DUT state, and skip other steps.

4.6.5. If *analyticsModuleOptionsFullList* list does not contain all items from *analyticsModuleOptionsList* list (AnalyticsModule and Name to be used as unique identifier), FAIL the test, restore the DUT state, and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **GetSupportedAnalyticsModulesResponse** messages.

- The DUT did not send **GetAnalyticsModuleOptionsResponse** messages.

5.4.3 GET ANALYTICS MODULES

Test Case ID: ANALYTICS-4-1-3

Specification Coverage: GetAnalyticsModules (ONVIF Analytics Service Spec)

Feature under test: GetAnalyticsModules (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify getting supported analytics modules using GetAnalyticsModules request.

Pre-Requisite: Analytics Service was received from the DUT. Media2 Service was received from the DUT. Analytics Modules is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

   • out *analyticsConfList* - a list of Analytics configurations

4. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

   4.1. ONVIF Client invokes **GetSupportedAnalyticsModules** request with parameters

      • ConfigurationToken := *analyticsConf*.@token

   4.2. DUT responds with **GetSupportedAnalyticsModulesResponse** message with parameters

      • SupportedAnalyticsModules =: *supportedModules*

   4.3. ONVIF Client invokes **GetAnalyticsModules** request with parameters

      • ConfigurationToken := *analyticsConf*.@token

   4.4. DUT responds with **GetAnalyticsModulesResponse** message with parameters

      • AnalyticsModule list =: *analyticsModuleList*

   4.5. If at least one item in *analyticsModuleList* list contains Type which in not equal to Name fileld of at least one item at *supportedModules*.AnalyticsModuleDescription list, FAIL the test, restore the DUT state, and skip other steps.

   4.6. For each Analytics Module Description *analyticsModuleDescription* in *supportedModules*.AnalyticsModuleDescription list repeat the following steps:

      4.6.1. If *analyticsModuleDescription*.fixed = true:

         4.6.1.1. If *analyticsModuleList* does not contain item with Type = *analyticsModuleDescription*.Name, FAIL the test, restore the DUT state, and skip other steps.

      4.6.2. If *analyticsModuleList* contains item with Type = *analyticsModuleDescription*.Name:

4.6.2.1.   Set   *analyticsModule*   :=   *analyticsModuleList*[Type   = *analyticsModuleDescription*.Name].

4.6.2.2.   If *analyticsModule*.Parameters.SimpleItem contains at least one SimpleItem which does not have item with the same Name in *analyticsModuleDescription*.Parameters.SimpleItemDescription list, FAIL the test, restore the DUT state, and skip other steps.

4.6.2.3.   If *analyticsModule*.Parameters.ElementItem contains at least one ElementItem which does not have item with the same Name in *analyticsModuleDescription*.Parameters.ElementItemDescription list, FAIL the test, restore the DUT state, and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **GetSupportedAnalyticsModulesResponse** messages.

- The DUT did not send **GetAnalyticsModulesResponse** messages.

5.4.4 GET SUPPORTED METADATA

Test Case ID: ANALYTICS-4-1-4

Specification Coverage: GetSupportedMetadata (ONVIF Analytics Service Spec)

Feature under test: GetSupportedMetadata (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify getting supported metadata using GetSupportedMetadata request.

Pre-Requisite: Analytics Service was received from the DUT. Media2 Service was received from the DUT. Analytics Modules is supported by the DUT. Supported Metadata feature is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves full list of supported Analytics Modules by following the procedure mentioned in Annex A.13 with the following input and output parameters

   • out *fullAnalyticsModuleDescriptionList* - a list of supported Analytics Module Description

4. Set *analyticsModuleTypeList* := list of @Name values from *fullAnalyticsModuleDescriptionList*

5. ONVIF Client retrieves a list of supported metadata by following the procedure mentioned in Annex A.14 with the following input and output parameters

   • out *analyticsModuleList1* - Analytics Module List with Metadata Info

6. If *analyticsModuleList1* is empty, FAIL the test and skip other steps.

7. If *analyticsModuleList1* contains at least two elements with the same @Type, FAIL the test and skip other steps.

8. For each Analytics Module *analyticsModule* from *analyticsModuleList1* repeat the following steps:

   8.1. If *analyticsModule*.@Type is not equal to at least one Type from *analyticsModuleTypeList*, FAIL the test and skip other steps.

   8.2. ONVIF Client checks that object bounding boxes and shapes are located in the top left quarter of the image by following the procedure mentioned in Annex A.25 with the following input and output parameters

      • in *analyticsModule*.SampleFrame - sample frame of analytics module

9. For each Analytics Module Type *analyticsModuleType* from *analyticsModuleTypeList* repeat the following steps:

   9.1. ONVIF Client invokes **GetSupportedMetadata** request with parameters

      • Type := *analyticsModuleType*

   9.2. DUT responds with **GetSupportedMetadataResponse** message with parameters

      • AnalyticsModule list =: *analyticsModuleList2*

   9.3. If *analyticsModuleList2* contains more than one element, FAIL the test and skip other steps.

9.4. If *analyticsModuleList2*[0].@Type is not equal to *analyticsModuleType*, FAIL the test and skip other steps.

9.5. If set of fields in *analyticsModuleList2*[0] is not equal to set of filels in corresponding AnalyticsModule from *analyticsModuleList1* (see Note at the end of the test), FAIL the test and skip other steps.

9.6. ONVIF Client checks that object bounding boxes and shapes are located in the top left quarter of the image by following the procedure mentioned in Annex A.25 with the following input and output parameters

- in *analyticsModuleList2*[0].SampleFrame - sample frame of analytics module

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **GetSupportedMetadataResponse** messages.

Note: Presence of all fields without their values are compared at step 8.5, @Type value is used as key.

5.4.5 CREATE ANALYTICS MODULES

Test Case ID: ANALYTICS-4-1-5

Specification Coverage: CreateAnalyticsModules (ONVIF Analytics Service Spec)

Feature under test: CreateAnalyticsModules (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify creating of analytics module using CreateAnalyticsModules request.

Pre-Requisite: Analytics Service was received from the DUT. Media2 Service was received from the DUT. Analytics Modules is supported by the DUT. Profile M is supported.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client configures device for adding of new analytics module by following the procedure mentioned in Annex A.23 with the following input and output parameters

    • out *nonFixedModuleSupported* - if non fixed Analytics Module is supported

    • out (optional) *analyticsToken* - a token of Analytics Configuration

    • out (optional) *analyticsModuleDescription* - analytics module description

    • out *moduleToRestore* (optional) - analytics module to restore

4. If *nonFixedModuleSupported* = false, skip other steps with PASS result.

5. Set *moduleName* := generated random string.

6. ONVIF Client invokes **CreateAnalyticsModules** request with parameters

    • ConfigurationToken := *profile*.Configurations.Analytics.token

    • AnalyticsModule[0].@Name := *moduleName*

    • AnalyticsModule[0].@Type := *analyticsModuleDescription*.@Name

    • AnalyticsModule[0].Parameters := empty Parameters element

7. DUT responds with **CreateAnalyticsModulesResponse** message

8. ONVIF Client retrieves a set of analytics modules assigned of Analytics Configuration by following the procedure mentioned in Annex A.20 with the following input and output parameters

    • in *profile*.Configurations.Analytics.token - a token of Analytics Configuration

    • out *analyticsModuleList* - analytics module list

9. If *analyticsModuleList* does not contain AnalyticsModule with @Type = *analyticsModuleDescription*.@Name and with @Name = *moduleName*, FAIL the test.

10. Set *createdAnalyticsModule1* := AnalyticsModule with @Type = *analyticsModuleDescription*.@Name and with @Name = *moduleName* from *analyticsModuleList*.

11. For each SimpleItem element *simpleItem* in *createdAnalyticsModule1*.Parameters repeat the following steps:

11.1. If *supportedAnalyticsModules*.AnalyticsModuleDescription[0].Parameters does not contain at least one SimpleItemDescription with @Name = *simpleItem*.@Name, FAIL the test and skip other steps.

12. For each ElementItem element *elementItem* in *createdAnalyticsModule1*.Parameters repeat the following steps:

12.1. If *supportedAnalyticsModules*.AnalyticsModuleDescription[0].Parameters does not contain at least one ElementItemDescription with @Name = *elementItem*.@Name, FAIL the test and skip other steps.

13. ONVIF Client retrieves an Analytics Configuration by following the procedure mentioned in Annex A.21 with the following input and output parameters

   • in *profile*.Configurations.Analytics.token - a token of Analytics Configuration

   • out *analyticsConf* - analytics configuration

14. If *analyticsConf*[0].@token != *profile*.Configurations.Analytics.token, FAIL the test.

15. If *analyticsConf*[0].AnalyticsEngineConfiguration does not contain AnalyticsModule element with @Type = *analyticsModuleDescription*.@Name and with @Name = *moduleName*, FAIL the test.

16. Set *createdAnalyticsModule2* := AnalyticsModule with @Type = *analyticsModuleDescription*.@Name and with @Name = *moduleName* from *analyticsConf*[0].AnalyticsEngineConfiguration.

17. If list of SimpleItem.@Name in *createdAnalyticsModule2* does not equal to list of SimpleItem.@Name in *createdAnalyticsModule1*, FAIL the test.

18. For each SimpleItem element *simpleItem* in *createdAnalyticsModule2*.Parameters repeat the following steps:

18.1. If *simpleItem*.@Value != @Value of corresponding SimpleItem in *createdAnalyticsModule1* (@Name will be used as key), FAIL the test.

19. If list of ElementItem.@Name in *createdAnalyticsModule2* does not equal to list of ElementItem.@Name in *createdAnalyticsModule1*, FAIL the test.

20. ONVIF Client deletes created analytics module to restore the DUT by following the procedure mentioned in Annex A.19 with the following input and output parameters

   • in *profile*.Configurations.Analytics.token - token of Analytics configuration

   • in *moduleName*.@Name - Name of the analytics module to be deleted

21. ONVIF Client retores *moduleToRestore*.

22. ONVIF Client restores media profile if it was changed at step 3.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **CreateAnalyticsModulesResponse** messages.

5.4.6 DELETE ANALYTICS MODULES

Test Case ID: ANALYTICS-4-1-6

Specification Coverage: DeleteAnalyticsModules (ONVIF Analytics Service Spec)

Feature under test: DeleteAnalyticsModules (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify delete of analytics module using DeleteAnalyticsModules request.

Pre-Requisite: Analytics Service was received from the DUT. Media2 Service was received from the DUT. Analytics Modules is supported by the DUT. Profile M is supported.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client configures device for adding of new analytics module by following the procedure mentioned in Annex A.23 with the following input and output parameters

   - out *nonFixedModuleSupported* - if non fixed Analytics Module is supported

   - out (optional) *analyticsToken* - a token of Analytics Configuration

   - out (optional) *analyticsModuleDescription* - analytics module description

   - out *moduleToRestore* (optional) - analytics module to restore

4. If *nonFixedModuleSupported* = false, skip other steps with PASS result.

5. Set *moduleName* := generated random string.

6. ONVIF Client creates new analytics module by following the procedure mentioned in Annex A.22 with the following input and output parameters

    • in *profile*.Configurations.Analytics.token - a token of Analytics Configuration

    • in *moduleName* - analytics module name

    • in *analyticsModuleDescription*.@Name - analytics module type

7. ONVIF Client invokes **DeleteAnalyticsModules** request with parameters

    • ConfigurationToken := *profile*.Configurations.Analytics.token

    • AnalyticsModule[0].@Name := *moduleName*

8. DUT responds with **DeleteAnalyticsModulesResponse** message

9. ONVIF Client retrieves a set of analytics modules assigned of Analytics Configuration by following the procedure mentioned in Annex A.20 with the following input and output parameters

    • in *profile*.Configurations.Analytics.token - a token of Analytics Configuration

    • out *analyticsModuleList* - analytics module list

10. If *analyticsModuleList* contains AnalyticsModule with @Type = *analyticsModuleDescription*.@Name and with @Name = *moduleName*, FAIL the test.

11. ONVIF Client retrieves an Analytics Configuration by following the procedure mentioned in Annex A.21 with the following input and output parameters

    • in *profile*.Configurations.Analytics.token - a token of Analytics Configuration

    • out *analyticsConf* - analytics configuration

12. If *analyticsConf*[0].@token != *profile*.Configurations.Analytics.token, FAIL the test.

13. If *analyticsConf*[0].AnalyticsEngineConfiguration contains AnalyticsModule element with @Type = *analyticsModuleDescription*.@Name and with @Name = *moduleName*, FAIL the test.

14. ONVIF Client retores *moduleToRestore*.

15. ONVIF Client restores media profile if it was changed at step 3.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **DeleteAnalyticsModulesResponse** messages.

5.4.7 MODIFY ANALYTICS MODULES (PARAMETERS WITH OPTIONS)

Test Case ID: ANALYTICS-4-1-7

Specification Coverage: ModifyAnalyticsModules (ONVIF Analytics Service Spec)

Feature under test: ModifyAnalyticsModules (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify modification of analytics module's parameters that have corresponding options using ModifyAnalyticsModules request.

Pre-Requisite: Analytics Service was received from the DUT. Media2 Service was received from the DUT. Analytics Modules is supported by the DUT. Analytics Module Options is supported by the DUT. Profile M is supported.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client selects analytics module description that has parameters with options by following the procedure mentioned in Annex A.38 with the following input and output parameters

   - in true - flag shows if parameters of selected Analytics Module have corresponding options

   - out (optional) *analyticsToken* - a token of Analytics Configuration

   - out (optional) *analyticsModuleDescription* - analytics module description

   - out (optional) *itemListDescription* - list of SimpleItemDescription/ElementItemDescription elements that have options

- out (optional) *limit* - maximum number of concurrent modules per configuration

4. If Annex A.38 procedure did not return *analyticsModuleDescription*, skip other steps with PASS result.

5. ONVIF Client selects or create analytics module with required type by following the procedure mentioned in Annex A.40 with the following input and output parameters

  - in *analyticsToken* - a token of Analytics Configuration

  - in *analyticsModuleDescription* - analytics module description

  - out *analyticsModule1* - Analytics Module with required type

6. For for the first SimpleItemDescription/ElementItemDescription (*itemDescription*) in *itemListDescription*

  6.1.   ONVIF Client invokes **ModifyAnalyticsModules** request with parameters

    - ConfigurationToken := *analyticsToken*

    - AnalyticsModule[0].@Name := *analyticsModule1*.@Name

    - AnalyticsModule[0].@Type := *analyticsModule1*.@Type

    - AnalyticsModule[0].Parameters.SimpleItem/ElementItem[@Name = *itemDescription*.@Name].@Value := value of corresponding parameter type with option provided in the tabe in Annex A.37

  6.2.   DUT responds with **ModifyAnalyticsModulesResponse** message

  6.3.   ONVIF Client retrieves a set of analytics modules assigned of Analytics Configuration by following the procedure mentioned in Annex A.20 with the following input and output parameters

    - in *analyticsToken* - a token of Analytics Configuration

    - out *analyticsModuleList* - analytics module list

  6.4.   If *analyticsModuleList* does not contain AnalyticsModule with @Type = *analyticsModule1*.@Type and with @Name = *analyticsModule1*.@Name, FAIL the test.

  6.5.   Set *modifiedAnalyticsModule1* := AnalyticsModule with @Type = *analyticsModule1*.@Type and with @Name = *analyticsModule1*.@Name from *analyticsModuleList*.

6.6.   If *modifiedAnalyticsModule1*.Parameters does not contain SimpleItem/ ElementItem with Name = *itemDescription*.@Name and with value set at step 6.1, FAIL the test and skip other steps.

6.7.   ONVIF Client retrieves an Analytics Configuration by following the procedure mentioned in Annex A.21 with the following input and output parameters

- in *analyticsToken* - a token of Analytics Configuration

- out *analyticsConf* - analytics configuration

6.8.   If *analyticsConf*[0].@token != *analyticsToken*, FAIL the test.

6.9.   If *analyticsConf*[0].AnalyticsEngineConfiguration does not contain AnalyticsModule element with @Type = *analyticsModule1*.@Type and with @Name = *analyticsModule1*.@Name, FAIL the test.

6.10.   Set *modifiedAnalyticsModule2* := AnalyticsModule with @Type = *analyticsModule1*.@Type and with @Name = *analyticsModule1*.@Name from *analyticsConf*[0].AnalyticsEngineConfiguration.

6.11.   If *modifiedAnalyticsModule2*.Parameters does not contain SimpleItem/ ElementItem with Name = *itemDescription*.@Name and with value set at step 6.1, FAIL the test and skip other steps.

7. ONVIF Client deletes analytics module created at step 5 if any to restore the DUT by following the procedure mentioned in Annex A.19 with the following input and output parameters

- in *analyticsToken* - token of Analytics configuration

- in *moduleName*.@Name - Name of the analytics module to be deleted

8. ONVIF Client retores *moduleToRestore* if any.

9. ONVIF Client restores analytics module if it was canged at step 5.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **ModifyAnalyticsModulesResponse** messages.

