# ONVIF™
# Analytics Service Specification

Version 20.06
June, 2020

© 2008-2020 by ONVIF: Open Network Video Interface Forum Inc.. All rights reserved.
Recipients of this document may copy, distribute, publish, or display this document so long as this copyright notice, license and disclaimer are retained with all copies of the document. No license is granted to modify this document.
THIS DOCUMENT IS PROVIDED "AS IS," AND THE CORPORATION AND ITS MEMBERS AND THEIR AFFILIATES, MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; OR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.
IN NO EVENT WILL THE CORPORATION OR ITS MEMBERS OR THEIR AFFILIATES BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL DAMAGES, ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THIS DOCUMENT, WHETHER OR NOT (1) THE CORPORATION, MEMBERS OR THEIR AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR (2) SUCH DAMAGES WERE REASONABLY FORESEEABLE, AND ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THIS DOCUMENT. THE FOREGOING DISCLAIMER AND LIMITATION ON LIABILITY DO NOT APPLY TO, INVALIDATE, OR LIMIT REPRESENTATIONS AND WARRANTIES MADE BY THE MEMBERS AND THEIR RESPECTIVE AFFILIATES TO THE CORPORATION AND OTHER MEMBERS IN CERTAIN WRITTEN POLICIES OF THE CORPORATION.

# CONTENTS

**1 Scope 5**

**2 Normative references 5**

**3 Terms and Definitions 5**

3.1 Definitions....5

3.2 Abbreviations ....5

3.3 Namespaces ....5

**4 Overview 6**

**5 Service 8**

5.1 Scene Description Interface ....8
5.1.1 Overview ....8
5.1.2 Frame Related Content ....9
5.1.2.1 Temporal Relation....9
5.1.2.2 Spatial Relation ....10
5.1.3 Scene Elements....11
5.1.3.1 Objects ....12
5.1.3.2 Object Tree ....14
5.1.3.3 Shape descriptor ....16
5.1.3.4 Colour descriptor....16
5.1.3.5 Object Class descriptor ....17
5.1.3.6 Motion In Cells descriptor....18
5.1.3.7 Vehicle information descriptor....20
5.1.3.8 Speed descriptor ....20
5.1.3.9 License plate information descriptor ....20
5.1.3.10 Image Data....21
5.1.3.11 Face descriptor ....21
5.1.3.12 Human body descriptor ....24

5.2 Configuration description language ....25
5.2.1 Configuration parameters ....25
5.2.2 Configuration description ....26
5.2.3 Configuration options....27
5.2.3.1 Overview ....27
5.2.3.2 Option type definition ....27
5.2.3.3 Example ....27

5.3 Rule interface ....28
5.3.1 Rule representation ....28
5.3.2 Rule description language ....29
5.3.3 Operations on rules ....29
5.3.3.1 GetSupportedRules....29
5.3.3.2 GetRules ....29
5.3.3.3 CreateRules ....30
5.3.3.4 ModifyRules ....31
5.3.3.5 DeleteRules....31
5.3.3.6 GetRuleOptions....32

5.4 Analytics modules interface ....33
5.4.1 Analytics module configuration....33
5.4.2 Analytics module description language ....33
5.4.3 Operations on analytics modules....34
5.4.3.1 GetSupportedAnalyticsModules....34
5.4.3.2 GetAnalyticsModules ....34
5.4.3.3 CreateAnalyticsModules ....35
5.4.3.4 ModifyAnalyticsModules....35

5.4.3.5 DeleteAnalyticsModules....36
5.4.3.6 GetAnalyticsModuleOptions....37
5.4.3.7 GetSupportedMetadata....37

5.5 Capabilities....38

5.6 Events....39
5.6.1 Audio Detected....39

**Annex A. Specified Rules (informative) 40**

A.1 LineDetector....40

A.2 Field Detector....41

A.3 Loitering Detector....42

A.4 Declarative motion detector....43

A.5 Counting rule....44

A.6 Object Classification....45

A.7 Query rule....47

**Annex B. Cell motion detection (informative) 48**

B.1 Cell motion detector....48

B.2 Cell motion analytics engine....50
B.2.1 Module configuration....50

**Annex C. Motion detection (normative) 53**

C.1 Motion region detector....53

**Annex D. Radiometry (normative) 55**

D.1 Radiometry Analytics Modules....57
D.1.1 Spot Measurement Module....57
D.1.2 Box Measurement Module....59

D.2 Radiometry Rules....61
D.2.1 Temperature Rule....61

**Annex E. Tampering Detection (normative) 64**

E.1 Tampering Detection....64

**Annex F. Face metadata values samples (informative) 66**

F.1 Face metadata values samples....66

**Annex G. Recognition rule engines (normative) 74**

G.1 Generic parameters for recognition rule engines....74

G.2 Face Recognition....75

G.3 License Plate Recognition....75

**Annex H. Revision History 78**

# 1 Scope

This document defines the web service interface for configuration and operation of video/audio analytics.

Web service usage is outside of the scope of this document. Please refer to the ONVIF core specification.

Since the scope of this service is extended to include audio data, the service is renamed from Video Analytics Service to Analytics Service.