5.4.8 MODIFY ANALYTICS MODULES (PARAMETERS WITHOUT OPTIONS)

Test Case ID: ANALYTICS-4-1-8

Specification Coverage: ModifyAnalyticsModules (ONVIF Analytics Service Spec)

Feature under test: ModifyAnalyticsModules (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify modification of analytics module's parameters that do not have corresponding options using ModifyAnalyticsModules request.

Pre-Requisite: Analytics Service was received from the DUT. Media2 Service was received from the DUT. Analytics Modules is supported by the DUT. Analytics Module Options is supported by the DUT. Profile M is supported.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client selects analytics module description that has parameters without options by following the procedure mentioned in Annex A.38 with the following input and output parameters

    • in false - flag shows if parameters of selected Analytics Module have corresponding options

    • out (optional) *analyticsToken* - a token of Analytics Configuration

    • out (optional) *analyticsModuleDescription* - analytics module description

    • out (optional) *itemListDescription* - list of SimpleItemDescription/ElementItemDescription elements that do not have options

    • out (optional) *limit* - maximum number of concurrent modules per configuration

4. If Annex A.38 procedure did not return *analyticsModuleDescription*, skip other steps with PASS result.

5. ONVIF Client selects or create analytics module with required type by following the procedure mentioned in Annex A.40 with the following input and output parameters

• in *analyticsToken* - a token of Analytics Configuration

• in *analyticsModuleDescription* - analytics module description

• out *analyticsModule1* - Analytics Module with required type

6. For the first SimpleItemDescription/ElementItemDescription (*itemDescription*) in *itemListDescription*

　　6.1.　ONVIF Client invokes **ModifyAnalyticsModules** request with parameters

　　　　• ConfigurationToken := *analyticsToken*

　　　　• AnalyticsModule[0].@Name := *analyticsModule1*.@Name

　　　　• AnalyticsModule[0].@Type := *analyticsModule1*.@Type

　　　　• AnalyticsModule[0].Parameters.SimpleItem/ElementItem[@Name = *itemDescription*.@Name].@Value := value of corresponding parameter type with skipped option provided in the tabe in Annex A.37

　　6.2.　DUT responds with **ModifyAnalyticsModulesResponse** message

　　6.3.　ONVIF Client retrieves a set of analytics modules assigned of Analytics Configuration by following the procedure mentioned in Annex A.20 with the following input and output parameters

　　　　• in *analyticsToken* - a token of Analytics Configuration

　　　　• out *analyticsModuleList* - analytics module list

　　6.4.　If *analyticsModuleList* does not contain AnalyticsModule with @Type = *analyticsModule1*.@Type and with @Name = *analyticsModule1*.@Name, FAIL the test.

　　6.5.　Set *modifiedAnalyticsModule1* := AnalyticsModule with @Type = *analyticsModule1*.@Type and with @Name = *analyticsModule1*.@Name from *analyticsModuleList*.

　　6.6.　If *modifiedAnalyticsModule1*.Parameters does not contain SimpleItem/ ElementItem with Name = *itemDescription*.@Name and with value set at step 6.1, FAIL the test and skip other steps.

　　6.7.　ONVIF Client retrieves an Analytics Configuration by following the procedure mentioned in Annex A.21 with the following input and output parameters

- in *analyticsToken* - a token of Analytics Configuration

- out *analyticsConf* - analytics configuration

6.8.   If *analyticsConf*[0].@token != *analyticsToken*, FAIL the test.

6.9.   If   *analyticsConf*[0].AnalyticsEngineConfiguration   does   not   contain AnalyticsModule element with @Type = *analyticsModule1*.@Type and with @Name = *analyticsModule1*.@Name, FAIL the test.

6.10.   Set   *modifiedAnalyticsModule2*   :=   AnalyticsModule   with   @Type   = *analyticsModule1*.@Type and with @Name = *analyticsModule1*.@Name from *analyticsConf*[0].AnalyticsEngineConfiguration.

6.11.   If   *modifiedAnalyticsModule2*.Parameters   does   not   contain   SimpleItem/ ElementItem with Name = *itemDescription*.@Name and with value set at step 6.1, FAIL the test and skip other steps.

7. ONVIF Client deletes analytics module created at step 5 if any to restore the DUT by following the procedure mentioned in Annex A.19 with the following input and output parameters

- in *analyticsToken* - token of Analytics configuration

- in *moduleName*.@Name - Name of the analytics module to be deleted

8. ONVIF Client retores *moduleToRestore* if any.

9. ONVIF Client restores analytics module if it was selected at step 5.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **ModifyAnalyticsModulesResponse** messages.

5.5 Scene Elements

5.5.1 OBJECT CLASSIFICATION METADATA

Test Case ID: ANALYTICS-5-1-1

Specification Coverage: Object Class descriptor (ONVIF Analytics Service Spec)

Feature under test: GetSupportedMetadata (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify Object Class descriptor.

Pre-Requisite: Analytics Service was received from the DUT. Object Classification feature is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of supported metadata by following the procedure mentioned in Annex A.14 with the following input and output parameters

    • out *analyticsModuleList* - Analytics Module List with Metadata Info

4. For each Object *object* from *analyticsModule*.SampleFrame.Object list repeat the following steps:

    4.1.   If *object* contains Appearance.Class:

        4.1.1.   For each Type in Appearance.Class:

            4.1.1.1. If Type.@Likelihood > 1, log WARNING message

        4.1.2.   PASS the test.

Test Result:

PASS –

    • DUT passes all assertions.

FAIL –

    • None.

Note: At step 4.1.1.1 if Appearance.Class.Type element does not contain @Likelihood, the ONVIF Client assumes it as 1.

5.5.2 VEHICLE INFORMATION DESCRIPTOR

Test Case ID: ANALYTICS-5-1-2

Specification Coverage: Vehicle information descriptor (ONVIF Analytics Service Spec)

Feature under test: GetSupportedMetadata (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify Vehicle information descriptor.

Pre-Requisite: Analytics Service was received from the DUT. Vehicle Info feature is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of supported metadata by following the procedure mentioned in Annex A.14 with the following input and output parameters

    • out *analyticsModuleList* - Analytics Module List with Metadata Info

4. If *analyticsModuleList* does not contain at least one AnalyticsModule that contains SampleFrame.Object.Appearance with VehicleInfo element, FAIL the test and skip other steps.

Test Result:

PASS –

• DUT passes all assertions.

FAIL –

• None.

5.5.3 GEO LOCATION METADATA

Test Case ID: ANALYTICS-5-1-3

Specification Coverage: GeoLocation metadata (ONVIF Profile M Spec)

Feature under test: GetSupportedMetadata (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify GeoLocation metadata.

Pre-Requisite: Analytics Service was received from the DUT. Geo Location feature is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of supported metadata by following the procedure mentioned in Annex A.14 with the following input and output parameters

   • out *analyticsModuleList* - Analytics Module List with Metadata Info

4. If *analyticsModuleList* does not contain at least one AnalyticsModule with SampleFrame.Object. Appearance.GeoLocation element, FAIL the test and skip other steps.

Test Result:

PASS –

   • DUT passes all assertions.

FAIL –

   • None.

5.5.4 HUMAN FACE DESCRIPTOR

Test Case ID: ANALYTICS-5-1-4

Specification Coverage: Face descriptor (ONVIF Analytics Service Spec)

Feature under test: GetSupportedMetadata (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify Human Face descriptor.

Pre-Requisite: Analytics Service was received from the DUT. Human Face feature is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of supported metadata by following the procedure mentioned in Annex A.14 with the following input and output parameters

   • out *analyticsModuleList* - Analytics Module List with Metadata Info

4. If *analyticsModuleList* does not contain at least one AnalyticsModule with SampleFrame.Object.Appearance with HumanFace element, FAIL the test and skip other steps.

Test Result:

PASS –

• DUT passes all assertions.

FAIL –

• None.

5.5.5 LICENSE PLATE INFORMATION DESCRIPTOR

Test Case ID: ANALYTICS-5-1-5

Specification Coverage: License plate information descriptor (ONVIF Analytics Service Spec)

Feature under test: GetSupportedMetadata (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify License Plate Information descriptor.

Pre-Requisite: Analytics Service was received from the DUT. License Plate Info feature is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of supported metadata by following the procedure mentioned in Annex A.14 with the following input and output parameters

   • out *analyticsModuleList* - Analytics Module List with Metadata Info

4. If *analyticsModuleList* does not contain at least one AnalyticsModule with SampleFrame.Object.Appearance with LicensePlateInfo element, FAIL the test and skip other steps.

Test Result:

PASS –

   • DUT passes all assertions.

FAIL –

   • None.

5.5.6 HUMAN BODY DESCRIPTOR

Test Case ID: ANALYTICS-5-1-6

Specification Coverage: Human body descriptor (ONVIF Analytics Service Spec)

Feature under test: GetSupportedMetadata (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify Human Body descriptor.

Pre-Requisite: Analytics Service was received from the DUT. Human Body feature is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of supported metadata by following the procedure mentioned in Annex A.14 with the following input and output parameters

> • out *analyticsModuleList* - Analytics Module List with Metadata Info

4. If *analyticsModuleList* does not contain at least one AnalyticsModule with SampleFrame.Object.Appearance with HumanBody element, FAIL the test and skip other steps.

Test Result:

PASS –

• DUT passes all assertions.

FAIL –

• None.

5.5.7 IMAGE DATA

Test Case ID: ANALYTICS-5-1-7

Specification Coverage: Image Data (ONVIF Analytics Service Spec)

Feature under test: GetSupportedMetadata (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify Image Data metadata.

Pre-Requisite: Analytics Service was received from the DUT. Image Sending feature is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of supported metadata by following the procedure mentioned in Annex A.14 with the following input and output parameters

   • out *analyticsModuleList* - Analytics Module List with Metadata Info

4. If the DUT supports Local Storage Image Sending Type feature or Remote Storage Image Sending Type feature:

4.1. If *analyticsModuleList* does not contain at least one AnalyticsModule with SampleFrame.Object. Appearance.ImageRef element, FAIL the test and skip other steps.

5. If the DUT supports Embedded Image Sending Type feature:

5.1. If *analyticsModuleList* does not contain at least one AnalyticsModule with SampleFrame.Object. Appearance.Image element, FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- None.

5.6 Rules

5.6.1 GET SUPPORTED RULES

Test Case ID: ANALYTICS-6-1-1

Specification Coverage: GetSupportedRules (ONVIF Analytics Service Spec)

Feature under test: GetSupportedRules (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify getting supported rules using GetSupportedRules request.

Pre-Requisite: Analytics Service was received from the DUT. Media2 Service was received from the DUT. Rule Engine is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

- out *analyticsConfList* - a list of Analytics configurations

4. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

    4.1. ONVIF Client invokes **GetSupportedRules** request with parameters

       • ConfigurationToken := *analyticsConf*.@token

    4.2. DUT responds with **GetSupportedRulesResponse** message with parameters

       • SupportedRules =: *supportedRules*

    4.3. If *supportedRules* contains at least two RuleDescription items with the same @Name value, FAIL the test and skip other steps.

    4.4. For each Rule Description *ruleDescription* in *supportedRules*.RuleDescription list repeat the following steps:

       4.4.1. If *ruleDescription* does not have maxInstances attribute, FAIL the test, restore the DUT state, and skip other steps.

       4.4.2. If at least one item in *ruleDescription*.Parameters list (SimpleItemDescription or ElementItemDescription item) has the same Name value with other item from the same list, FAIL the test, restore the DUT state, and skip other steps.

       4.4.3. If at least one item in *ruleDescription*.Messages list (Source, or Key, or Data; SimpleItemDescription or ElementItemDescription item) has the same Name value with other item from the same list, FAIL the test, restore the DUT state, and skip other steps.

       4.4.4. If *ruleDescription*.ParentTopic is not valid topic (see Annex A.11), FAIL the test, restore the DUT state, and skip other steps.

       4.4.5. If *ruleDescription*.fixed = true:

          4.4.5.1. If *analyticsConf* does not contain at least one RuleEngineConfiguration.Rule with Type = *ruleDescription*.Name, FAIL the test, restore the DUT state, and skip other steps.

Test Result:

PASS –

• DUT passes all assertions.

FAIL –

• The DUT did not send **GetSupportedRulesResponse** messages.

5.6.2 GET RULE OPTIONS

Test Case ID: ANALYTICS-6-1-2

Specification Coverage: GetRuleOptions (ONVIF Analytics Service Spec)

Feature under test: GetRuleOptions (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify getting rule options using GetRuleOptions request.

Pre-Requisite: Analytics Service was received from the DUT. Media2 Service was received from the DUT. Rule Engine is supported by the DUT. Rule Options is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

   • out *analyticsConfList* - a list of Analytics configurations

4. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

   4.1. ONVIF Client invokes **GetSupportedRules** request with parameters

      • ConfigurationToken := *analyticsConf*.@token

   4.2. DUT responds with **GetSupportedRulesResponse** message with parameters

      • SupportedRules =: *supportedRules*

   4.3. ONVIF Client invokes **GetRuleOptions** request with parameters

      • RuleType is skipped

      • ConfigurationToken := *analyticsConf*.@token

   4.4. DUT responds with **GetRuleOptionsResponse** message with parameters

      • RuleOptions list =: *ruleOptionsFullList*

4.5.   If at least one item in *ruleOptionsFullList* list contains AnalyticsModule, FAIL the test, restore the DUT state, and skip other steps.

4.6.   For each Rule Description *ruleDescription* in *supportedRules*.RuleDescription list repeat the following steps:

   4.6.1.   ONVIF Client invokes **GetRuleOptions** request with parameters

      • RuleType := *ruleDescription*.Name

      • ConfigurationToken := *analyticsConf*.@token

   4.6.2.   DUT responds with **GetRuleOptionsResponse** message with parameters

      • RuleOptions list =: *ruleOptionsList*

   4.6.3.   If at least one item in *ruleOptionsList* list contains AnalyticsModule, FAIL the test, restore the DUT state, and skip other steps.

   4.6.4.   If at least one item in *ruleOptionsList* list contains RuleType which is not equal to *ruleDescription*.Name, FAIL the test, restore the DUT state, and skip other steps.

   4.6.5.   If *ruleOptionsFullList* list does not contain all items from *ruleOptionsList* list (see NOTE at the end of the test case), FAIL the test, restore the DUT state, and skip other steps.

Test Result:

PASS –

   • DUT passes all assertions.

FAIL –

   • The DUT did not send **GetSupportedRulesResponse** messages.

   • The DUT did not send **GetRuleOptionsResponse** messages.

NOTE: At step 4.6.5 RuleType and Name to be used as unique identifier. If RuleType is skipped in full list, the ONVIF Client assumes that this option is compatible with each supported RuleType that supports parameter with such Name.

5.6.3 GET RULES

Test Case ID: ANALYTICS-6-1-3

Specification Coverage: GetRules (ONVIF Analytics Service Spec)

Feature under test: GetRules (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify getting rules using GetRules request.

Pre-Requisite: Analytics Service was received from the DUT. Media2 Service was received from the DUT. Rule Engine is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

    • out *analyticsConfList* - a list of Analytics configurations

4. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

    4.1.   ONVIF Client invokes **GetSupportedRules** request with parameters

        • ConfigurationToken := *analyticsConf*.@token

    4.2.   DUT responds with **GetSupportedRulesResponse** message with parameters

        • SupportedRules =: *supportedRules*

    4.3.   ONVIF Client invokes **GetRules** request with parameters

        • ConfigurationToken := *analyticsConf*.@token

    4.4.   DUT responds with **GetRulesResponse** message with parameters

        • Rule list =: *ruleList*

    4.5.   If at least one item in *ruleList* list contains Type which in not equal to Name fileld of at least one item at *supportedRules*.RuleDescription list, FAIL the test, restore the DUT state, and skip other steps.

    4.6.   For each Rule Description *ruleDescription* in *supportedRules*.RuleDescription list repeat the following steps:

4.6.1. If *ruleDescription*.fixed = true:

      4.6.1.1. If *ruleList* does not contain item with Type = *ruleDescription*.Name, FAIL the test, restore the DUT state, and skip other steps.

4.6.2. If *ruleList* contains item with Type = *ruleDescription*.Name:

      4.6.2.1. Set *rule* := *ruleList*[Type = *ruleDescription*.Name].

      4.6.2.2. If *rule*.Parameters.SimpleItem contains at least one SimpleItem which does not have item with the same Name in *ruleDescription*.Parameters.SimpleItemDescription list, FAIL the test, restore the DUT state, and skip other steps.

      4.6.2.3. If *rule*.Parameters.ElementItem contains at least one ElementItem which does not have item with the same Name in *ruleDescription*.Parameters.ElementItemDescription list, FAIL the test, restore the DUT state, and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **GetSupportedRulesResponse** messages.

- The DUT did not send **GetRulesResponse** messages.

5.6.4 CREATE RULES

Test Case ID: ANALYTICS-6-1-4

Specification Coverage: CreateRules (ONVIF Analytics Service Spec)

Feature under test: CreateRules (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify rule creation with empty parameters list using CreateRules request.

Pre-Requisite: Analytics Service was received from the DUT. Media2 Service was received from the DUT. Rule Engine is supported by the DUT. Profile M is supported.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client configures device for adding of new rule by following the procedure mentioned in Annex A.27 with the following input and output parameters

    • out *nonFixedRuleSupported* - indication of non-fixed Rule support

    • out *analyticsToken* - a token of Analytics Configuration

    • out *ruleDescription* - a rule description

    • out *ruleToRestore* - a rule to restore, if it was removed

4. If *nonFixedRuleSupported* = false, skip other steps with PASS result, restore the DUT state.

5. Set *ruleName* := generated random string.

6. ONVIF Client invokes **CreateRules** request with parameters

    • ConfigurationToken := *analyticsToken*

    • Rule[0].@Name := *ruleName*

    • Rule[0].@Type := *ruleDescription*.@Name

    • Rule[0].Parameters := empty Parameters element

7. DUT responds with **CreateRulesResponse** message

8. ONVIF Client retrieves a set of rules assigned to Analytics Configuration by following the procedure mentioned in Annex A.5 with the following input and output parameters

    • in *analyticsToken* - a token of Analytics Configuration

    • out *ruleList* - rule list

9. If *ruleList* does not contain Rule with @Type = *ruleDescription*.@Name and with @Name = *ruleName*, FAIL the test, restore the DUT state, and skip other steps.

10. Set *createdRule1* := Rule with @Type = *ruleDescription*.@Name and with @Name = *ruleName* from *ruleList*.

11. For each SimpleItem element *simpleItem* in *createdRule1*.Parameters repeat the following steps:

11.1. If *supportedRule*.RuleDescription[0].Parameters does not contain at least one SimpleItemDescription with @Name = *simpleItem*.@Name, FAIL the test, restore the DUT state, and skip other steps.

12. For each ElementItem element *elementItem* in *createdRule1*.Parameters repeat the following steps:

12.1. If *supportedRule*.RuleDescription[0].Parameters does not contain at least one ElementItemDescription with @Name = *elementItem*.@Name, FAIL the test, restore the DUT state, and skip other steps.

13. ONVIF Client retrieves an Analytics Configuration by following the procedure mentioned in Annex A.21 with the following input and output parameters

   • in *analyticsToken* - a token of Analytics Configuration

   • out *analyticsConf* - analytics configuration

14. If *analyticsConf*[0].@token != *analyticsToken*, FAIL the test, restore the DUT state, and skip other steps.

15. If *analyticsConf*[0].RuleEngineConfiguration does not contain Rule element with @Type = *ruleDescription*.@Name and with @Name = *ruleName*, FAIL the test, restore the DUT state, and skip other steps.

16. Set *createdRule2* := Rule with @Type = *ruleDescription*.@Name and with @Name = *ruleName* from *analyticsConf*[0].RuleEngineConfiguration.

17. If list of SimpleItem.@Name in *createdRule2* does not equal to list of SimpleItem.@Name in *createdRule1*, FAIL the test, restore the DUT state, and skip other steps.

18. For each SimpleItem element *simpleItem* in *createdRule2*.Parameters repeat the following steps:

18.1. If *simpleItem*.@Value != @Value of corresponding SimpleItem in *createdRule1* (@Name will be used as key), FAIL the test, restore the DUT state, and skip other steps.

19. If list of ElementItem.@Name in *createdRule2* does not equal to list of ElementItem.@Name in *createdRule1*, FAIL the test, restore the DUT state, and skip other steps.

20. ONVIF Client restores the DUT state.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **CreateRulesResponse** messages.

5.6.5 DELETE RULES

Test Case ID: ANALYTICS-6-1-5

Specification Coverage: DeleteRules (ONVIF Analytics Service Spec)

Feature under test: DeleteRules (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify rule deletion using DeleteRules request.

Pre-Requisite: Analytics Service was received from the DUT. Media2 Service was received from the DUT. Rule Engine is supported by the DUT. Profile M is supported.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client configures device for adding of new rule by following the procedure mentioned in Annex A.27 with the following input and output parameters

    - out *nonFixedRuleSupported* - indication of non-fixed Rule support

    - out *analyticsToken* - a token of Analytics Configuration

    - out *ruleDescription* - a rule description

    - out *ruleToRestore* - a rule to restore, if it was removed

4. If *nonFixedRuleSupported* = false, skip other steps with PASS result, restore the DUT state.

5. Set *ruleName* := generated random string.

6. ONVIF Client creates new rule by following the procedure mentioned in Annex A.32 with the following input and output parameters

    - in *analyticsToken* - a token of Analytics Configuration

- in *ruleName* - rule name

- in *ruleDescription*.@Name - rule type

7. ONVIF Client invokes **DeleteRules** request with parameters

- ConfigurationToken := *analyticsToken*

- RuleName := *ruleName*

8. DUT responds with **DeleteRulesResponse** message

9. ONVIF Client retrieves a set of rules assigned of Analytics Configuration by following the procedure mentioned in Annex A.5 with the following input and output parameters

- in *analyticsToken* - a token of Analytics Configuration

- out *ruleList* - rule list

10. If *ruleList* contains Rule with @Type = *ruleDescription*.@Name and with @Name = *ruleName*, FAIL the test, restore the DUT state, and skip other steps.

11. ONVIF Client retrieves an Analytics Configuration by following the procedure mentioned in Annex A.21 with the following input and output parameters

- in *analyticsToken* - a token of Analytics Configuration

- out *analyticsConf* - analytics configuration

12. If *analyticsConf*[0].@token != *analyticsToken*, FAIL the test, restore the DUT state, and skip other steps.

13. If *analyticsConf*[0].RuleEngineConfiguration contains Rule element with @Type = *ruleDescription*.@Name and with @Name = *ruleName*, FAIL the test, restore the DUT state, and skip other steps.

14. ONVIF Client restores the DUT state.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **DeleteRulesResponse** messages.

5.6.6 MODIFY RULES (PARAMETERS WITH OPTIONS)

Test Case ID: ANALYTICS-6-1-6

Specification Coverage: ModifyRules (ONVIF Analytics Service Spec)

Feature under test: ModifyRules (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify modification of rule's parameters that have corresponding options using ModifyRules request.

Pre-Requisite: Analytics Service was received from the DUT. Media2 Service was received from the DUT. Rules is supported by the DUT. Rule Options is supported by the DUT. Profile M is supported.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client selects rule description that has parameters with options by following the procedure mentioned in Annex A.41 with the following input and output parameters

    • in true - flag shows if parameters of selected Rule have corresponding options

    • out (optional) *analyticsToken* - a token of Analytics Configuration

    • out (optional) *ruleDescription* - rule description

    • out (optional) *itemListDescription* - list of SimpleItemDescription/ElementItemDescription elements that have options

    • out (optional) *limit* - maximum number of concurrent rules per configuration

4. If Annex A.41 procedure did not return *ruleDescription*, skip other steps with PASS result.

5. ONVIF Client selects or create rule with required type by following the procedure mentioned in Annex A.42 with the following input and output parameters

    • in *analyticsToken* - a token of Analytics Configuration

    • in *ruleDescription* - rule description

- out *rule1* - Rule with required type

6. For the first SimpleItemDescription/ElementItemDescription (*itemDescription*) in *itemListDescription*

    6.1.   ONVIF Client invokes **ModifyRules** request with parameters

- ConfigurationToken := *analyticsToken*

- Rule[0].@Name := *rule1*.@Name

- Rule[0].@Type := *rule1*.@Type

- Rule[0].Parameters.SimpleItem/ElementItem[@Name = *itemDescription*.@Name].@Value := value of corresponding parameter type with option provided in the tabe in Annex A.37

    6.2.   DUT responds with **ModifyRulesResponse** message

    6.3.   ONVIF Client retrieves a set of rules assigned of Analytics Configuration by following the procedure mentioned in Annex A.5 with the following input and output parameters

- in *analyticsToken* - a token of Analytics Configuration

- out *ruleList* - rule list

    6.4.   If *ruleList* does not contain Rule with @Type = *rule1*.@Type and with @Name = *rule1*.@Name, FAIL the test.

    6.5.   Set *modifiedRule1* := Rule with @Type = *rule1*.@Type and with @Name = *rule1*.@Name from *ruleList*.

    6.6.   If *modifiedRule1*.Parameters does not contain SimpleItem/ElementItem with Name = *itemDescription*.@Name and with value set at step 6.1, FAIL the test and skip other steps.

    6.7.   ONVIF Client retrieves an Analytics Configuration by following the procedure mentioned in Annex A.21 with the following input and output parameters

- in *analyticsToken* - a token of Analytics Configuration

- out *analyticsConf* - analytics configuration

    6.8.   If *analyticsConf*[0].@token != *analyticsToken*, FAIL the test.

6.9.   If *analyticsConf*[0].AnalyticsEngineConfiguration does not contain Rule element with @Type = *rule1*.@Type and with @Name = *rule1*.@Name, FAIL the test.

6.10.   Set *modifiedRule2* := Rule with @Type = *rule1*.@Type and with @Name = *rule1*.@Name from *analyticsConf*[0].AnalyticsEngineConfiguration.

6.11.   If *modifiedRule2*.Parameters does not contain SimpleItem/ElementItem with Name = *itemDescription*.@Name and with value set at step 6.1, FAIL the test and skip other steps.

7. ONVIF Client deletes rule created at step 5 if any to restore the DUT by following the procedure mentioned in Annex A.31 with the following input and output parameters

- in *analyticsToken* - token of Analytics configuration

- in *ruleName*.@Name - Name of the rule to be deleted

8. ONVIF Client retores *ruleToRestore* if any.

9. ONVIF Client restores rule if it was canged at step 5.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **ModifyRulesResponse** messages.

5.6.7 MODIFY RULES (PARAMETERS WITHOUT OPTIONS)

Test Case ID: ANALYTICS-6-1-7

Specification Coverage: ModifyRules (ONVIF Analytics Service Spec)

Feature under test: ModifyRules (Analytics)

WSDL Reference: analytics.wsdl

Test Purpose: To verify modification of rule's parameters that do not have corresponding options using ModifyRules request.

Pre-Requisite: Analytics Service was received from the DUT. Media2 Service was received from the DUT. Rules is supported by the DUT. Rule Options is supported by the DUT. Profile M is supported.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client selects rule description that has parameters without options by following the procedure mentioned in Annex A.41 with the following input and output parameters

   - in false - flag shows if parameters of selected Rule have corresponding options

   - out (optional) *analyticsToken* - a token of Analytics Configuration

   - out (optional) *ruleDescription* - rule description

   - out (optional) *itemListDescription* - list of SimpleItemDescription/ElementItemDescription elements that do not have options

   - out (optional) *limit* - maximum number of concurrent rules per configuration

4. If Annex A.41 procedure did not return *ruleDescription*, skip other steps with PASS result.

5. ONVIF Client selects or create rule with required type by following the procedure mentioned in Annex A.42 with the following input and output parameters

   - in *analyticsToken* - a token of Analytics Configuration

   - in *ruleDescription* - rule description

   - out *rule1* - Rule with required type

6. For the first SimpleItemDescription/ElementItemDescription (*itemDescription*) in *itemListDescription*

   6.1.   ONVIF Client invokes **ModifyRules** request with parameters

   - ConfigurationToken := *analyticsToken*

   - Rule[0].@Name := *rule1*.@Name

   - Rule[0].@Type := *rule1*.@Type

   - Rule[0].Parameters.SimpleItem/ElementItem[@Name = *itemDescription*.@Name].@Value := value of corresponding parameter type with skipped option provided in the tabe in Annex A.37

   6.2.   DUT responds with **ModifyRulesResponse** message

6.3.   ONVIF Client retrieves a set of rules assigned of Analytics Configuration by following the procedure mentioned in Annex A.5 with the following input and output parameters

- in *analyticsToken* - a token of Analytics Configuration

- out *ruleList* - rule list

6.4.   If *ruleList* does not contain Rule with @Type = *rule1*.@Type and with @Name = *rule1*.@Name, FAIL the test.

6.5.   Set *modifiedRule1* := Rule with @Type = *rule1*.@Type and with @Name = *rule1*.@Name from *ruleList*.

6.6.   If *modifiedRule1*.Parameters does not contain SimpleItem/ElementItem with Name = *itemDescription*.@Name and with value set at step 6.1, FAIL the test and skip other steps.

6.7.   ONVIF Client retrieves an Analytics Configuration by following the procedure mentioned in Annex A.21 with the following input and output parameters

- in *analyticsToken* - a token of Analytics Configuration

- out *analyticsConf* - analytics configuration

6.8.   If *analyticsConf*[0].@token != *analyticsToken*, FAIL the test.

6.9.   If *analyticsConf*[0].AnalyticsEngineConfiguration does not contain Rule element with @Type = *rule1*.@Type and with @Name = *rule1*.@Name, FAIL the test.

6.10.   Set *modifiedRule2* := Rule with @Type = *rule1*.@Type and with @Name = *rule1*.@Name from *analyticsConf*[0].AnalyticsEngineConfiguration.

6.11.   If *modifiedRule2*.Parameters does not contain SimpleItem/ElementItem with Name = *itemDescription*.@Name and with value set at step 6.1, FAIL the test and skip other steps.

7. ONVIF Client deletes rule created at step 5 if any to restore the DUT by following the procedure mentioned in Annex A.31 with the following input and output parameters

- in *analyticsToken* - token of Analytics configuration

- in *ruleName*.@Name - Name of the rule to be deleted

8. ONVIF Client retores *ruleToRestore* if any.

9. ONVIF Client restores rule if it was canged at step 5.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **ModifyRulesResponse** messages.

5.7 Face Recognition Rule

5.7.1 GET SUPPORTED RULES (FACE RECOGNITION)

Test Case ID: ANALYTICS-7-1-1

Specification Coverage: Get Supported rules (ONVIF Analytics Service Spec), Recognition rule engines (ONVIF Analytics Service Spec), Face Recognition (ONVIF Analytics Service Spec)

Feature Under Test: GetSupportedRules, RuleDescription for tt:FaceRecognition

WSDL Reference: analytics.wsdl, media2.wsdl

Test Purpose: To verify that device includes tt:FaceRecognition in GetSupportedRulesResponse. To verify structure of Face Recognition.

Pre-Requisite: Analytics Service is received from the DUT. Media2 Service is received from the DUT. Rule Engine is supported by the Device as indicated by the RuleSupport capabilities. Face Recognition Rule is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

   - out *analyticsConfList* - a list of Analytics configurations

4. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

   4.1. ONVIF Client invokes **GetSupportedRules** request with parameters

• ConfigurationToken := *analyticsConf*.@token

4.2. DUT responds with **GetSupportedRulesResponse** message with parameters

• SupportedRules =: *supportedRules*

4.3. If *supportedRules* contains RuleDescription element (*faceRecognitionRuleDescription*) with Name value is equal to **tt:FaceRecognition**:

4.3.1. If *faceRecognitionRuleDescription* does not have maxInstances attribute, FAIL the test and skip other steps.

4.3.2. If *faceRecognitionRuleDescription*.Parameters has SimpleItemDescription element with @Name = "IncludeImage" and its @Type value != xs:string, FAIL the test and skip other steps.

4.3.3. ONVIF Client verifies rule engine message format by following the procedure mentioned in Annex A.46 with the following input and output parameters

• in *faceRecognitionRuleDescription*.Messages - message description to be verified

4.3.4. If *faceRecognitionRuleDescription* does not have Messages.Data.SimpleItemDescription element with @Name = "Likelihood", FAIL the test and skip other steps.

4.3.5. If @Type != "xs:float" for *faceRecognitionRuleDescription*.Messages.Data.SimpleItemDescription with @Name = "Likelihood", FAIL the test and skip other steps.

4.3.6. If *faceRecognitionRuleDescription*.Messages.Data has SimpleItemDescription element with @Name = "Label" and its @Type value != xs:string, FAIL the test and skip other steps.

4.3.7. If *faceRecognitionRuleDescription*.Messages.Data has SimpleItemDescription element with @Name = "ImageUri" and its @Type value != xs:anyURI, FAIL the test and skip other steps.

4.3.8. If *faceRecognitionRuleDescription*.Messages.Data has SimpleItemDescription element with @Name = "EnrollmentID" and its @Type value != xs:string, FAIL the test and skip other steps.

4.3.9. If *faceRecognitionRuleDescription*.Messages.Data has SimpleItemDescription element with @Name = "RefImageUri" and its @Type value != xs:anyURI, FAIL the test and skip other steps.

4.3.10. If *faceRecognitionRuleDescription*.Messages.Data has ElementItemDescription element with @Name = "Image" and its @Type value != xs:base64Binary, FAIL the test and skip other steps.

4.3.11. If *faceRecognitionRuleDescription*.Messages.Data has ElementItemDescription element with @Name = "BoundingBox" and its @Type value != tt:Rectangle, FAIL the test and skip other steps.