# 2 Normative references

ONVIF Core Specification

<http://www.onvif.org/specs/core/ONVIF-Core-Specification.pdf>

ONVIF Media Service Specification

<http://www.onvif.org/specs/srv/media/ONVIF-Media-Service-Spec.pdf>

ONVIF Streaming Specification

<http://www.onvif.org/specs/stream/ONVIF-Streaming-Spec.pdf>

ISO 12639:2004. Graphic Technology -- Prepress digital data exchange -- Tag image file format for image technology (TIFF/IT). TIFF Revision 6.0

<http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=34342>

ISO 3166-1:2013 Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes

<https://www.iso.org/standard/63545.html>

# 3 Terms and Definitions

## 3.1 Definitions

| | |
|---|---|
| **Video Analytics** | Algorithms used to evaluate video data for meaning of content |
| **Audio Analytics** | Algorithms used to evaluate audio data for meaning of content |

## 3.2 Abbreviations

| | |
|---|---|
| PTZ | Pan Tilt Zoom |

## 3.3 Namespaces

Table 1 lists the prefix and namespaces used in this specification. Listed prefixes are not part of the standard and an implementation can use any prefix.

**Table 1: Namespaces used in this specification**

| Prefix | Namespace URI | Description |
|---|---|---|
| tt | http://www.onvif.org/ver10/schema | XML schema descriptions in this specification. |
| ter | http://www.onvif.org/ver10/error | The namespace for ONVIF defined faults. |
| tns1 | http://www.onvif.org/ver10/topics | The namespace for the ONVIF topic namespace |
| axt | http://www.onvif.org/ver20/analytics | XML schema descriptions for the analytics service |
| ttr | http://www.onvif.org/ver20/analytics/radiometry | XML schema descriptions for Radiometry |
| fc | http://www.onvif.org/ver20/analytics/humanface | XML schema descriptions for human face metadata |
| bd | http://www.onvif.org/ver20/analytics/humanbody | XML schema descriptions for human body metadata |

This specification references to the following namespaces (listed in Table 2) by specified prefix.

**Table 2: Referenced namespaces**

| Prefix | Namespace URI | Description |
|---|---|---|
| xs | http://www.w3.org/2001/XMLSchema | Instance namespace as defined by XS [XML-Schema, Part1] and [XML-Schema, Part 2] |

## 4 Overview

Analytic applications are divided into image/audio analysis and application-specific parts. The interface between these two parts produces an abstraction that describes the scene based on the objects present. The application specific part performs a comparison of the scene descriptions and of the scene rules (such as virtual lines that are prohibited to cross, or polygons that define a protected area). Other rules may represent intra-object behaviour such as objects following other objects (to form a tailgating detection). Such rules can also be used to describe prohibited object motion, which may be used to establish a speed limit.

These two separate parts, referred to as the video/audio analytics engine and as the rule engine, together with the events and actions, form the analytics architecture according to this specification as illustrated in Figure 1.

The analytics architecture consists of elements and interfaces. Each element provides a functionality corresponding to a semantically unique entity of the complete video/audio analytics solution. Interfaces are unidirectional and define an information entity with a unique content. Only the Interfaces are subject to this specification. Central to this architecture is the ability to distribute any elements or sets of adjacent elements to any device in the network.

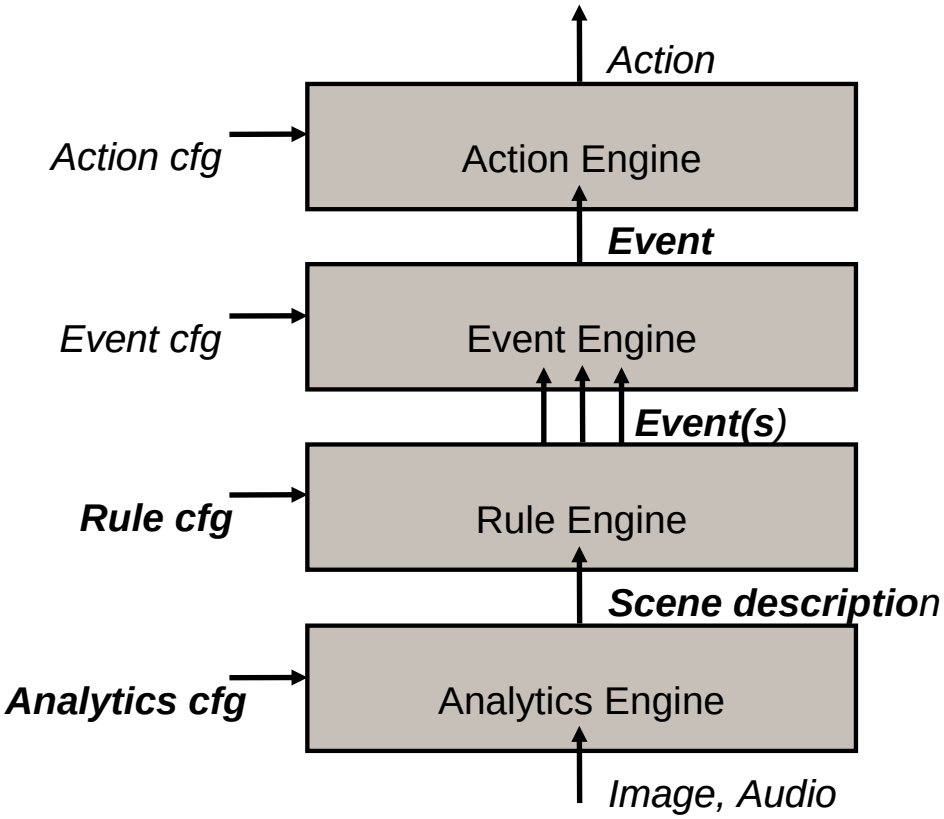



**Figure 1: Analytics architecture**

The following interfaces are defined in this standard:

- Analytics Configuration Interface
- Scene Description
- Rule Configuration Interface

This specification defines a configuration framework for the analytics engine called ONVIF Analytics Service. This framework enables a client to contact a device implementing the ONVIF Analytics Service for supported analytics modules and their configurations. Configurations of such modules can be dynamically added, removed or modified by a client, allowing a client to run multiple Analytics Modules in parallel if supported by the device.