4.3.12. If Messages.ParentTopic value is not equal to "tns1:RuleEngine/ Recognition/Face" for Messages with Source.SimpleItemDescription.@Name = VideoSource, and with Source.SimpleItemDescription.@Name = Rule, FAIL the test and skip other steps.

5. If there was no RuleDescription element with Name value is equal to **tt:FaceRecognition** in at least one *supportedRules* at step 4.2 [100], FAIL the test.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.7.2 GET FACE RECOGNITION RULE OPTIONS

Test Case ID: ANALYTICS-7-1-2

Specification Coverage: Get Rule Options (ONVIF Analytics Service Spec), Recognition rule engines (ONVIF Analytics Service Spec), Face Recognition (ONVIF Analytics Service Spec), Configuration options (ONVIF Analytics Service Spec).

Feature Under Test: GetRuleOptions, Face Recognition Rule Options

WSDL Reference: analytics.wsdl, media2.wsdl

Test Purpose: To verify retrieving of Face Recognition Options by GetRuleOptions operation.

Pre-Requisite: Analytics Service is received from the DUT. Media2 Service is received from the DUT. Rule Engine is supported by the Device as indicated by the RuleSupport capabilities, Rule

Options is supported by the Device as indicated by the RuleOptionsSupported capabilities. Face Recognition Rule is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of Analytics Configurations that supports tt:FaceRecognition type by following the procedure mentioned in Annex A.2 with the following input and output parameters

   • in **tt:FaceRecognition** - rule type

   • out *analyticsConfList* - a list of Analytics configurations with supporting of Face Recognition

4. If *analyticsConfList* is empty, FAIL the test and skip other steps.

5. For each Analytics Configuration (*analyticsConf*) in *analyticsConfList* repeat the following steps:

   5.1. ONVIF Client retrieves a list of supported rules by following the procedure mentioned in Annex A.29 with the following input and output parameters

      • in *analyticsConf*.@token - a token of Analytics Configuration

      • out *supportedRules* - supported rules.

   5.2. ONVIF Client invokes **GetRuleOptions** request with parameters

      • RuleType := tt:FaceRecognition

      • ConfigurationToken := *analyticsConf*.@token

   5.3. DUT responds with **GetRuleOptionsResponse** message with parameters

      • RuleOptions list =: *ruleOptionsList*

   5.4. If *ruleOptionsList* contains RuleOption with @Name = **IncludeImage** (*includeImageOption*)

      5.4.1. If *includeImageOption* contains @RuleType and @RuleType != "tt:FaceRecognition", FAIL the test and skip other steps.

5.4.2. If *includeImageOption* does not contain **StringItems** element or **StringList** element, FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.8 License Plate Recognition Rule

5.8.1 GET SUPPORTED RULES (LICENSE PLATE RECOGNITION)

Test Case ID: ANALYTICS-8-1-1

Specification Coverage: Get Supported rules (ONVIF Analytics Service Spec), Recognition rule engines (ONVIF Analytics Service Spec), License Plate Recognition (ONVIF Analytics Service Spec)

Feature Under Test: GetSupportedRules, RuleDescription for tt:LicensePlateRecognition

WSDL Reference: analytics.wsdl, media2.wsdl

Test Purpose: To verify that device includes tt:LicensePlateRecognition in GetSupportedRulesResponse. To verify structure of License Plate Recognition.

Pre-Requisite: Analytics Service is received from the DUT. Media2 Service is received from the DUT. Rule Engine is supported by the Device as indicated by the RuleSupport capabilities. License Plate Recognition Rule is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

- out *analyticsConfList* - a list of Analytics configurations

4. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

    4.1. ONVIF Client invokes **GetSupportedRules** request with parameters

- ConfigurationToken := *analyticsConf*.@token

    4.2. DUT responds with **GetSupportedRulesResponse** message with parameters

- SupportedRules =: *supportedRules*

    4.3. If *supportedRules* contains RuleDescription element (*licensePlateRecognitionRuleDescription*) with Name value is equal to **tt:LicensePlateRecognition**:

        4.3.1. If *licensePlateRecognitionRuleDescription* does not have maxInstances attribute, FAIL the test and skip other steps.

        4.3.2. ONVIF Client checks set of items inside Parameters of *licensePlateRecognitionRuleDescription*:

            4.3.2.1. If *licensePlateRecognitionRuleDescription*.Parameters has SimpleItemDescription element with @Name = "IncludeImage" and its @Type value != xs:string, FAIL the test and skip other steps.

            4.3.2.2. If *licensePlateRecognitionRuleDescription*.Parameters has SimpleItemDescription element with @Name = "PlateLocation" and its @Type value != xs:string, FAIL the test and skip other steps.

            4.3.2.3. If *licensePlateRecognitionRuleDescription*.Parameters has ElementItemDescription element with @Name = "Region" and its @Type value != tt:Polygon, FAIL the test and skip other steps.

            4.3.2.4. If *licensePlateRecognitionRuleDescription*.Parameters has ElementItemDescription element with @Name = "SnapLine" and its @Type value != tt:Polyline, FAIL the test and skip other steps.

        4.3.3. ONVIF Client checks set of items inside Messages.Source of *licensePlateRecognitionRuleDescription*:

            4.3.3.1. ONVIF Client verifies rule engine message format by following the procedure mentioned in Annex A.46 with the following input and output parameters

• in *licensePlateRecognitionRuleDescription*.Messages - message description to be verified

4.3.4. ONVIF Client checks set of items inside Messages.Data of *licensePlateRecognitionRuleDescription*:

4.3.4.1. If *licensePlateRecognitionRuleDescription* does not have Messages.Data.SimpleItemDescription element with @Name = "Likelihood", FAIL the test and skip other steps.

4.3.4.2. If @Type != "xs:float" for *licensePlateRecognitionRuleDescription*.Messages.Data.SimpleItemDescription with @Name = "Likelihood", FAIL the test and skip other steps.

4.3.4.3. If *licensePlateRecognitionRuleDescription*.Messages.Data has SimpleItemDescription element with @Name = "Label" and its @Type value != xs:string, FAIL the test and skip other steps.

4.3.4.4. If *licensePlateRecognitionRuleDescription*.Messages.Data has SimpleItemDescription element with @Name = "ImageUri" and its @Type value != xs:anyURI, FAIL the test and skip other steps.

4.3.4.5. If *licensePlateRecognitionRuleDescription*.Messages.Data has SimpleItemDescription element with @Name = "VehicleImageURI" and its @Type value != xs:anyURI, FAIL the test and skip other steps.

4.3.4.6. If *licensePlateRecognitionRuleDescription*.Messages.Data has ElementItemDescription element with @Name = "BoundingBox" and its @Type value != tt:Rectangle, FAIL the test and skip other steps.

4.3.4.7. If *licensePlateRecognitionRuleDescription*.Messages.Data has ElementItemDescription element with @Name = "Image" and its @Type value != xs:base64Binary, FAIL the test and skip other steps.

4.3.4.8. If *licensePlateRecognitionRuleDescription* does not have Messages.Data.ElementItemDescription element with @Name = "LicensePlateInfo", FAIL the test and skip other steps.

4.3.4.9. If @Type != "tt:LicensePlateInfo" for *licensePlateRecognitionRuleDescription*.Messages.Data.ElementItemDescription

with @Name = "LicensePlateInfo", FAIL the test and skip other steps.

4.3.4.10. If *licensePlateRecognitionRuleDescription*.Messages.Data has ElementItemDescription element with @Name = "VehicleInfo" and its @Type value != tt:VehicleInfo, FAIL the test and skip other steps.

4.3.4.11. If *licensePlateRecognitionRuleDescription*.Messages.Data has ElementItemDescription element with @Name = "VehicleImage" and its @Type value != xs:base64Binary, FAIL the test and skip other steps.

4.3.5. If Messages.ParentTopic value is not equal to "tns1:RuleEngine/Recognition/ LicensePlate" for Messages with Source.SimpleItemDescription.@Name = VideoSource, and with Source.SimpleItemDescription.@Name = Rule, FAIL the test and skip other steps.

5. If there was no RuleDescription element with Name value is equal to **tt:LicensePlateRecognition** for at least one *supportedRules* at step 4.3, FAIL the test.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.8.2 GET LICENSE PLATE RECOGNITION RULE OPTIONS

Test Case ID: ANALYTICS-8-1-2

Specification Coverage: Get Rule Options (ONVIF Analytics Service Spec), Recognition rule engines (ONVIF Analytics Service Spec), License Plate Recognition (ONVIF Analytics Service Spec), Configuration options (ONVIF Analytics Service Spec).

Feature Under Test: GetRuleOptions, MotionRegionConfigOptions

WSDL Reference: analytics.wsdl, media2.wsdl

Test Purpose: To verify retrieving of License Plate Recognition Options by GetRuleOptions operation.

Pre-Requisite: Analytics Service is received from the DUT. Media2 Service is received from the DUT. Rule Engine is supported by the Device as indicated by the RuleSupport capabilities, Rule

Options is supported by the Device as indicated by the RuleOptionsSupported capabilities. License Plate Recognition Rule is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of Analytics Configurations that supports tt:LicensePlateRecognition type by following the procedure mentioned in Annex A.2 with the following input and output parameters

    • in **tt:LicensePlateRecognition** - rule type

    • out *analyticsConfList* - a list of Analytics configurations with supporting of License Plate Recognition

4. If *analyticsConfList* is empty, FAIL the test and skip other steps.

5. For each Analytics Configuration (*analyticsConf*) in *analyticsConfList* repeat the following steps:

    5.1. ONVIF Client retrieves a list of supported rules by following the procedure mentioned in Annex A.29 with the following input and output parameters

    • in *analyticsConf*.@token - a token of Analytics Configuration

    • out *supportedRules* - supported rules.

    5.2. ONVIF Client invokes **GetRuleOptions** request with parameters

    • RuleType := tt:LicensePlateRecognition

    • ConfigurationToken := *analyticsConf*.@token

    5.3. DUT responds with **GetRuleOptionsResponse** message with parameters

    • RuleOptions list =: *ruleOptionsList*

    5.4. If *ruleOptionsList* contains RuleOption with @Name = **IncludeImage** (*includeImageOption*)

        5.4.1. If *includeImageOption* does not contain **StringItems** element or **StringList** element, FAIL the test and skip other steps.

5.5. If *ruleOptionsList* contains RuleOption with @Name = **PlateLocation** (*plateLocationOption*)

    5.5.1. If *plateLocationOption* does not contain **StringItems** element or **StringList** element, FAIL the test and skip other steps.

5.6. If *ruleOptionsList* contains RuleOption with @Name = **Region** (*regionOption*)

    5.6.1. If *regionOption* does not contain **tt:PolygonOptions** element, FAIL the test and skip other steps.

5.7. If *ruleOptionsList* contains RuleOption with @Name = **SnapLine** (*snapLineOption*)

    5.7.1. If *snapLineOption* does not contain **tt:IntRange** element, FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.9 Line Crossing Counting Rule

5.9.1 GET SUPPORTED RULES (LINE CROSSING COUNTER)

Test Case ID: ANALYTICS-9-1-1

Specification Coverage: Get Supported rules (ONVIF Analytics Service Spec), Line crossing counting rule (ONVIF Analytics Service Spec)

Feature Under Test: GetSupportedRules, RuleDescription for tt:LineCounting

WSDL Reference: analytics.wsdl, media2.wsdl

Test Purpose: To verify that device includes tt:LineCounting in GetSupportedRulesResponse. To verify structure of Line crossing counting.

Pre-Requisite: Analytics Service is received from the DUT. Media2 Service is received from the DUT. Rule Engine is supported by the Device as indicated by the RuleSupport capabilities. Line crossing counting Rule is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

    • out *analyticsConfList* - a list of Analytics configurations

4. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

    4.1. ONVIF Client invokes **GetSupportedRules** request with parameters

        • ConfigurationToken := *analyticsConf*.@token

    4.2. DUT responds with **GetSupportedRulesResponse** message with parameters

        • SupportedRules =: *supportedRules*

    4.3. If *supportedRules* contains RuleDescription element (*lineCountingRuleDescription*) with Name value is equal to **tt:LineCounting**:

        4.3.1. If *lineCountingRuleDescription* does not have maxInstances attribute, FAIL the test and skip other steps.

        4.3.2. ONVIF Client checks set of items inside Parameters of *lineCountingRuleDescription*:

            4.3.2.1. If *lineCountingRuleDescription*.Parameters has SimpleItemDescription element with @Name = "ReportTimeInterval" and its @Type value != xs:duration, FAIL the test and skip other steps.

            4.3.2.2. If *lineCountingRuleDescription*.Parameters has SimpleItemDescription element with @Name = "ResetTime" and its @Type value != xs:time, FAIL the test and skip other steps.

            4.3.2.3. If *lineCountingRuleDescription*.Parameters has SimpleItemDescription element with @Name = "Direction" and its @Type value != tt:Direction, FAIL the test and skip other steps.

4.3.2.4. If *lineCountingRuleDescription*.Parameters has SimpleItemDescription element with @Name = "PassAllPolylines" and its @Type value != xs:boolean, FAIL the test and skip other steps.

4.3.2.5. If *lineCountingRuleDescription*.Parameters does not have ElementItemDescription element with @Name = "Segments", FAIL the test and skip other steps.

4.3.2.6. If Type value of ElementItemDescription element with @Name = "Segments" is not equal to tt:Polyline, FAIL the test and skip other steps.

4.3.3. If Messages.IsProperty != "true", FAIL the test and skip other steps.

4.3.4. ONVIF Client checks set of items inside Messages.Source of *lineCountingRuleDescription*:

4.3.4.1. ONVIF Client verifies rule engine message format by following the procedure mentioned in Annex A.46 with the following input and output parameters

- in *lineCountingRuleDescription*.Messages - message description to be verified

4.3.5. ONVIF Client checks set of items inside Messages.Data of *lineCountingRuleDescription*:

4.3.5.1. If *lineCountingRuleDescription* does not have Messages.Data.SimpleItemDescription element with @Name = "Count", FAIL the test and skip other steps.

4.3.5.2. If Type value of Messages.Data.SimpleItemDescription element with @Name = "Count" is not equal to xs:int, FAIL the test and skip other steps.

4.3.6. If Messages.ParentTopic value is not equal to "tns1:RuleEngine/CountAggregation/Counter" for Messages with Source.SimpleItemDescription.@Name = VideoSource, and with Source.SimpleItemDescription.@Name = Rule, FAIL the test and skip other steps.

5. If there was no RuleDescription element with Name value is equal to **tt:LineCounting** for at least one *supportedRules* at step 4.3, FAIL the test.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.9.2 GET LINE CROSSING COUNTER RULE OPTIONS

Test Case ID: ANALYTICS-9-1-2

Specification Coverage: Get Rule Options (ONVIF Analytics Service Spec), Line crossing counting rule (ONVIF Analytics Service Spec), Configuration options (ONVIF Analytics Service Spec).

Feature Under Test: GetRuleOptions, Line crossing counting rule

WSDL Reference: analytics.wsdl, media2.wsdl

Test Purpose: To verify retrieving of Line crossing counting Options by GetRuleOptions operation.

Pre-Requisite: Analytics Service is received from the DUT. Media2 Service is received from the DUT. Rule Engine is supported by the Device as indicated by the RuleSupport capabilities, Rule Options is supported by the Device as indicated by the RuleOptionsSupported capabilities. Line crossing counting Rule is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client retrieves a list of Analytics Configurations that supports tt:LineCounting type by following the procedure mentioned in Annex A.2 with the following input and output parameters

   - in **tt:LineCounting** - rule type

   - out *analyticsConfList* - a list of Analytics configurations with supporting of Line crossing counting

4. If *analyticsConfList* is empty, FAIL the test and skip other steps.

5. For each Analytics Configuration (*analyticsConf*) in *analyticsConfList* repeat the following steps:

    5.1. ONVIF Client retrieves a list of supported rules by following the procedure mentioned in Annex A.29 with the following input and output parameters

        • in *analyticsConf*.@token - a token of Analytics Configuration

        • out *supportedRules* - supported rules.

    5.2. ONVIF Client invokes **GetRuleOptions** request with parameters

        • RuleType := tt:LineCounting

        • ConfigurationToken := *analyticsConf*.@token

    5.3. DUT responds with **GetRuleOptionsResponse** message with parameters

        • RuleOptions list =: *ruleOptionsList*

    5.4. If *ruleOptionsList* contains RuleOption with @Name = **ReportTimeInterval** (*reportTimeIntervalOption*)

        5.4.1. If *reportTimeIntervalOption* does not contain **tt:DurationRange** element, FAIL the test and skip other steps.

    5.5. If *ruleOptionsList* contains RuleOption with @Name = **Segments** (*segmentsOption*)

        5.5.1. If *segmentsOption* does not contain **tt:IntRange** element, FAIL the test and skip other steps.

Test Result:

PASS –

• DUT passes all assertions.