The output from the Analytics Engine is called a *Scene Description*. The Scene Description represents the abstraction of the scene in terms of metadata for the objects, either static or dynamic, that are part of the scene. This specification defines an XML-based Scene Description Interface including data types.

Rules describe how the scene description is interpreted and how to react on that information. The specification defines standard rule syntax and methods to communicate configuration of these rules from the application to the device.

A device supporting ONVIF Analytics Service shall implement the Scene Description Interface and allow events to be dispatched using the Event Service. If the device additionally supports a rule engine then it shall implement the Rules Analytics Modules Interface.

Event and action engine interfaces and configuration is out of scope of this specification. The event interface is handled through the Event Service as described in the ONVIF Core specification.

An analytics configuration can be attached to a Media Profile if the ONVIF Media Service is present. In that case the analytics configuration becomes connected to a specific source.

For server based analytics the ONVIF Analytics Device Service provides for the necessary configuration commands to bundle single analytic algorithm configurations represented as AnalyticsConfiguration to engines or application like processing chains (e.g. all algorithms and rules necessary to build a “lost baggage detector”).

Note that the Media 1 Service Specification uses the name VideoAnalyticsConfiguration for historic reasons which maps to the same entity as the AnalyticsConfiguration.

WSDL for the analytics service is part of the framework and provided in the Analytics WSDL file http://www.onvif.org/ver20/analytics/wsdl/analytics.wsdl.

Rule and Module description reference types that are not already defined in the [ONVIF Schema] are defined in the RULES Schema file http://www.onvif.org/ver20/analytics/wsdl/rules.xsd.

Radiometry types, modules and rules are defined in the RADIOMETRY Schema file http://www.onvif.org/ver20/analytics/radiometry.xsd.

# 5 Service

This section covers the following main areas of this architecture:

- Analytics Module interface
- Scene description
- Rules interface

The analytics service allows fine-grained configuration of individual rules and individual analytics modules (see 5.3 and 5.4). 5.1 introduces the XML-based scene description, which can be streamed as metadata to clients via RTP as defined in the ONVIF Streaming Specification.

## 5.1 Scene Description Interface

### 5.1.1 Overview

This specification defines the XML schema that shall be used to encode Scene Descriptions by a device. The scope of the Scene Description covers basic Scene Elements which can be displayed in a video overlay to the end-user as well as a framework for vendor-specific extensions. Annex A shows additional Scene Elements that may be used for processing vendor-specific rules.

### 5.1.2 Frame Related Content

The input of the analytics engine is images from a video source. The extracted scene elements are associated with the image from which they were extracted. An extracted scene is distinguished from the general description of the video source processed by the analytics engine (information such as video input line, video resolution, frame cropping, frame rate etc.), the temporal frame association within the input stream, and the spatial positioning of elements within a frame.

The temporal and spatial relation of scene elements with respect to the selected video source is discussed in sections 5.1.2.1 and 5.1.2.2. The appearance and behaviour of tracked objects is discussed in section 5.1.3.1. Interactions between objects like splits and merges are described in section 5.1.3.2.

A PTZ device can put information about the pan, tilt and zoom at the beginning of a frame, allowing a client to estimate the 3D coordinates of scene elements. Next, the image coordinate system can be adapted with an optional transformation node which is described in the next subsection. Finally, multiple object descriptions can be placed and their association can be specified within an ObjectTree node. Optionally SceneImageRef and SceneImage can also be included inside Frame. Below, the definitions are included for convenience[1]:

```
<xs:complexType name="Frame">
  <xs:sequence>
    <xs:element name="PTZStatus" type="tt:PTZStatus"
         minOccurs="0"/>
    <xs:element name="Transformation" type="tt:Transformation"
         minOccurs="0"/>
    <xs:element name="Object" type="tt:Object" minOccurs="0"
         maxOccurs="unbounded"/>
    <xs:element name="ObjectTree" type="tt:ObjectTree" minOccurs="0"/>
    <xs:element name="SceneImageRef" type="tt:anyURI" minOccurs="0"/>
    <xs:element name="SceneImage" type="xs:base64Binary" minOccurs="0"/>
    ...
  </xs:sequence>
  <xs:attribute name="UtcTime" type="xs:dateTime" use="required"/>
   ...
</xs:complexType>

<xs:element name="Frame" type="tt:Frame">
```

Subsection 5.1.2.1 describes how frames processed by the analytics algorithm are referenced within the analytics stream.

#### 5.1.2.1 Temporal Relation

Since multiple scene elements can be extracted from the same image, scene elements are listed below a frame node that establishes the link to a specific image from the video input. The frame node contains a mandatory UtcTime attribute. This UtcTime timestamp shall enable a client to map the frame node exactly to one video frame. For example, the RTP timestamp of the corresponding encoded video frame shall result in the same UTC timestamp after conversion. The synchronization between video and metadata streams is further described in the ONVIF Streaming Specification.