FAIL –

• DUT did not send **GetRuleOptionsResponse** message.

Annex A Helper Procedures and Additional Notes

A.1 Get Analytics Configurations List

Name: HelperGetAnalyticsConfigurationsList

Procedure Purpose: Helper procedure to retrieve Analytics Configurations List.

Pre-requisite: Media2 Service is received from the DUT.

Input: None.

Returns: Analytics Configurations list (*analyticsConfList*).

Procedure:

1. ONVIF Client invokes **GetAnalyticsConfigurations** request with parameters

    • ConfigurationToken skipped

    • ProfileToken skipped

2. The DUT responds with **GetAnalyticsConfigurationsResponse** with parameters

    • Configurations list =: *analyticsConfList*

3. If *analyticsConfList* is empty, FAIL the test.

Procedure Result:

PASS –

 • DUT passes all assertions.

FAIL –

 • DUT did not send **GetAnalyticsConfigurationsResponse** message.

A.2 Get List of Analytics Configurations With Supporting of Required Rule Type

Name: HelperGetAnalyticsConfigurationSupportsRequiredRuleTypeList

Procedure Purpose: Helper procedure to retrieve full list of Analytics Configuration that supports required rule type.

Pre-requisite: Analytics Service is received from the DUT. Rule Engine is supported by the Device as indicated by the RuleSupport capabilities.

Input: Rule Type (*ruleType*).

Returns: List of Analytics Configuration that supports rule with type equals to *ruleType* (*analyticsConfSupportsRuleTypeList*).

Procedure:

1. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

   • out *analyticsConfList* - a list of Analytics configurations

2. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

   2.1. ONVIF Client invokes **GetSupportedRules** request with parameters

      • ConfigurationToken := *analyticsConf*.@token

   2.2. DUT responds with **GetSupportedRulesResponse** message with parameters

      • SupportedRules =: *supportedRules*

   2.3. If *supportedRules* contains RuleDescription element with Name value is equal to *ruleType*, set *analyticsConfSupportsRuleTypeList* := *analyticsConfSupportsRuleTypeList* + *analyticsConf*.@token

Procedure Result:

PASS –

   • DUT passes all assertions.

FAIL –

   • DUT did not send **GetSupportedRulesResponse** message.

A.3 Get Specific Rule Options

Name: HelperGetSpecificRuleOptions

Procedure Purpose: Helper procedure to retrieve options of required rule type.

Pre-requisite: Analytics Service is received from the DUT. Rule Engine is supported by the Device as indicated by the RuleSupport capabilities.

Input: Analytics Configuration Token (*analyticsConfToken*), Rule Type (*ruleType*).

Returns: Rule Options *ruleOptions* of *ruleType*.

Procedure:

1. ONVIF Client invokes **GetRuleOptions** request with parameters

    • RuleType := *ruleType*

    • ConfigurationToken := *analyticsConf*.@token

2. DUT responds with **GetRuleOptionsResponse** message with parameters

    • RuleOptions list =: *ruleOptionsList*

3. If **ruleOptionsList** contains more than one RuleOptions element, FAIL the test abd skip other steps.

4. If **ruleOptionsList** is empty, FAIL the test abd skip other steps.

5. Set *ruleOptions* := **ruleOptionsList**[0].

Procedure Result:

PASS –

    • DUT passes all assertions.

FAIL –

    • DUT did not send **GetRuleOptionsResponse** message.

A.4 Configure Media Profile with required Analytics Configuration

Name: HelperConfigureMediaProfileWithRequiredAnalytics

Procedure Purpose: Helper procedure to configure Media Profile to contain required Analytics Configuration.

Pre-requisite: Media2 Service is received from the DUT. Analytics is supported by the DUT.

Input: List of Analytics configurations *analyticsConfList*

Returns: Media Profile (*profile*) that contains Analytics Configuration from *analyticsConfList* and Video Source Configuration.

Procedure:

1. ONVIF Client invokes **GetProfiles** request with parameters

   - Token skipped

   - Type[0] := VideoSource

   - Type[1] := Analytics

2. The DUT responds with **GetProfilesResponse** message with parameters

   - Profiles list =: *profileList*

3. For each Media Profile *profile1* in *profileList* with both Configuration.VideoSource and Configuration.Analytics repeat the following steps:

   3.1. For each Analytics (*analytics*) in *analyticsConfList*:

      3.1.1.If *profile1*.Configuration.Analytics.@token value is equal to *analytics*.@token, set *profile* := *profile1* and skip other steps in procedure.

4. For each Media Profile *profile1* in *profileList* that contains VideoSource configuration repeat the following steps:

   4.1. ONVIF Client invokes **GetAnalyticsConfigurations** request with parameters

      - ConfigurationToken skipped

      - ProfileToken := *profile*.@token

   4.2. The DUT responds with **GetAnalyticsConfigurationsResponse** message with parameters

      - Configurations list =: *acList*

   4.3. If *acList* contains analytics (*analytics*) from *analyticsConfList* (comparing by analytics token):

      4.3.1. ONVIF Client invokes **AddConfiguration** request with parameters

         - ProfileToken := *profile1*.@token

         - Name skipped

         - Configuration[0].Type := Analytics

         - Configuration[0].Token := *analytics*.Configurations.@token

      4.3.2. The DUT responds with **AddConfigurationResponse** message.

4.3.3. Set *profile* := *profile1* and skip other steps in procedure.

5. FAIL the test and skip other steps.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

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

A.5 Get Rules

Name: HelperGetRules

Procedure Purpose: Helper procedure to retrieve Rules list for Analytics configuration.

Pre-requisite: Media2 Service is received from the DUT. Rule Engine is supported by the Device as indicated by the RuleSupport capabilities.

Input: Analytics configuration token (*analyticsToken*).

Returns: Rule list (*ruleList*).

Procedure:

1. ONVIF Client invokes **GetRules** request with parameters

   - ConfigurationToken := (*analyticsToken*)

2. The DUT responds with **GetRulesResponse** with parameters

   - Rule list =: *ruleList*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.6 Create Pull Point Subscription

Name: HelperCreatePullPointSubscription

Procedure Purpose: Helper procedure to create PullPoint Subscription with specified Topic.

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

Input: Notification Topic (*topic*).

Returns: Subscription reference (*s*), current time for the DUT (*ct*), subscription termination time (*tt*).

Procedure:

1. ONVIF Client invokes **CreatePullPointSubscription** request with parameters

    • Filter.TopicExpression := *topic*

    • Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"

2. The DUT responds with **CreatePullPointSubscriptionResponse** message with parameters

    • SubscriptionReference =: *s*

    • CurrentTime =: *ct*

    • TerminationTime =: *tt*

Procedure Result:

PASS –

    • DUT passes all assertions.

FAIL –

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

A.7 Delete Subscription

Name: HelperDeleteSubscription

Procedure Purpose: Helper procedure to delete supscribtion.

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

Input: Subscription reference (*s*)

Returns: None

Procedure:

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

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

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.8 Calculate Free Space for Rule

Name: Annex_HelperCalculateFreeSpaceForRule

Procedure Purpose: Helper procedure to calculate free space for additional rules with required Rule Type for requested Analytics Configuration.

Pre-requisite: Analytics Service is received from the DUT. Rule Engine is supported by the Device as indicated by the RuleSupport capabilities.

Input: Analytics Configuration token *analyticsConfigToken*. Required Rule Type *ruleType*.

Returns: Flag if maxInstances is supported (*maxInstances*). Amount of additional rules with requested type that may be added for requested Analytics Configuration (*amountOfAdditionalRules*) (optional, returned in case *maxInstances* = true).

Procedure:

1. ONVIF Client invokes **GetSupportedRules** request with parameters

   - ConfigurationToken := *analyticsConfigToken*

2. DUT responds with **GetSupportedRulesResponse** message with parameters

   - SupportedRules =: *supportedRules*

3. If *supportedRules* does not contain RuleDescription element with Name value is equal to *ruleType*, FAIL the test and skip other steps.

4. Set *rule* := *supportedRules*.RuleDescription[0], where RuleDescription[0] is RuleDescription element with Name value is equal to *ruleType*.

5. If *rule* does not contain maxInstances attribute, set *maxInstances* := false, return it in test procedure and skip other annex steps.

6. Set *maxInstances* := true.

7. ONVIF Client invokes **GetRules** request with parameters

    • ConfigurationToken := *analyticsConfigToken*

8. DUT responds with **GetRulesResponse** message with parameters

    • Rule list =: *ruleList*

9. Set *amountOfExistingRules* := amount of Rules in *ruleList* with Type = *ruleType*.

10. Set *amountOfAdditionalRules* := *rule*.maxInstances - *amountOfExistingRules*.

Procedure Result:

PASS –

    • DUT passes all assertions.

FAIL –

    • DUT did not send **GetSupportedRulesResponse** message.

    • DUT did not send **GetRulesResponse** message.

A.9 Delete Rule with Requested Type

Name: HelperDeleteRuleWithRequestedType

Procedure Purpose: Helper procedure to delete existing Rule with requested type for specified Analytics Configuration.

Pre-requisite: Media2 Service is received from the DUT. Rule Engine is supported by the Device as indicated by the RuleSupport capabilities.

Input: Analytics configuration token (*analyticsToken*). Rule type to delete (*ruleType*).

Returns: None.

Procedure:

1. ONVIF Client invokes **GetRules** request with parameters

    • ConfigurationToken := *analyticsToken*

2. DUT responds with **GetRulesResponse** message with parameters

- Rule list =: *ruleList*

3. Set *ruleToDelete* := *ruleList*[0], where *ruleList*[0] is the first Rule with Type = *ruleType*.

4. ONVIF Client invokes **DeleteRules** request with parameters

   - ConfigurationToken := (*analyticsToken*)

   - RuleName[0] := *ruleToDelete*.Name

5. The DUT responds with **DeleteRulesResponse**.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.10 Create Motion Region Detector Rule

Name: HelperCreateMotionRegionDetectorRule

Procedure Purpose: Helper procedure to create Motion Region Detector Rule.

Pre-requisite: Media2 Service is received from the DUT. Rule Engine is supported by the Device as indicated by the RuleSupport capabilities.

Input: Rule Options *ruleOptions*. Analytics Configuration Token (*analyticsToken*).

Returns: None.

Procedure:

1. ONVIF Client invokes **CreateRules** request with parameters

   - ConfigurationToken := *analyticsToken*

   - Rule[0].@Name := TestMotionRegion

   - Rule[0].@Type := tt:MotionRegionDetector

   - Rule[0].Parameters.ElementItem[0].@Name := "MotionRegion"

   - Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[0].@x   := *profile*.Configurations.VideoSource.Bounds.@x

- Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[0].@y := *profile*.Configurations.VideoSource.Bounds.@y

- Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[1].@x := *profile*.Configurations.VideoSource.Bounds.@x

- Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[1].@y := *profile*.Configurations.VideoSource.Bounds.@y + *profile*.Configurations.VideoSource.Bounds.@height - 1

- Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[2].@x := *profile*.Configurations.VideoSource.Bounds.@x + *profile*.Configurations.VideoSource.Bounds.@width - 1

- Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[2].@y := *profile*.Configurations.VideoSource.Bounds.@y + *profile*.Configurations.VideoSource.Bounds.@height - 1

- Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[3].@x := *profile*.Configurations.VideoSource.Bounds.@x + *profile*.Configurations.VideoSource.Bounds.@width - 1

- Rule[0].Parameters.ElementItem[0].MotionRegion.Polygon.Point[3].@y := *profile*.Configurations.VideoSource.Bounds.@y

- Rule[0].Parameters.ElementItem[0].MotionRegion.@Armed := true if *motionRegionConfigOptions*.DisarmSupport = true, otherwise skipped.

- Rule[0].Parameters.ElementItem[0].MotionRegion.@Sensitivity := 1

2. The DUT responds with **CreateRulesResponse**.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.11 Topic Format Verification

Name: HelperTopicCheck

Procedure Purpose: Helper procedure to verifye topic format.

Pre-requisite: None.

Input: Topic to be verified (*topic*).

Returns: None.

Procedure:

1. If *topic* contains at least one whitespace, FAIL the test, restore the DUT state, and skip other steps.

2. If *topic* does not corrspond to the format defined in Annex A.12, FAIL the test, restore the DUT state, and skip other steps.

3. If *topic* contains at least one namespace prefix which namespace is not declared according to [XML Schema Part 2], section 3.2.18 QName, FAIL the test, restore the DUT state, and skip other steps.

Procedure Result:

PASS –

• DUT passes all assertions.

FAIL –

• DUT does not pass all assertions.

A.12 Valid Topic Format

Topic shall correspond to the following format in Extended Backus Naur Form:

• TopicExpression ::= TopicPath

• TopicPath ::= RootTopic ChildTopicExpression*

• RootTopic ::= QName

• ChildTopicExpression ::= '/' ChildTopicName

• ChildTopicName ::= QName | NCName

A.13 Get All Supported Analytics Modules

Name: GetAllSupportedAnalyticsModules

Procedure Purpose: Helper procedure to retrieve full list of Analytics Modules supported by a DUT.

Pre-requisite: Media2 Service is received from the DUT. Analytics Modules is supported by the DUT.

Input: None.

Returns: Full Analytics Module Description List (*fullAnalyticsModuleDescriptionList*).

Procedure:

1. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

    • out *analyticsConfList* - a list of Analytics configurations

2. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

    2.1. ONVIF Client invokes **GetSupportedAnalyticsModules** request with parameters

       • ConfigurationToken := *analyticsConf*.@token

    2.2. DUT responds with **GetSupportedAnalyticsModulesResponse** message with parameters

       • SupportedAnalyticsModules =: *supportedAnalyticsModules*

    2.3. Set *supportedAnalyticsModules*.AnalyticsModuleDescription list =: *analyticsModuleDescriptionList1*

    2.4. Set *fullAnalyticsModuleDescriptionList* := *fullAnalyticsModuleDescriptionList* + *analyticsModuleDescriptionList1*

Procedure Result:

PASS –

   • DUT passes all assertions.

FAIL –

   • DUT did not send **GetSupportedAnalyticsModulesResponse** message.

A.14 Get Supported Metadata

Name: HelperGetSupportedMetadata

Procedure Purpose: Helper procedure to retrieve full list of metadata supported by a DUT.

Pre-requisite: Supported Metadata feature is supported by the DUT.

Input: None.

Returns: Analytics Module List with Metadata Info (*analyticsModuleList*).

Procedure:

1. ONVIF Client invokes **GetSupportedMetadata** request with parameters

   • Type skipped

2. DUT responds with **GetSupportedMetadataResponse** message with parameters

   • AnalyticsModule list =: *analyticsModuleList*

Procedure Result:

PASS –

   • DUT passes all assertions.

FAIL –

   • DUT did not send **GetSupportedMetadataResponse** message.

A.15 Get Supported Metadata for Analytics Module Type

Name: HelperGetSupportedMetadataForAnalyticsModuleType

Procedure Purpose: Helper procedure to retrieve metadata supported by a requested Analytics Module Type.

Pre-requisite: Supported Metadata feature is supported by the DUT.

Input: Analytics Module Type (*analyticsModuleType*).

Returns: Analytics Module List with Metadata Info (*analyticsModuleList*).

Procedure:

1. ONVIF Client invokes **GetSupportedMetadata** request with parameters

   • Type := *analyticsModuleType*

2. DUT responds with **GetSupportedMetadataResponse** message with parameters

   • AnalyticsModule list =: *analyticsModuleList*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.16 Get List of Analytics Configurations With Supporting of Non Fixed Analytics Modules

Name: HelperGetAnalyticsConfigurationSupportsNonFixedAnalyticsModules

Procedure Purpose: Helper procedure to retrieve full list of Analytics Configuration that supports at least one non fixed Analytics Module.

Pre-requisite: Analytics Service is received from the DUT. Media2 Service is received from the DUT. Analytics Modules is supported by the DUT. Media2 Service .

Input: None.

Returns: List of Analytics Configuration that supports Analytics Modules (*analyticsConfListSupportsAnalyticsModules*).

Procedure:

1. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

   - out *analyticsConfList* - a list of Analytics configurations

2. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

   2.1. ONVIF Client invokes **GetSupportedAnalyticsModules** request with parameters

      - ConfigurationToken := *analyticsConf*.@token

   2.2. DUT responds with **GetSupportedAnalyticsModulesResponse** message with parameters

      - SupportedAnalyticsModules =: *supportedModules*

   2.3. If *supportedModules*.AnalyticsModuleDescription list is not empty and has at least one AnalyticsModuleDescription with @fixed ! = true, set *analyticsConfListSupportsAnalyticsModules* := *analyticsConfListSupportsAnalyticsModules* + *analyticsConf*.@token

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.17 Get Supported Analytics Modules

Name: HelperGetSupportedAnalyticsModules

Procedure Purpose: Helper procedure to retrieve a list of analytics modules that are supported by the given analytics configuration.

Pre-requisite: Analytics Modules is supported by the DUT.

Input: Analytics configuration token (*analyticsConfigToken*).