Note that there is not necessarily a one to one relation between video and metadata frames. Typically an analytics module generates metadata at a lower frame rate. However when more than one analytics module generates metadata, multiple metadata frames may occur for the same video frame and also the temporal order may vary because different analytics modules may have differing computational delay.

---

[1] Please note that the schema is included here for *information only.* [ONVIF Schema] contains the normative schema definition.

When multiple analytics modules are streaming metadata in interleaved frames than the name of the originating analytics module shall be signaled via the Frame Source attribute.

Example:

```
<tt:Frame UtcTime="2008-10-10T12:24:57.321">
  ...
</tt:Frame>
...
<tt:Frame UtcTime="2008-10-10T12:24:57.521">
  ...
</tt:Frame>
```

#### 5.1.2.2 Spatial Relation

Most scene elements refer to some part in an image from which information has been extracted. For instance, when tracking objects over time, their position within each frame shall be specified. These positions shall relate to a coordinate system. The default coordinate system is shown in Figure 2.

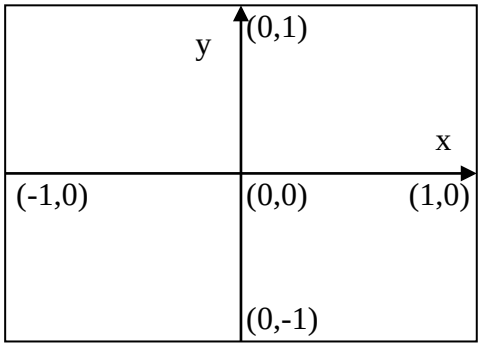



**Figure 2: Default frame coordinate system**

This specification allows modification of the coordinate system for individual nodes of the XML tree. As a result, each frame node starts with the default coordinate system. Each child node inherits the most recent coordinate system of its parent. A transformation node modifies the most recent coordinate system of its parent. Coordinate specifications are always related to the most recent coordinate system of the parent node.

The specification defines transformation nodes for scaling and translation. The Scene Description contains placeholders where these transformation nodes are placed[2].

```
<xs:complexType name="Transformation">
  <xs:sequence>
    <xs:element name="Translate" type="Vector" minOccurs="0"/>
    <xs:element name="Scale" type="Vector" minOccurs="0"/>
    ...
  </xs:sequence>
</xs:complexType>
```

It follows a mathematical description of coordinate systems and transformations. A coordinate transformation consists of a translational vector $t = \begin{pmatrix} t_x \\ t_y \end{pmatrix}$ and scaling $s = \begin{pmatrix} s_x \\ s_y \end{pmatrix}$. A point

[2] Please note that the schema is included here for *information only.* [ONVIF Schema] contains the normative schema definition.

$p = \begin{pmatrix} p_x \\ p_y \end{pmatrix}$ given with respect to this coordinate system is transformed into the corresponding point $q = \begin{pmatrix} q_x \\ q_y \end{pmatrix}$ of the default coordinate system by the following formula: $\begin{pmatrix} q_x \\ q_y \end{pmatrix} = \begin{pmatrix} p_x \cdot s_x + t_x \\ p_y \cdot s_y + t_y \end{pmatrix}$. Similarly, a vector ***v*** given with respect to the coordinate system is transformed into the corresponding vector ***w*** of the default coordinate system by: $\begin{pmatrix} w_x \\ w_y \end{pmatrix} = \begin{pmatrix} v_x \cdot s_x \\ v_y \cdot s_y \end{pmatrix}$.

A transformation node has an optional scaling vector $u = \begin{pmatrix} u_x \\ u_y \end{pmatrix}$ and an optional translational vector $v = \begin{pmatrix} v_x \\ v_y \end{pmatrix}$. If the scaling is not specified, its default value $u = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ is assumed. Similarly, the default value for the translation is $v = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. The transformation node modifies the top-most coordinate system in the following way:

$\begin{pmatrix} t_x^{'} \\ t_y^{'} \end{pmatrix} = \begin{pmatrix} v_x \cdot s_x + t_x \\ v_y \cdot s_y + t_y \end{pmatrix}, \begin{pmatrix} s_x^{'} \\ s_y^{'} \end{pmatrix} = \begin{pmatrix} u_x \cdot s_x \\ u_y \cdot s_y \end{pmatrix}$, where $\begin{pmatrix} t_x^{'} \\ t_y^{'} \end{pmatrix}$ and $\begin{pmatrix} s_x^{'} \\ s_y^{'} \end{pmatrix}$ replace the top-most coordinate system.

For example, the coordinates of the scene description are given in a frame coordinate system, where the lower-left corner has coordinates (0,0) and the upper-right corner coordinates (320,240). The frame node resembles the following code where the scaling is set to the doubled reciprocal of the frame width and the frame height:

```
<tt:Frame UtcTime="2008-10-10T12:24:57.321">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1.0"/>
    <tt:Scale x="0.00625" y="0.00834"/>
  </tt:Transformation>
  ...
</tt:Frame>
```

### 5.1.3 Scene Elements

This section focuses on scene elements generated by object tracking algorithms and defines object handling and object shapes for them.

Frames where no objects have been detected can be skipped within the scene description to save bandwidth, as long as the last frame in the scene description is empty as well. . It is recommended that the device regularly sends the scene description even if it is empty. This is in order to indicate that the analytics engine is operational. The device shall send a scene description if a SynchronizationPoint is requested for the corresponding stream.

When the receiver of a scene description receives an empty frame, the receiver should assume that all subsequent frames are empty as well until the next non-empty frame is received. When the last received frame is non-empty, the receiver should assume that a description of the next processed frame will be transmitted.