Returns: Supported Analytics Modules (*supportedAnalyticsModules*).

Procedure:

1. ONVIF Client invokes **GetSupportedAnalyticsModules** request with parameters

    - ConfigurationToken := *analyticsConfigToken*

2. DUT responds with **GetSupportedAnalyticsModulesResponse** message with parameters

    - SupportedAnalyticsModules =: *supportedAnalyticsModules*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.18 Prepare Free Space for Analytics Module

Name: HelperFreeSpaceForModule

Procedure Purpose: Helper procedure to configure Analytics Configuration with free space for adding of Analytics Module with requested type.

Pre-requisite: Analytics Service is received from the DUT. Analytics Modules is supported by the DUT.

Input: Analytics Configuration token *analyticsConfigToken*. Analytics module type (*analyticsModuleType*) for which free space is required. Maximum number of instances per configuration *maxInstances*. (optional) Maximum number of concurrent modules per configuration (*limit*).

Returns: Analytics module (optional) (*deletedModule*).

Procedure:

1. ONVIF Client invokes **GetAnalyticsModules** request with parameters

   • ConfigurationToken := *analyticsConfigToken*

2. DUT responds with **GetAnalyticsModulesResponse** message with parameters

   • AnalyticsModule list =: *analyticsModuleList*

3. Set *amountOfExistingRequieredModules* := amount of AnalyticsModule in *analyticsModuleList* with Type = *analyticsModuleType*.

4. If *amountOfExistingRequieredModules* = *maxInstances*

   4.1. Set *moduleToDelete* := the first analytics module from *analyticsModuleList* that has @Type = *analyticsModuleType*.

   4.2. ONVIF Client deletes one analytics module by following the procedure mentioned in Annex A.19 with the following input and output parameters

      • in *analyticsConfigToken* - token of Analytics configuration

      • in *moduleToDelete*.@Name - Name of the analytics module to be deleted

   4.3. Update *analyticsModuleList*.

   4.4. Returns *moduleToDelete*.

5. If *limit* is not specified, skip other steps of the procedure and return to the test.

6. Set *amountOfExistingModules* := amount of AnalyticsModule items in *analyticsModuleList*.

7. If *amountOfExistingModules* = *limit*

   7.1. ONVIF Client retrieves a list of supported analytics modules by following the procedure mentioned in Annex A.17 with the following input and output parameters

      • in *analyticsConfigToken* - a token of Analytics Configuration

• out *supportedAnalyticsModules* - supported analytics modules.

7.2. Set *nonFixedAnalyticsModulesTypeList* := list of AnalyticsModuleDescription.@Name items from *supportedAnalyticsModules* with AnalyticsModuleDescription.@fixed != true.

7.3. If *analyticsModuleList* does not contain item with @Type listed in *nonFixedAnalyticsModulesTypeList*, FAIL the test, restore the DUT state, and skip other steps.

7.4. Set *moduleToDelete* := the first analytics module from *analyticsModuleList* that has @Type listed in *nonFixedAnalyticsModulesTypeList*.

7.5. ONVIF Client deletes one analytics module by following the procedure mentioned in Annex A.19 with the following input and output parameters

• in *analyticsConfigToken* - token of Analytics configuration

• in *moduleToDelete*.@Name - Name of the analytics module to be deleted

7.6. Returns *moduleToDelete*.

Procedure Result:

PASS –

• DUT passes all assertions.

FAIL –

• DUT did not send **GetAnalyticsModulesResponse** message.

A.19 Delete Analytics Module

Name: HelperDeleteAnalyticsModule

Procedure Purpose: Helper procedure to delete analytics module from specified Analytics Configuration.

Pre-requisite: Analytics Service is received from the DUT. Analytics Modules is supported by the DUT.

Input: Analytics configuration token (*analyticsToken*). Name of the analytics module (*analyticsModuleName*).

Returns: None.

Procedure:

1. ONVIF Client invokes **DeleteAnalyticsModules** request with parameters

    • ConfigurationToken := *analyticsToken*

    • AnalyticsModuleName[0] := *analyticsModuleName*

2. The DUT responds with **DeleteAnalyticsModulesResponse**.

Procedure Result:

PASS –

    • DUT passes all assertions.

FAIL –

    • DUT did not send **DeleteAnalyticsModulesResponse** message.

A.20 Get Analytics Modules

Name: HelperGetAnalyticsModules

Procedure Purpose: Helper procedure to retrieve currently assigned set of analytics modules of a Analytics Configuration.

Pre-requisite: Analytics Modules is supported by the DUT.

Input: Token of Analytics Configuration (*analyticsConfToken*).

Returns: Analytics Module List (*analyticsModuleList*).

Procedure:

1. ONVIF Client invokes **GetAnalyticsModules** request with parameters

    • ConfigurationToken := *analyticsConfToken*

2. DUT responds with **GetAnalyticsModulesResponse** message with parameters

    • AnalyticsModule list =: *analyticsModuleList*

Procedure Result:

PASS –

    • DUT passes all assertions.

FAIL –

• DUT did not send **GetAnalyticsModulesResponse** message.

A.21 Get Analytics Configuration

Name: HelperGetAnalyticsConfiguration

Procedure Purpose: Helper procedure to retrieve Analytics Configuration.

Pre-requisite: Media2 Service is received from the DUT.

Input: Token of Analytics Configuration (*analyticsConfToken*).

Returns: Analytics Configuration (*analyticsConf*).

Procedure:

1. ONVIF Client invokes **GetAnalyticsConfigurations** request with parameters

   • ConfigurationToken := *analyticsConfToken*

   • ProfileToken skipped

2. The DUT responds with **GetAnalyticsConfigurationsResponse** with parameters

   • Configurations list =: *analyticsConfList*

Procedure Result:

PASS –

• DUT passes all assertions.

FAIL –

• DUT did not send **GetAnalyticsConfigurationsResponse** message.

A.22 Create Analytics Module

Name: HelperCreateAnalyticsModule

Procedure Purpose: Helper procedure to create analytics module from specified Analytics Configuration.

Pre-requisite: Analytics Service is received from the DUT. Analytics Modules is supported by the DUT.

Input: Analytics configuration token (*analyticsToken*). Name of the analytics module (*analyticsModuleName*). Type of the analytics module (*analyticsModuleType*).

Returns: None.

Procedure:

1. ONVIF Client invokes **CreateAnalyticsModules** request with parameters

   - ConfigurationToken := *analyticsToken*

   - AnalyticsModule[0].@Name := *analyticsModuleName*

   - AnalyticsModule[0].@Type := *analyticsModuleType*

   - AnalyticsModule[0].Parameters := empty Parameters element

2. DUT responds with **CreateAnalyticsModulesResponse** message

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.23 Device Configuration For Create Analytics Module

Name: HelperDeviceConfigurationForCreateAnalyticsModule

Procedure Purpose: Helper procedure for device configuration for adding of analytics module.

Pre-requisite: Analytics Service is received from the DUT. Analytics Modules is supported by the DUT.

Input: None.

Returns: If non fixed Analytics Module is supported (*nonFixedModuleSupported*). Analytics configuration token (optional) (*analyticsToken*). Analytics module description (optional) (*analyticsModuleDescription*). Analytics module to restore (optional) (*moduleToRestore*)

Procedure:

1. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

   - out *analyticsConfList* - a list of Analytics configurations

2. ONVIF Client retrieves a list of Analytics Configurations that supports non fixed AnalyticsModules by following the procedure mentioned in Annex A.16 with the following input and output parameters

   • out *analyticsConfListSupportsAnalyticsModules* - a list of Analytics configurations

3. If *analyticsConfListSupportsAnalyticsModules* is empty, set *nonFixedModuleSupported* = false and return to test procedure.

4. ONVIF Client configures media profile with Analytics Configuration from *analyticsConfListSupportsAnalyticsModules* list by following the procedure mentioned in Annex A.4 with the following input and output parameters

   • in *analyticsConfListSupportsAnalyticsModules* - list of Analytics configurations.

   • out *profile* - media profile.

5. Set *analyticsToken* := *profile*.Configurations.Analytics.token

6. ONVIF Client retrieves a list of supported analytics modules by following the procedure mentioned in Annex A.17 with the following input and output parameters

   • in *analyticsToken* - a token of Analytics Configuration

   • out *supportedAnalyticsModules* - supported analytics modules.

7. If *supportedAnalyticsModules*.AnalyticsModuleDescription list is empty, FAIL the test and skip other steps.

8. Set *analyticsModuleDescription* := first *supportedAnalyticsModules*.AnalyticsModuleDescription item with @fixed != true.

9. Set *limit* := *supportedAnalyticsModules*.@Limit (if specified).

10. If *analyticsModuleDescription* does not contain maxInstances attribute, FAIL the test.

11. If *analyticsModuleDescription*.@maxInstances = 0, FAIL the test.

12. ONVIF Client prepares free space for adding of new analytics module with required type into the Analytics Configuration by following the procedure mentioned in Annex A.18 with the following input and output parameters

   • in *analyticsToken* - a token of Analytics Configuration

   • in *analyticsModuleDescription*.@Name - analytics module type

   • in *analyticsModuleDescription*.@maxInstances - maximum number of instances per configuration

- (optional) in *limit* - limit

    - out *moduleToRestore* (optional) - analytics module to restore

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- None.

A.24 Select Existing Analytics Module

Name: HelperSelectExistingAnalyticsModule

Procedure Purpose: Helper procedure to select Analytics Module existing on the device.

Pre-requisite: Analytics Service is received from the DUT. Analytics Modules is supported by the DUT.

Input: None.

Returns: Analytics module (optional) (*analyticsModule*), Analytics Configuration token (*analyticsToken*).

Procedure:

1. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

    - out *analyticsConfList* - a list of Analytics configurations

2. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

    2.1. ONVIF Client retrieves a set of analytics modules assigned of Analytics Configuration by following the procedure mentioned in Annex A.20 with the following input and output parameters

        - in *analyticsConf*.@token - a token of Analytics Configuration

        - out *analyticsModuleList* - analytics module list

    2.2. If *analyticsModuleList* list is not empty

2.2.1. Set *analyticsModule* := *analyticsModuleList*[0].

2.2.2. Set *analyticsToken* := *analyticsConf*.@token.

2.2.3. Return to test procedure.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- None.

A.25 Check Shape Descriptor Coordinates

Name: HelperSelectExistingAnalyticsModule

Procedure Purpose: Helper procedure to check that object bounding boxes or shapes included into sample frame of GetSupportedMetadataResponse are located in the top left quarter of the image.

Pre-requisite: None.

Input: SampleFrame *sampleFrame*

Returns: None.

Procedure:

1. Set *translateX* = 0.

2. Set *translateY* = 0.

3. Set *scaleX* = 1.

4. Set *scaleY* = 1.

5. If *sampleFrame* contains Transformation element, rewrite variables:

   5.1. If *sampleFrame*.Transformation.Translate.@x is specified, set *translateX* = *sampleFrame*.Transformation.Translate.@x.

   5.2. If *sampleFrame*.Transformation.Translate.@y is specified, set *translateY* = *sampleFrame*.Transformation.Translate.@y.

   5.3. If *sampleFrame*.Transformation.Scale.@x is specified, set *scaleX* = *sampleFrame*.Transformation.Scale.@x.

    5.4. If *sampleFrame*.Transformation.Scale.@y is specified, set *scaleY* = *sampleFrame*.Transformation.Scale.@y.

6. For each Object.Appearance with Shape element (*appearance*) in *sampleFrame* repeat the following steps:

    6.1. If *appearance* contains Transformation element, rewrite variables:

        6.1.1. If *appearance*.Transformation.Translate.@x is specified, set *translateX* = *appearance*.Transformation.Translate.@x. Otherwise set *translateX* = 0.

        6.1.2. If *appearance*.Transformation.Translate.@y is specified, set *translateY* = *appearance*.Transformation.Translate.@y. Otherwise set *translateY* = 0.

        6.1.3. If *appearance*.Transformation.Scale.@x is specified, set *scaleX* = *appearance*.Transformation.Scale.@x. Otherwise set *scaleX* = 1.

        6.1.4. If *appearance*.Transformation.Scale.@y is specified, set *scaleY* = *appearance*.Transformation.Scale.@y. Otherwise set *scaleY* = 1.

    6.2. If *appearance*.Shape contains BoundingBox element:

        6.2.1. ONVIF Client transforms bottom coordinate to default coordinate system by following the procedure mentioned in Annex A.26 with the following input and output parameters

- in *appearance*.Shape.BoundingBox.@bottom - coordinate of point to transform

- in *translateY* - Translational value

- in *scaleY* - Scaling value

- out *bottom* - bottom in default coordinate system

        6.2.2. ONVIF Client transforms top coordinate to default coordinate system by following the procedure mentioned in Annex A.26 with the following input and output parameters

- in *appearance*.Shape.BoundingBox.@top - coordinate of point to transform

- in *translateY* - Translational value

- in *scaleY* - Scaling value

- out *top* - top in default coordinate system

6.2.3. ONVIF Client transforms right coordinate to default coordinate system by following the procedure mentioned in Annex A.26 with the following input and output parameters

- in *appearance*.Shape.BoundingBox.@right - coordinate of point to transform

- in *translateX* - Translational value

- in *scaleX* - Scaling value

- out *right* - right in default coordinate system

6.2.4. ONVIF Client transforms left coordinate to default coordinate system by following the procedure mentioned in Annex A.26 with the following input and output parameters

- in *appearance*.Shape.BoundingBox.@left - coordinate of point to transform

- in *translateX* - Translational value

- in *scaleX* - Scaling value

- out *left* - left in default coordinate system

6.2.5. If *bottom* > 1 or *bottom* < 0, FAIL the test.

6.2.6. If *top* > 1 or *top* < 0, FAIL the test.

6.2.7. If *right* > 0 or *top* < -1, FAIL the test.

6.2.8. If *left* > 0 or *top* < -1, FAIL the test.

6.2.9. For each *appearance*.Polygon.Point:

6.2.9.1. ONVIF Client transforms X coordinate to default coordinate system by following the procedure mentioned in Annex A.26 with the following input and output parameters

- in *appearance*.Polygon.Point.@x - coordinate of point to transform

- in *translateX* - Translational value

- in *scaleX* - Scaling value

- out *pointX* - X of point in default coordinate system

6.2.9.2 ONVIF Client transforms Y coordinate to default coordinate system by following the procedure mentioned in Annex A.26 with the following input and output parameters

- in *appearance*.Polygon.Point.@y - coordinate of point to transform

- in *translateY* - Translational value

- in *scaleY* - Scaling value

- out *pointY* - Y of point in default coordinate system

6.2.9.3 If *pointY* > 1 or *bottom* < 0, FAIL the test.

6.2.9.4 If *pointX* > 0 or *top* < -1, FAIL the test.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- None.

A.26 Transform Coordinate To Default Coordinate System

Name: HelperTransformCoordinateToDefaultCoordinateSystem

Procedure Purpose: Helper procedure to transform coordinate into default coordinate system.

Pre-requisite: None.

Input: Coordinate of point to transform *coordinate*, Scaling value *scale*, Translational value (*translate*).

Returns: Coordinate of point transformed into default coordinate system *coordinate*.

Procedure:

1. Set *coordinate* = *coordinate* * *scale* + *translate*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- None.

A.27 Device Configuration For Create Rule

Name: HelperDeviceConfigurationForCreateRule

Procedure Purpose: Helper procedure for configuring DUT in order to add a rule.

Pre-requisite: Analytics Service is received from the DUT. Rule Engine is supported by the DUT.

Input: None.

Returns: Indication of non-fixed Rule support (*nonFixedRuleSupported*). Analytics configuration token (optional) (*analyticsToken*). Rule description (optional) (*ruleDescription*). Rule to restore (optional) (*ruleToRestore*)

Procedure:

1. ONVIF Client retrieves a list of Analytics Configurations that supports non-fixed Rules by following the procedure mentioned in Annex A.28 with the following input and output parameters

   - out *analyticsConfListSupportsRules* - a list of Analytics configurations

2. If *analyticsConfListSupportsRules* is empty, set *nonFixedRuleSupported* = false and return to test procedure.

3. ONVIF Client configures media profile with Analytics Configuration from *analyticsConfListSupportsRules* list by following the procedure mentioned in Annex A.4 with the following input and output parameters

   - in *analyticsConfListSupportsRules* - list of Analytics configurations.

   - out *profile* - media profile.

4. Set *analyticsToken* := *profile*.Configurations.Analytics.token

5. ONVIF Client retrieves a list of supported rules by following the procedure mentioned in Annex A.29 with the following input and output parameters

   - in *analyticsToken* - a token of Analytics Configuration

   - out *supportedRules* - supported rules.

6. If *supportedRules*.RuleDescription list is empty, FAIL the test, restore the DUT state, and skip other steps.

7. Set *limit* := *supportedRules*.@Limit (if specified).

8. If *supportedRules*.RuleDescription list does not contain item with @fixed != true, FAIL the test, restore the DUT state, and skip other steps.