#### 5.1.3.1 Objects

Objects are identified via their ObjectID. Features relating to one particular object are collected in an object node with the corresponding ObjectID as an attribute. Associations of objects, such as Object Renaming, Object Splits, Object Merges and Object Deletions are expressed in a separate ObjectTree node. An ObjectID is implicitly created with the first appearance of the ObjectID within an object node[3].

```
<xs:complexType name="ObjectId">
  <xs:attribute name="ObjectId" type="xs:integer"/>
</xs:complexType>

<xs:complexType name="Object">
  <xs:complexContent>
    <xs:extension base="ObjectId">
      <xs:sequence>
        <xs:element name="Appearance" type="Appearance" minOccurs="0"/>
        <xs:element name="Behaviour" type="Behaviour" minOccurs="0"/>
        ...
      </xs:sequence>
      <xs:attribute name="Parent" type="xs:integer">
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The object node has two placeholders for appearance and behaviour information. The appearance node starts with an optional transformation node which can be used to change from a frame-centric coordinate system to an object-centric coordinate system. Next, the Shape of an object can be specified. If an object is detected in a frame, the shape information should be present in the appearance description. The analytics algorithm may add object nodes for currently not visible objects, if it is able to infer information for this object otherwise. In such cases, the shape description may be omitted.

Object features such as shape (see 5.1.3.3), colour (see 5.1.3.4) and object class (see 5.1.3.5) can be added to the appearance node.

This specification defines two standard behaviours for objects: Removed or Idle. These behaviours shall be listed as child nodes of the behaviour node of an object. The presence of a removed or idle node does not automatically delete the corresponding ObjectID, making it possible to reuse the same ObjectID when the object starts moving again.

An object marked with the removed behaviour specifies the place from where the real object was removed. The marker should not be used as the behaviour of the removed object. It is possible to detect the removal even if the action of taking away the object was not detected.

Objects previously in motion can be marked as Idle to indicate that the object stopped moving. As long as such objects don't change, they will not be listed in the scene description anymore. When an Idle object appears again in the Scene Description, the Idle flag is removed automatically.

Example:

```
...
<tt:Frame UtcTime="2008-10-10T12:24:57.321">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1.0"/>
    <tt:Scale x="0.003125" y="0.00416667"/>
  </tt:Transformation>
  <tt:Object ObjectId="12">
    <tt:Appearance>
```

[3] Please note that the schema is included here for *information only.* [ONVIF Schema] contains the normative schema definition.

```
      <tt:Shape>
        <tt:BoundingBox left="20.0" top="30.0" right="100.0" bottom="80.0"/>
        <tt:CenterOfGravity x="60.0" y="50.0"/>
      </tt:Shape>
    </tt:Appearance>
  </tt:Object>
</tt:Frame>
...
<tt:Frame UtcTime="2008-10-10T12:24:57.421">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1.0"/>
    <tt:Scale x="0.003125" y="0.00416667"/>
  </tt:Transformation>
  <tt:Object ObjectId="12">
    <tt:Appearance>
      <tt:Shape>
        <tt:BoundingBox left="20.0" top="30.0" right="100.0" bottom="80.0"/>
        <tt:CenterOfGravity x="60.0" y="50.0"/>
      </tt:Shape>
    </tt:Appearance>
    <tt:Behaviour>
      <tt:Idle/>
    </tt:Behaviour>
  </tt:Object>
</tt:Frame>
...
<tt:Frame UtcTime="2008-10-10T12:24:57.521">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1.0"/>
    <tt:Scale x="0.003125" y="0.00416667"/>
  </tt:Transformation>
</tt:Frame>
...
<tt:Frame UtcTime="2008-10-10T12:24:57.621">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1.0"/>
    <tt:Scale x="0.003125" y="0.00416667"/>
  </tt:Transformation>
  <tt:Object ObjectId="12">
    <tt:Appearance>
      <tt:Shape>
        <tt:BoundingBox left="25.0" top="30.0" right="105.0" bottom="80.0"/>
        <tt:CenterOfGravity x="65.0" y="50.0"/>
      </tt:Shape>
    </tt:Appearance>
  </tt:Object>
</tt:Frame>
...
<tt:Frame UtcTime="2008-10-10T12:24:57.721">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1.0"/>
    <tt:Scale x="0.003125" y="0.00416667"/>
  </tt:Transformation>
  <tt:Object ObjectId="19">
    <tt:Appearance>
      <tt:Shape>
        <tt:BoundingBox left="20.0" top="30.0" right="100.0" bottom="80.0"/>
        <tt:CenterOfGravity x="60.0" y="50.0"/>
      </tt:Shape>
    </tt:Appearance>
    <tt:Behaviour>
      <tt:Removed/>
    </tt:Behaviour>
  </tt:Object>
</tt:Frame>
```

Objects can be related, for example, a LicensePlate object can be related to Vehicle Object to which it belongs, in this case LicensePlate object which is child has the Vehicle's ObjectId as Parent.

Example:

```
<tt:Frame UtcTime="2019-06-10T12:24:57.321">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1.0"/>
    <tt:Scale x="0.003125" y="0.00416667"/>
  </tt:Transformation>
  <tt:Object ObjectId="12">
    <tt:Appearance>
      <tt:Shape>
        <tt:BoundingBox left="20.0" top="30.0" right="100.0"
bottom="180.0"/>
        <tt:CenterOfGravity x="60.0" y="80.0"/>
      </tt:Shape>
      <tt:Class>
           <tt:Type Likelihood="0.9">Vehicle</tt:Type>
        </tt:Class>
    </tt:Appearance>
  </tt:Object>
</tt:Frame>