9. Set *ruleDescription* := first *supportedRules*.RuleDescription item with @fixed != true.

10. If *ruleDescription* does not contain maxInstances attribute, FAIL the test, restore the DUT state, and skip other steps.

11. If *ruleDescription*.@maxInstances = 0, FAIL the test, restore the DUT state, and skip other steps.

12. Set *nonFixedRuleTypeList* := list of RuleDescription.@Name items from *supportedRules* with RuleDescription.@fixed != true.

13. ONVIF Client prepares free space for adding of new rule with required type into the Analytics Configuration by following the procedure mentioned in Annex A.30 with the following input and output parameters

    - in *analyticsToken* - a token of Analytics Configuration

    - in *supportedRules*.@Limit - maximum number of concurrent rules per configuration

    - in *nonFixedRuleTypeList* - list of non-fixed rule types that could be removed

    - in *ruleDescription*.@Name - rule type for which free space is required

    - in *ruleDescription*.@maxInstances - maximum number of rules with *ruleDescription*.@Name type per configuration

    - (optional) in *limit* - limit

    - out *ruleToRestore* - rule to restore (if a rule was removed)

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- None.

A.28 Get List of Analytics Configurations With Supporting of Non-Fixed Rules

Name: HelperGetAnalyticsConfigurationSupportsNonFixedRules

Procedure Purpose: Helper procedure for retrieving full list of Analytics Configurations that supports at least one non-fixed Rule.

Pre-requisite: Analytics Service is received from the DUT. Media2 Service is received from the DUT. Rule Engine is supported by the DUT.

Input: None.

Returns: List of Analytics Configuration that supports non-fixed Rules (*analyticsConfListSupportsRules*).

Procedure:

1. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

   • out *analyticsConfList* - a list of Analytics Configurations

2. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

   2.1. ONVIF Client invokes **GetSupportedRules** request with parameters

      • ConfigurationToken := *analyticsConf*.@token

   2.2. DUT responds with **GetSupportedRulesResponse** message with parameters

      • SupportedRules =: *supportedRules*

   2.3. If *supportedRules*.RuleDescription list is not empty and has at least one RuleDescription with @fixed != true, set *analyticsConfListSupportsRules* := *analyticsConfListSupportsRules* + *analyticsConf*.@token

Procedure Result:

PASS –

   • DUT passes all assertions.

FAIL –

   • DUT did not send **GetSupportedRulesResponse** message.

A.29 Get Supported Rules

Name: HelperGetSupportedRules

Procedure Purpose: Helper procedure for retrieving a list of rule types which are supported by the given analytics configuration.

Pre-requisite: Analytics Service is received from the DUT. Rule Engine is supported by the DUT.

Input: Analytics configuration token (*analyticsConfigToken*).

Returns: Supported Rules (*supportedRules*).

Procedure:

1. ONVIF Client invokes **GetSupportedRules** request with parameters

   • ConfigurationToken := *analyticsConfigToken*

2. DUT responds with **GetSupportedRulesResponse** message with parameters

   • SupportedRules =: *supportedRules*

Procedure Result:

PASS –

   • DUT passes all assertions.

FAIL –

   • DUT did not send **GetSupportedRulesResponse** message.

A.30 Prepare Free Space for Rule

Name: HelperFreeSpaceForRule

Procedure Purpose: Helper procedure for configuring Analytics Configuration with free space in order to add Rule with requested type.

Pre-requisite: Analytics Service is received from the DUT. Rule Engine is supported by the DUT.

Input: Analytics Configuration token (*analyticsConfigToken*). Maximum number of concurrent rules per configuration (*limit*). List of non-fixed rule types (*nonFixedRuleTypeList*). Rule type for which free space is required (*ruleType*). Maximum number of rules with *ruleType* type per configuration (*maxInstances*).

Returns: Rule (optional) (*deletedRule*).

Procedure:

1. ONVIF Client invokes **GetRules** request with parameters

   • ConfigurationToken := *analyticsConfigToken*

2. DUT responds with **GetRulesResponse** message with parameters

   • Rule list =: *ruleList*

3. Set *amountOfExistingRequieredRules* := amount of Rule items in *ruleList* with Type = *ruleType*.

4. If *amountOfExistingRequieredRules* = *maxInstances*

   4.1. Set *ruleToDelete* := the first rule from *ruleList* that has @Type = *ruleType*.

   4.2. ONVIF Client deletes one rule by following the procedure mentioned in Annex A.31 with the following input and output parameters

      • in *analyticsConfigToken* - token of Analytics configuration

      • in *ruleToDelete*.@Name - Name of the rule to be deleted

   4.3. Skip other steps of the procedure and return to the test.

5. If *limit* is not specified, skip other steps of the procedure and return to the test.

6. Set *amountOfExistingRules* := amount of Rule items in *ruleList*.

7. If *amountOfExistingRules* = *limit*

   7.1. If *ruleList* does not contain item with @Type listed in *nonFixedRuleTypeList*, FAIL the test, restore the DUT state, and skip other steps.

   7.2. Set *ruleToDelete* := the first rule from *ruleList* that has @Type listed in *nonFixedRuleTypeList*.

   7.3. ONVIF Client deletes one rule by following the procedure mentioned in Annex A.31 with the following input and output parameters

      • in *analyticsConfigToken* - token of Analytics configuration

      • in *ruleToDelete*.@Name - Name of the rule to be deleted

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.31 Delete Rule

Name: HelperDeleteRule

Procedure Purpose: Helper procedure to delete rule from specified Analytics Configuration.

Pre-requisite: Analytics Service is received from the DUT. Rule Engine is supported by the DUT.

Input: Analytics configuration token (*analyticsToken*). Rule name (*ruleName*).

Returns: None.

Procedure:

1. ONVIF Client invokes **DeleteRules** request with parameters

    - ConfigurationToken := *analyticsToken*

    - RuleName[0] := *ruleName*

2. The DUT responds with **DeleteRulesResponse**.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.32 Create Rule

Name: HelperCreateRule

Procedure Purpose: Helper procedure for rule creation for specified Analytics Configuration.

Pre-requisite: Analytics Service is received from the DUT. Rule Engine is supported by the DUT.

Input: Analytics configuration token (*analyticsToken*). Name of the rule (*ruleName*). Type of the rule (*ruleType*).

Returns: None.

Procedure:

1. ONVIF Client invokes **CreateRules** request with parameters

   - ConfigurationToken := *analyticsToken*

   - Rule[0].@Name := *ruleName*

   - Rule[0].@Type := *ruleType*

   - Rule[0].Parameters := empty Parameters element

2. DUT responds with **CreateRulesResponse** message.

Procedure Result:

PASS –

   - DUT passes all assertions.

FAIL –

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

A.33 Select Existing Rule

Name: HelperSelectRule

Procedure Purpose: Helper procedure for existing rule selection.

Pre-requisite: Analytics Service is received from the DUT. Rule Engine is supported by the DUT.

Input: None.

Returns: Rule (optional) (*rule*). Analytics Configuration token (*analyticsToken*).

Procedure:

1. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

   - out *analyticsConfList* - a list of Analytics configurations

2. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

   2.1. ONVIF Client retrieves a set of rules related to Analytics Configuration by following the procedure mentioned in Annex A.5 with the following input and output parameters

- in *analyticsConf*.@token - a token of Analytics Configuration

- out *ruleList* - rule list

2.2. If *ruleList* list is not empty

2.2.1.Set *rule* := *ruleList*[0].

2.2.2.Set *analyticsToken* := *analyticsConf*.@token.

2.2.3.Skip other steps and return to the test procedure.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- None.

A.34 Device Configuration For Create Specific Rule

Name: HelperDeviceConfigurationForCreateSpecificRule

Procedure Purpose: Helper procedure for configuring DUT in order to add a rule with specified rule type.

Pre-requisite: Analytics Service is received from the DUT. Rule Engine is supported by the DUT.

Input: Rule type (*ruleType*).

Returns: Indication of non-fixed Rule support (*nonFixedRuleSupported*). Analytics configuration token (optional) (*analyticsToken*). Rule description (optional) (*ruleDescription*). Rule to restore (optional) (*ruleToRestore*). Media profile (optional) (*profile*).

Procedure:

1. ONVIF Client retrieves a list of Analytics Configurations that supports non-fixed rule with *ruleType* type by following the procedure mentioned in Annex A.36 with the following input and output parameters

   - in *ruleType* - rule type

   - out *analyticsConfListSupportsRules* - a list of Analytics configurations that support non-fixed reqiured rule type

2. If *analyticsConfListSupportsRules* is empty, set *nonFixedRuleSupported* = false and return to test procedure.

3. ONVIF Client configures media profile with Analytics Configuration from *analyticsConfListSupportsRules* list by following the procedure mentioned in Annex A.4 with the following input and output parameters

   • in *analyticsConfListSupportsRules* - list of Analytics configurations.

   • out *profile* - media profile.

4. Set *analyticsToken* := *profile*.Configurations.Analytics.token

5. ONVIF Client retrieves a list of supported rules by following the procedure mentioned in Annex A.29 with the following input and output parameters

   • in *analyticsToken* - a token of Analytics Configuration

   • out *supportedRules* - supported rules.

6. If *supportedRules*.RuleDescription list does not contain item with @Name = *ruleType* and with @fixed != true, FAIL the test, restore the DUT state, and skip other steps.

7. Set *ruleDescription* := first *supportedRules*.RuleDescription item with @Name = *ruleType* and with @fixed != true.

8. If *ruleDescription* does not contain maxInstances attribute, FAIL the test, restore the DUT state, and skip other steps.

9. If *ruleDescription*.@maxInstances = 0, FAIL the test, restore the DUT state, and skip other steps.

10. Set *nonFixedRuleTypeList* := list of RuleDescription.@Name items from *supportedRules* with RuleDescription.@fixed != true.

11. ONVIF Client prepares free space for adding of new rule with required type into the Analytics Configuration by following the procedure mentioned in Annex A.30 with the following input and output parameters

   • in *analyticsToken* - a token of Analytics Configuration

   • in *supportedRules*.@Limit - maximum number of concurrent rules per configuration

   • in *nonFixedRuleTypeList* - list of non-fixed rule types that could be removed

   • in *ruleType* - rule type for which free space is required

- in *ruleDescription*.@maxInstances - maximum number of rules with *ruleDescription*.@Name type per configuration

- out *ruleToRestore* - rule to restore (if a rule was removed)

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- None.

A.35 Select Value For Rule String Parameter Type

Name: HelperSelectValueForRuleStringParameterType

Procedure Purpose: Helper procedure for select value for string parameter type of a rule.

Pre-requisite: Analytics Service is received from the DUT. Rule Engine is supported by the DUT.

Input: Rule Option of String Parameter Type (*ruleOption*).

Returns: Value for string parameter type *stringValue*.

Procedure:

1. If *ruleOption* does not contain **StringItems** element or **StringList** element, FAIL the test and skip other steps.

2. If *ruleOption* contains **StringItems** element

    2.1.   Set *stringValue* := value of the 1st Item element.

    2.2.   Skip other steps and return to the test procedure.

3. If *ruleOption* contains **StringList** element

    3.1.   Set *stringValue* := value of the 1st element in the string list.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

• None.

A.36 Get List of Non-Fixed Analytics Configurations With Supporting of Specific Rules

Name: HelperGetAnalyticsConfigurationSupportsNonFixedSpecificRules

Procedure Purpose: Helper procedure for retrieving full list of Analytics Configurations that supports at least one non-fixed Rule with specified rule type.

Pre-requisite: Analytics Service is received from the DUT. Media2 Service is received from the DUT. Rule Engine is supported by the DUT.

Input: Rule Type (*ruleType*).

Returns: List of Analytics Configuration that supports non-fixed Rules (*analyticsConfListSupportsRules*).

Procedure:

1. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

   • out *analyticsConfList* - a list of Analytics Configuration that supports non-fixed Rules

2. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

   2.1. ONVIF Client invokes **GetSupportedRules** request with parameters

      • ConfigurationToken := *analyticsConf*.@token

   2.2. DUT responds with **GetSupportedRulesResponse** message with parameters

      • SupportedRules =: *supportedRules*

   2.3. If *supportedRules*.RuleDescription list contains at least one RuleDescription with @Name = *ruleType* and with @fixed != true, set *analyticsConfListSupportsRules* := *analyticsConfListSupportsRules* + *analyticsConf*.@token

Procedure Result:

PASS –

   • DUT passes all assertions.

FAIL –

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

A.37 Supported Parameter Types

ONVIF Client needs to generate values for parameter types Modify Module/Rule request. This table provides a list of rule/module parameter types supported by the ONVIF Client and logic of its values generation.

Table A.1. Supported Parameter Types

Module/Rule Parameter Types	Module/Rule parameter Options Type	Module/Rule Parameter Value to be Generated
xs:boolean (SimpleItemDescription)	skipped	true
xs:string (SimpleItemDescription)	skipped	"test value"
xs:int (SimpleItemDescription)	skipped	1
xs:float (SimpleItemDescription)	skipped	5.4
xs:duration (SimpleItemDescription)	skipped	PT60S
tt:IntRectangle (ElementItemDescription)	skipped	x = 0, y = 0, width = 250, height = 250
tt:StringList (SimpleItemDescription)	skipped	"test"
tt:StringItems (ElementItemDescription)	skipped	"test"
tt:Polyline (ElementItemDescription)	skipped	Point[0].@x="-0 Point[0].@y="0

Module/Rule Parameter Types	Module/Rule parameter Options Type	Module/ Rule Parameter Value to be Generated
		Point[1].@x="0 Point[1].@y="0
tt:Polygon (ElementItemDescription)	skipped	Point[0].@x="-0 Point[0].@y="-0 Point[1].@x="-0 Point[1].@y="0 Point[2].@x="0 Point[2].@y="0
xs:string (SimpleItemDescription)	tt:StringList	Random value from the list.
xs:string (SimpleItemDescription)	tt:StringItems	Random value from the list.
xs:int (SimpleItemDescription)	tt:IntRange	Random value from the range.
xs:int (SimpleItemDescription)	tt:IntList	Random value from the list.
xs:float (SimpleItemDescription)	tt:FloatRange	Random value from the range.

Module/Rule Parameter Types	Module/Rule parameter Options Type	Module/Rule Parameter Value to be Generated
xs:float (SimpleItemDescription)	tt:FloatList	Random value from the list.
xs:duration (SimpleItemDescription)	tt:DurationRange	Random value from the range.
tt:IntRectangle (ElementItemDescription)	tt:IntRectangleRange	Random value from the range for each value, with limitation that the rectangle in the borders of (XRange.Min, YRange.Min, XRange.Min + WidthRange.Ma YRange.Min +

Module/Rule Parameter Types	Module/Rule parameter Options Type	Module/ Rule Parameter Value to be Generated
		HeightRange.M rectangle.
tt:StringList (SimpleItemDescription)	tt:StringList	Random list with values from the list.
tt:StringItems (ElementItemDescription)	tt:StringItems	Random list with values from the list.
tt:Polyline (ElementItemDescription)	tt:IntRange	Random number (based on range) of points based on range with normalized coordinates.
tt:Polygon (ElementItemDescription)	tt:PolygonOptions	If RectangleOnly = true random

Module/Rule Parameter Types	Module/Rule parameter Options Type	Module/ Rule Parameter Value to be Generated
		rectangle. Otherwise, random number (based on range) of points based on range.

A.38 Select Analytics Module Description To Modify

Name: HelperSelectAnalyticsModuleDescriptionToModify

Procedure Purpose: Helper procedure to select Analytics Modules Description that has parameter without corresponding option or that has parameter with corresponding option.

Pre-requisite: Media2 Service is received from the DUT. Analytics Modules is supported by the DUT. Analytics Module Options is supported by the DUT.

Input: Flag shows if parameters of selected Analytics Module have corresponding options *parametersWithOptions*.

Returns: (optional) Analytics Module Description (*analyticsModuleDescription*). (optional) Maximum number of concurrent modules per configuration (*limit*). (optional) Analytics Configurations token (*analyticsConfToken*). (optional) List of SimpleItemDescription/ ElementItemDescription elements that do not have options (*itemListDescription*).

Procedure:

1. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

   • out *analyticsConfList* - a list of Analytics configurations

2. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

    2.1. Set *analyticsConf*.token =: *analyticsConfToken*.

    2.2. ONVIF Client retrieves a list of supported analytics modules by following the procedure mentioned in Annex A.17 with the following input and output parameters

- in *analyticsConfToken* - a token of Analytics Configuration

- out *supportedAnalyticsModules* - supported analytics modules.