<tt:Frame UtcTime="2019-06-10T12:24:57.721">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1.0"/>
    <tt:Scale x="0.003125" y="0.00416667"/>
  </tt:Transformation>
  <tt:Object ObjectId="14" Parent="12">
    <tt:Appearance>
      <tt:Shape>
        <tt:BoundingBox left="40.0" top="100.0" right="70.0" bottom="150.0"
/>
        <tt:CenterOfGravity x="57.0" y="130.0" />
      </tt:Shape>
      <tt:Class>
          <tt:Type Likelihood="0.6">LicensePlate</tt:Type>
      </tt:Class>
    </tt:Appearance>
  </tt:Object>
</tt:Frame>
```

#### 5.1.3.2 Object Tree

When two objects come too close to each other, such that the analytics can no longer track them individually, an object merge should be signalled by adding a merge node to the ObjectTree node of the frame node. The merge node contains a From node listing the merging ObjectIds and a To node containing the ObjectId. The merged object is used in future frames as the tracking ID. If the analytics algorithm detects that one object is occluding the others and is able to track this object further, the occluding object should be put in the To node.

The separation of objects is indicated by a Split node. In this case, the From node contains a single ObjectId representing the object which is split in the current frame. The objects separating from this split object are listed in the To node. The ObjectId of the From node can reappear in the To node, if this object did occlude the others and the analytics algorithm was able to track this object during the occlusion.

An object does not need to be involved in a merge operation in order to be part of a split operation. For example, if an object is moving together with a person, and the person leaves the object somewhere, the object might be detected the first time by the analytics when the person moves away from the object left behind. In such cases, the first appearance of the object can be combined with a split operation.

When a merged object reappears as an object node in a later frame without a split indication, then this object is implicitly split. The analytics algorithm, however, could not determine where the split object came from.

An analytics algorithm can track and remember a limited number of objects. In order to indicate that a certain object has been removed from the memory of the algorithm and therefore never appear again, the scene description can contain a Delete node within the ObjectTree node.

If the analytics algorithm can not decide during a Split operation the identity of an object, it should use a new ObjectId. When the algorithm has collected sufficient evidence for the identity of this object, it can change the ObjectId via the Rename operation. The Rename operation can also be used when an object reenters the scene and the true identity is discovered after some time.

A deleted ObjectId shall not be reused within the scene description until the ObjectId container has wrapped around.

Example:

```
<tt:Frame UtcTime="2008-10-10T12:24:57.321">
  <tt:Object ObjectId="12">
    ...
  </tt:Object>
  <tt:Object ObjectId="17">
    ...
  </tt:Object>
</tt:Frame>

<tt:Frame UtcTime="2008-10-10T12:24:57.421">
  <tt:Object ObjectId="12">
    ...
  </tt:Object>
  <tt:ObjectTree>
    <tt:Merge>
      <tt:From ObjectId="12"/>
      <tt:From ObjectId="17"/>
      <tt:To ObjectId="12"/>
    </tt:Merge>
  </tt:ObjectTree>
</tt:Frame>

<tt:Frame UtcTime="2008-10-10T12:24:57.521">
  <tt:Object ObjectId="12">
    ...
  </tt:Object>
</tt:Frame>

<tt:Frame UtcTime="2008-10-10T12:24:57.621">
  <tt:Object ObjectId="12">
    ...
  </tt:Object>
  <tt:Object ObjectId="17">
    ...
  </tt:Object>
  <tt:ObjectTree>
    <tt:Split>
      <tt:From ObjectId="12"/>
      <tt:To ObjectId="17"/>
      <tt:To ObjectId="12"/>
    </tt:Split>
  </tt:ObjectTree>
</tt:Frame>
```

#### 5.1.3.3 Shape descriptor

Shape information shall be placed below the optional shape node of in an object appearance node. If present, the shape node holds information where the object under consideration has been detected in the specified frame. A shape node shall at least contain two nodes representing the BoundingBox and the CenterOfGravity of the detected object.

The coarse BoundingBox is further refined with additional child nodes, each representing a shape primitive. If multiple shape primitives are present, their union defines the object's shape. In this specification, a generic polygon descriptor is provided.

Polygons that describe the shape of an object shall be simple polygons defined by a list of points.

Two consecutive points (where the last point is connected with the first one) in the list define a line segment. The order of the points shall be chosen such that the enclosed object region can be found on the left-hand side all line segments. The polyline defined by the list of points shall not be self-intersecting.

Example:

```
<tt:Frame UtcTime="2008-10-10T12:24:57.321">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1".0/>
    <tt:Scale x="0.003125" y="0.00416667"/>
  </tt:Transformation>
  <tt:Object ObjectId="12">
    <tt:Appearance>
      <tt:Shape>
        <tt:BoundingBox left="20.0" top="30.0" right="100.0" bottom="80.0"/>
        <tt:CenterOfGravity x="60.0" y="50.0"/>
        <tt:Polygon>
          <tt:Point x="20.0" y="30.0"/>
          <tt:Point x="100.0" y="30.0"/>
          <tt:Point x="100.0" y="80.0"/>
          <tt:Point x="20.0" y="80.0"/>
        </tt:Polygon>
      </tt:Shape>
    </tt:Appearance>
  </tt:Object>
</tt:Frame>
```

#### 5.1.3.4 Colour descriptor

A colour descriptor is defined as an optional element of the appearance node of an object node. The colour descriptor is defined by a list of colour clusters, each consisting of a colour value, an optional weight and an optional covariance matrix. The colour descriptor does not specify, how the colour clusters are created. They can represent bins of a colour histogram or the result of a clustering algorithm.

Colours are represented by three-dimensional vectors. Additionally, the colourspace of each colour vector can be specified by a colourspace attribute. If the colourspace attribute is missing, the YcbCr colourspace is assumed. It refers to the 'sRGB' gamut with the RGB to YcbCr transformation as of ISO/IEC 10918-1 (Information technology -- Digital compression and coding of continuous-tone still images: Requirements and guidelines), a.k.a. JPEG. The Colourspace URI for the YcbCr colourspace is www.onvif.org/ver10/colorspace/YcbCr.

An example metadata containing colour information of the detected object is given below.

```
<tt:Frame UtcTime="2010-09-10T12:24:57.721">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1.0"/>
```

```
    <tt:Scale x="0.003125" y="0.00416667"/>
  </tt:Transformation>
  <tt:Object ObjectId="34">
    <tt:Appearance>
     <tt:Shape>
       <tt:BoundingBox left="20.0" top="30.0" right="100.0" bottom="80.0"/>
       <tt:CenterOfGravity x="60.0" y="50.0"/>
     </tt:Shape>
     <tt:Color>
      <tt:ColorCluster>
       <tt:Color X=”58” Y=”105” Z=”212”/>
       <tt:Covariance XX=”7.2” YY=”6” ZZ=”3”/>
       <tt:Weight>90</tt:Weight>
      </tt:ColorCluster>
      <tt:ColorCluster>
       <tt:Color X=”165” Y=”44” Z=”139”/>
       <tt:Covariance XX=”4” YY=”4” ZZ=”4”/>
       <tt:Weight>5</tt:Weight>
      </tt:ColorCluster>
     </tt:Color>
    </tt:Appearance>
  </tt:Object>

</tt:Frame>
```

Colour descriptor contains the representative colours in detected object/region. The representative colours are computed from image of detected object/region each time. Each representative colour value is a vector of specified colour space. The representative colour weight (Weight) denotes the fraction of pixels assigned to the representative colour. Colour covariance describes the variation of colour values around the representative colour value in colour space thus representative colour denotes a region in colour space. The following table lists the acceptable values for Colourspace attribute

**Table 3 Colourspace namespace values**

| Namespace URI | Description |
|---|---|
| http://www.onvif.org/ver10/colorspace/YcbCr | YcbCr colourspace |
| http://www.onvif.org/ver10/colorspace/CIELUV | CIE LUV |
| http://www.onvif.org/ver10/colorspace/CIELAB | CIE 1976 (L*a*b*) |
| http://www.onvif.org/ver10/colorspace/HSV | HSV colourspace |

#### 5.1.3.5 Object Class descriptor

A Class Descriptor is defined as an optional element of the appearance node of an abject node. The class descriptor is defined by a list of object classes together with a likelihood that the corresponding object belongs to this class. The sum of the likelihoods shall not exceed 1.

The following example shows an object metadata sample that contains object class information about a detected object:

```
<tt:Frame UtcTime="2010-11-10T12:24:57.721">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1.0"/>
    <tt:Scale x="0.003125" y="0.00416667"/>
  </tt:Transformation
  <tt:Object ObjectId="22">
   <tt:Appearance>
    <tt:Shape>
      <tt:BoundingBox left="20.0" top="30.0" right="100.0" bottom="80.0"/>
      <tt:CenterOfGravity x="60.0" y="50.0"/>
    </tt:Shape>
```

```
    <tt:Class>
      <tt:Type Likelihood=”0.8”>Vehicle</tt:Type>
    </tt:Class>
   </tt:Appearance>
  </tt:Object>
</tt:Frame>
```

Beside the small set of predefined class types free type definitions can be added using the Type member defined as StringLikelihood.

#### 5.1.3.6 Motion In Cells descriptor

The scene description of a cell motion contains the cells where motion is detected. To decode base64Binary the columns and rows of the cell grid is provided.

For spatial relation the "Transformation" element is used, see section 5.1.2.2. The cell grid is starting from the upper left corner. The X dimension is going from left to right and the Y dimension is going from up to down, see Figure B-1.

This example contains a snippet of a metadata stream using the Cell Motion Detector.

```
<?xml version="1.0" encoding="UTF-8"?>
<tt:MetaDataStream
  xmlns:tt="http://www.onvif.org/ver10/schema">

  <tt:VideoAnalytics>
    <tt:Frame UtcTime="2010-10-20T12:24:57.321">
     <tt:Transformation>
       <tt:Translate x="-0.66666" y="-0.6"/>
       <tt:Scale x="0.1666666" y="-0.2"/>
     </tt:Transformation>
     <tt:Extension>
       <tt:MotionInCells Columns="8" Rows="6" Cells="AAD8AA==”/>
     </tt:Extension>
    </tt:Frame>
    <tt:Frame UtcTime="2010-10-20T12:24:57.621">
      ...
    </tt:Frame>
  </tt:VideoAnalytics>