    2.3. Set *supportedAnalyticsModules*.AnalyticsModuleDescription list =: *analyticsModuleDescriptionList*

    2.4. For each analytics module description (*analyticsModuleDescription*) in *analyticsModuleDescriptionList* repeat the following steps:

        2.4.1. ONVIF Client retrives analytics module options by following the procedure mentioned in Annex A.39 with the following input and output parameters

- in *analyticsConfToken* - token of Analytics Configuration

- in *analyticsModuleDescription*.@Name - type of Analytics Module

- out *analyticsModuleOptionsList* - options list

        2.4.2. For each SimpleItemDescription/ElementItemDescription element (*itemDescription*) in *analyticsModuleDescription*.Parameters

            2.4.2.1. If *parametersWithOptions* = false

                2.4.2.1. If *itemDescription*.@Type value corresponds to parameter type values provided in the Annex A.37 and *analyticsModuleOptionsList* does not contain option item with @Name = *itemDescription*.@Name, set *itemListDescription* := *itemListDescription* + *itemDescription*

            2.4.2.2. If *parametersWithOptions* = true

                2.4.2.2. If *itemDescription*.@Type value corresponds to parameter type values provided in the Annex A.37 and *analyticsModuleOptionsList* contains corresponding option item with @Name = *itemDescription*.@Name and its type corresponds to value provided in the Annex A.37, set *itemListDescription* := *itemListDescription* + *itemDescription*

        2.4.3. If *itemListDescription* is not empty

2.4.5.1 If *analyticsModuleDescription* does not have maxInstances attribute, FAIL the test, restore the DUT state, and skip other steps.

2.4.5.2 Set *limit* := *supportedAnalyticsModules*.@Limit (if specified).

2.4.5.3 Skip other steps and return to test procedure.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.39 Get Analytics Module Options

Name: HelperGetAnalyticsModuleOptions

Procedure Purpose: Helper procedure to retrieve analytics module options.

Pre-requisite: Analytics Modules is supported by the DUT. Analytics Module Options is supported by the DUT.

Input: Token of Analytics Configuration (*analyticsConfToken*). (optional) Type of Analytics Module (*analyticsModuleType*).

Returns: Options list (*analyticsModuleOptionsList*).

Procedure:

1. ONVIF Client invokes **GetAnalyticsModuleOptions** request with parameters

   - Type := *analyticsModuleType* if specified, otherwise skipped.

   - ConfigurationToken := analyticsConfToken

2. DUT responds with **GetAnalyticsModuleOptionsResponse** message with parameters

   - Options list =: *analyticsModuleOptionsList*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.40 Device Configuration For Modify Module

Name: HelperDeviceConfigurationForModifyModule

Procedure Purpose: Helper procedure to select or create analytics module with specified type.

Pre-requisite: Analytics Modules is supported by the DUT.

Input: Token of Analytics Configuration (*analyticsToken*). Analytics Module Description (*analyticsModuleDescription*).

Returns: Analytics Module with required type *analyticsModule1*.

Procedure:

1. ONVIF Client retrieves a set of analytics modules assigned of Analytics Configuration by following the procedure mentioned in Annex A.20 with the following input and output parameters

    - in *analyticsToken* - a token of Analytics Configuration

    - out *analyticsModuleList* - analytics module list

2. If *analyticsModuleList* contains AnalyticsModule item with @Type = *analyticsModuleDescription*.@Name

    2.1. Set *analyticsModule1* := AnalyticsModule item with @Type = *analyticsModuleDescription*.@Name from *analyticsModuleList*

    2.2. Return to test procedure.

3. If *analyticsModuleList* does not contain AnalyticsModule item with @Type = *analyticsModuleDescription*.@Name and *analyticsModuleDescription*.fixed = true, FAIL the test and skip other steps.

4. ONVIF Client creates new analytics module with Type = *analyticsModuleDescription*.@Name:

    4.1. ONVIF Client prepares free space for adding of new analytics module with required type into the Analytics Configuration by following the procedure mentioned in Annex A.18 with the following input and output parameters

- in *analyticsToken* - a token of Analytics Configuration

- in *analyticsModuleDescription*.@Name - analytics module type

- in *analyticsModuleDescription*.@maxInstances - maximum number of instances per configuration

- in *limit* (optional) - maximum number of concurrent modules per configuration

- out *moduleToRestore* (optional) - analytics module to restore

4.2. Set *moduleName* := generated random string.

4.3. ONVIF Client creates new analytics module by following the procedure mentioned in Annex A.22 with the following input and output parameters

- in *analyticsToken* - a token of Analytics Configuration

- in *moduleName* - analytics module name

- in *analyticsModuleDescription*.@Name - analytics module type

4.4. ONVIF Client retrieves a set of analytics modules assigned of Analytics Configuration by following the procedure mentioned in Annex A.20 with the following input and output parameters

- in *analyticsToken* - a token of Analytics Configuration

- out *analyticsModuleList* - analytics module list

4.5. If *analyticsModuleList* does not contain AnalyticsModule with @Type = *analyticsModuleDescription*.@Name and with @Name = *moduleName*, FAIL the test.

4.6. Set *analyticsModule1* := AnalyticsModule with @Type = *analyticsModuleDescription*.@Name and with @Name = *moduleName* from *analyticsModuleList*.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.41 Select Rule Description To Modify

Name: HelperSelectRuleDescriptionToModify

Procedure Purpose: Helper procedure to select Rules Description that has parameter without corresponding option or that has parameter with corresponding option.

Pre-requisite: Media2 Service is received from the DUT. Rules is supported by the DUT. Rule Options is supported by the DUT.

Input: Flag shows if parameters of selected Rule have corresponding options *parametersWithOptions*.

Returns: (optional) Rule Description (*ruleDescription*). (optional) Maximum number of concurrent rules per configuration (*limit*). (optional) Analytics Configurations token (*analyticsConfToken*). (optional) List of SimpleItemDescription/ElementItemDescription elements that do not have options (*itemListDescription*).

Procedure:

1. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

   • out *analyticsConfList* - a list of Analytics configurations

2. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

   2.1.   Set *analyticsConf*.token =: *analyticsConfToken*.

   2.2.   ONVIF Client retrieves a list of supported rules by following the procedure mentioned in Annex A.29 with the following input and output parameters

      • in *analyticsConfToken* - a token of Analytics Configuration

      • out *supportedRules* - supported rules.

   2.3.   Set *supportedRules*.RuleDescription list =: *ruleDescriptionList*

   2.4.   For each rule description (*ruleDescription*) in *ruleDescriptionList* repeat the following steps:

      2.4.1. ONVIF Client retrives rule options by following the procedure mentioned in Annex A.3 with the following input and output parameters

         • in *analyticsConfToken* - token of Analytics Configuration

- in *ruleDescription*.@Name - type of Rule

- out *ruleOptionsList* - options list

2.4.2.For each SimpleItemDescription/ElementItemDescription element (*itemDescription*) in *ruleDescription*.Parameters

2.4.2.1.If *parametersWithOptions* = false

2.4.2.1.1.If *itemDescription*.@Type value corresponds to parameter type values provided in the Annex A.37 and *ruleOptionsList* does not contain option item with @Name = *itemDescription*.@Name, set *itemListDescription* := *itemListDescription* + *itemDescription*

2.4.2.2.If *parametersWithOptions* = true

2.4.2.2.1.If *itemDescription*.@Type value corresponds to parameter type values provided in the Annex A.37 and *ruleOptionsList* contains corresponding option item with @Name = *itemDescription*.@Name and its type corresponds to value provided in the Annex A.37, set *itemListDescription* := *itemListDescription* + *itemDescription*

2.4.3.If *itemListDescription* is not empty

2.4.3.1.If *ruleDescription* does not have maxInstances attribute, FAIL the test, restore the DUT state, and skip other steps.

2.4.3.2.Set *limit* := *supportedRules*.@Limit (if specified).

2.4.3.3.Skip other steps and return to test procedure.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.42 Device Configuration For Modify Rule

Name: HelperDeviceConfigurationForModifyRule

Procedure Purpose: Helper procedure to select or create rule with specified type.

Pre-requisite: Rules is supported by the DUT.

Input: Token of Analytics Configuration (*analyticsToken*). Rule Description (*ruleDescription*).

Returns: Rule with required type *rule1*.

Procedure:

1. ONVIF Client retrieves a set of rules assigned of Analytics Configuration by following the procedure mentioned in Annex A.5 with the following input and output parameters

   • in *analyticsToken* - a token of Analytics Configuration

   • out *ruleList* - rule list

2. If *ruleList* contains Rule item with @Type = *ruleDescription*.@Name

   2.1.   Set *rule1* := Rule item with @Type = *ruleDescription*.@Name from *ruleList*

   2.2.   Return to test procedure.

3. If *ruleList* does not contain Rule item with @Type = *ruleDescription*.@Name and *ruleDescription*.fixed = true, FAIL the test and skip other steps.

4. ONVIF Client creates new rule with Type = *ruleDescription*.@Name:

   4.1. ONVIF Client prepares free space for adding of new rule with required type into the Analytics Configuration by following the procedure mentioned in Annex A.30 with the following input and output parameters

      • in *analyticsToken* - a token of Analytics Configuration

      • in *ruleDescription*.@Name - rule type

      • in *ruleDescription*.@maxInstances - maximum number of instances per configuration

      • in *limit* (optional) - maximum number of concurrent rules per configuration

      • out *ruleToRestore* (optional) - rule to restore

   4.2. Set *ruleName* := generated random string.

   4.3. ONVIF Client creates new rule by following the procedure mentioned in Annex A.32 with the following input and output parameters

      • in *analyticsToken* - a token of Analytics Configuration

      • in *ruleName* - rule name

• in *ruleDescription*.@Name - rule type

4.4. ONVIF Client retrieves a set of rules assigned of Analytics Configuration by following the procedure mentioned in Annex A.5 with the following input and output parameters

• in *analyticsToken* - a token of Analytics Configuration

• out *ruleList* - rule list

4.5. If *ruleList* does not contain Rule with @Type = *ruleDescription*.@Name and with @Name = *ruleName*, FAIL the test.

4.6. Set *rule1* := Rule with @Type = *ruleDescription*.@Name and with @Name = *ruleName* from *ruleList*.

Procedure Result:

PASS –

• DUT passes all assertions.

FAIL –

• DUT did not send **GetRuleOptionsResponse** message.

A.43 Profile Configuration With Specific Rule Type

Name: HelperProfileConfigurationWithSpecificRuleType

Procedure Purpose: Helper procedure to select or configure media profile with Analytics Configuration that contains rule with specified type.

Pre-requisite: Media2 Service is supported by the DUT. Rules is supported by the DUT.

Input: Rule Type (*ruleType*).

Returns: Token of the video source (*videoSourceToken*). Name of the rule present in the media profile (*ruleName*). Analytics Configuration token present in the media profile (*analyticsToken*). Removed rule to restore (*ruleToRestore*). Created rule to restore (*ruleNameToDelete*).

Procedure:

1. ONVIF Client tries to find media profile that contains rule with Type = *ruleType* by following the procedure mentioned in Annex A.44 with the following input and output parameters

• in *ruleType* - required Rule Type

- (optional) out *profile* - profile with Video Source Configuration, and with Analytics Configuration that contains rule with requested type.

2. If *profile* was returnd at step 1.

   2.1. Set *videoSourceToken* := *profile*.Configurations.VideoSource.@token.

   2.2. Set *ruleName* := *profile*.Configurations.Analytics.RuleEngineConfiguration.Rule[N].@Name, where Rule[N] is the 1st Rule with @Type = *ruleType*.

   2.3. Set *analyticsToken* := *profile*.Configurations.Analytics.@token.

   2.4. Return to test procedure.

3. ONVIF Client tries to find existing rule with Type = *ruleType* by following the procedure mentioned in Annex A.45 with the following input and output parameters

   - in *ruleType* - required Rule Type

   - (optional) out *rule* - Rule

   - (optional) *analyticsToken* - Analytics Configuration token

4. If *rule* was returnd at step 3.

   4.1. ONVIF Client configures media profile with Analytics Configuration by following the procedure mentioned in Annex A.4 with the following input and output parameters

      - in *analyticsToken* - token of Analytics configuration.

      - out *profile* - media profile.

   4.2. Set *videoSourceToken* := *profile*.Configurations.VideoSource.@token.

   4.3. Set *ruleName* := *profile*.Configurations.Analytics.RuleEngineConfiguration.Rule[N].@Name, where Rule[N] is the 1st Rule with @Type = *ruleType*.

   4.4. Return to test procedure.

5. ONVIF Client configures device for adding of new rule with required type by following the procedure mentioned in Annex A.34 with the following input and output parameters

   - in *ruleType* - Rule type

   - out *nonFixedRuleSupported* - Indication of non-fixed Rule support

- out *analyticsToken* - a token of Analytics Configuration

- out *ruleToRestore* - a rule to restore, if it was removed

- out *profile* - media profile with Analytics Configuration

6. If *nonFixedRuleSupported* = false, FAIL the test and skip other steps.

7. Set *videoSourceToken* := *profile*.Configurations.VideoSource.@token.

8. Set *analyticsToken* := *profile*.Configurations.Analytics.@token.

9. Set *ruleName* := generated random string.

10. ONVIF Client creates new rule by following the procedure mentioned in Annex A.32 with the following input and output parameters

- in *analyticsToken* - a token of Analytics Configuration

- in *ruleName* - rule name

- in *ruleType* - rule type

11. Set *ruleNameToDelete* := *ruleName*.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.44 Select Profile With Specific Rule Type

Name: HelperSelectProfileWithSpecificRuleType

Procedure Purpose: Helper procedure to select existing Media Profile that contains Analytics Configuration with required rule type.

Pre-requisite: Media2 Service is received from the DUT. Analytics is supported by the DUT.

Input: Required Rule Type (*ruleType*).

Returns: Media Profile (*profile*) that contains Video Source Configuration, and Analytics Configuration with rule with required type.

Procedure:

1. ONVIF Client invokes **GetProfiles** request with parameters

   • Token skipped

   • Type[0] := VideoSource

   • Type[1] := Analytics

2. The DUT responds with **GetProfilesResponse** message with parameters

   • Profiles list =: *profileList*

3. For each Media Profile (*profile*) in *profileList* with Configuration.Analytics repeat the following steps:

   3.1. If *profile*.Configuration.Analytics contains RuleEngineConfiguration.Rule with @Type = *ruleType*

      3.1.1. If *profile* contains Configuration.VideoSource, skip other steps and return to test procedure.

      3.1.2. ONVIF Client invokes **GetVideoSourceConfigurations** request with parameters

         • ConfigurationToken skipped

         • ProfileToken := *profile*.@token

      3.1.3. The DUT responds with **GetVideoSourceConfigurationsResponse** message with parameters

         • Configurations list =: *vscList*

      3.1.4. If *vscList* is empty, FAIL the test and skip other steps.

      3.1.5. ONVIF Client invokes **AddConfiguration** request with parameters

         • ProfileToken := *profile*.@token

         • Name skipped

         • Configuration[0].Type := VideoSource

         • Configuration[0].Token := *vscList*[0].@token

3.1.6 The DUT responds with **AddConfigurationResponse** message.

3.1.7 Skip other steps and return to test procedure.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

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

A.45 Select Existing Rule With Specific Type

Name: HelperSelectExistingRuleWithSpecificType

Procedure Purpose: Helper procedure to find existing rule with required type.

Pre-requisite: Analytics Service is received from the DUT. Rule Engine is supported by the DUT.

Input: Rule Type (*ruleType*).

Returns: (optional) Rule (*rule*). (optional) Analytics Configuration token (*analyticsToken*).

Procedure:

1. ONVIF Client retrieves a list of Analytics Configurations by following the procedure mentioned in Annex A.1 with the following input and output parameters

   - out *analyticsConfList* - a list of Analytics configurations

2. For each Analytics Configuration *analyticsConf* in *analyticsConfList* repeat the following steps:

   2.1. ONVIF Client retrieves a set of rules related to Analytics Configuration by following the procedure mentioned in Annex A.5 with the following input and output parameters

      - in *analyticsConf*.@token - a token of Analytics Configuration

      - out *ruleList* - rule list

2.2. If *ruleList* contains Rule with @Type = *ruleType*

2.2.1.Set *rule* := Rule with @Type = *ruleType* from *ruleList*.

2.2.2.Set *analyticsToken* := *analyticsConf*.@token.

2.2.3.Skip other steps and return to the test procedure.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- None.

A.46 Rule Engine Message Format Verification

Name: HelperRuleEngineMessageFormatVerification

Procedure Purpose: Helper procedure to verify rule engine message format.

Pre-requisite: None.

Input: Message to be verified (*message*).

Returns: None.

Procedure:

1. If *message* does not contain Source.SimpleItemDescription item with Name = "VideoSource", FAIL the test and skip other steps.

2. If *message*.Source.SimpleItemDescription with Name = "VideoSource" does not have Type = "tt:ReferenceToken", FAIL the test and skip other steps.

3. If *message*.Source contains SimpleItemDescription with Name = "AnalyticsConfiguration" and its Type != "tt:ReferenceToken", FAIL the test and skip other steps.

4. If *message* does not contain Source.SimpleItemDescription item with Name = "Rule", FAIL the test and skip other steps.

5. If *message*.Source.SimpleItemDescription with Name = "Rule" does not have Type = "xs:string", FAIL the test and skip other steps.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.