  <tt:Event>
    <wsnt:NotficationMessage>
      <wsnt:Topic Dialect="...Concrete">
       tns1:RuleEngine/CellMotionDetector/Motion
      </wsnt:Topic>
      <wsnt:Message>
        <tt:Message UtcTime= "2010-10-20T12:24:57.628">
          <tt:Source>
           <tt:SimpleItem Name="VideoSourceConfigurationToken" Value="1"/>
           <tt:SimpleItem Name="VideoAnalyticsConfigurationToken" Value="1"/>
           <tt:SimpleItem Name="Rule" Value="MotionInDefinedCells"/>
          </tt:Source>
          <tt:Data>
           <tt:SimpleItem Name="IsMotion" Value="true"/>
          </tt:Data>
        </tt:Message>
      </wsnt:Message>
    </wsnt:NotficationMessage>
  </tt:Event>

</tt:MetaDataStream>
```

**Table 4 Description of attributes of MotionInCells type**

| Attribute | Description |
|---|---|
| Columns | Number of columns of the cell grid (x dimension) |
| Rows | Number of rows of the cell grid (y dimension). |
| Cells | A “1” denotes a cell where motion is detected and a “0” an empty cell.<br><br>The first cell is in the upper left corner. Then the cell order goes first from left to right and then from up to down (see Figure B-1).<br><br>If the number of cells is not a multiple of 8 the last byte is filled with zeros.<br><br>The information is run-length encoded according to Packbit coding in ISO 12639 (TIFF, Revision 6.0). |

#### 5.1.3.7 Vehicle information descriptor

The vehicle information descriptor defines an optional extension of the object appearance node. It is used to describe other features of the vehicle, like type, brand and model. The details are shown in Table 5. For each element, there is a class, which is asscocaited with a likelihood. A likelihood/probability means the corresponding object belongs to this class. The default value is 1. Missing likelihood attribute means 100%.

**Table 5 Description of the vehicle detail**

| VehicleInfo | Description |
|---|---|
| Type | Define the vehicle's type, including Car, Truck, Bus, Motorcycle, Bicycle and Others. |
| Brand | Describe the vehicle's brand which could be the same as its logo. |
| Model | Describe the vehicle's model which can give more detail information of the vehicle brand. |

#### 5.1.3.8 Speed descriptor

The SpeedDescriptor describes the speed value of the object. The unit is meters per second.

An example metadata containing speed information of the detected object is given below.

```
<tt:VideoAnalytics>
  <tt:Frame UtcTime="2016-9-10T12:24:57.321">

    <tt:Object ObjectId="23">
      <tt:Behaviour>
        <tt:Speed>120.5</tt:Speed>
      </tt:Behaviour>
    </tt:Object>
  </tt:Frame>
</tt:VideoAnalytics>
```

#### 5.1.3.9 License plate information descriptor

The license plate detail descriptor defines an optional extension of the object appearance node. It is used to extend the details of the vehicle license plate information. The details are shown in Table 6. For each element, there is a class, which is asscociated with a likelihood. A likelihood/probability means the corresponding object belongs to this class. The default value is 1. Missing likelihood attribute means 100%.

**Table 6 Description of the License plate detail**

| LicensePlateInfo | Description |
|---|---|
| PlateNumber | A string of vehicle license plate number,e.g., "6DZG261", "756 JWB", and etc. |
| PlateType | Description of the vehicle license plate type, e.g., "Normal", "Police", "Trainning". |
| CountryCode | Describes the country of the license plate. The country code shall be encoded as two letter code according to ISO 3166-1:2013 Alpha-2. |

| | |
|---|---|
| | Refer to the ISO 3166-1 alpha-2 code of the country,e.g.,"CN" means China,"US" means United States,"JP"means Japan. |
| IssuingEntity | State province or authority that issue the license plate. |

An example metadata containing the license plate information of the detected object is given below.

```
<tt:VideoAnalytics>
  <tt:Frame UtcTime="2010-10-20T12:24:57.321">

    <tt:Object ObjectId="12">
      <tt:Appearance>
        <tt:LicensePlateInfo>
          <tt:PlateNumber>6DZG261</tt:PlateNumber>
          <tt:PlateType>Normal</tt:PlateType>
          <tt:CountryCode>US</tt:CountryCode>
          <tt:IssuingEntity>California</tt:IssuingEntity>
        </tt:LicensePlateInfo>
      </tt:Appearance>
    </tt:Object>
  </tt:Frame>
</tt:VideoAnalytics>
```

#### 5.1.3.10 Image Data

Frames where objects have been detected can also carry image data within the Scene Description. Image Data relating to a detected object are added in the Appearance Node. Below, the definitions of Image and ImageRef are given for convenience:

```
<xs:complexType name="Appearance">
  <xs:sequence>
    <xs:element name="Transformation" type="tt:Transformation" minOccurs="0"/>
    <xs:element name="Shape" type="tt:ShapeDescriptor" minOccurs="0"/>
    <xs:element name="Color" type="tt:ColorDescriptor" minOccurs="0"/>
    <xs:element name="Class" type="tt:ClassDescriptor" minOccurs="0"/>
    <xs:element name="Extension" type="tt:AppearanceExtension" minOccurs="0"/>
    <xs:element name="GeoLocation" type="tt:GeoLocation" minOccurs="0"/>
    <xs:element name="VehicleInfo" type="tt:VehicleInfo" minOccurs="0"/>
    <xs:element name="LicensePlateInfo" type="tt:LicensePlateInfo" minOccurs="0"/>
    <xs:element name="ImageRef" type="xs:anyURI" minOccurs="0"/>
    <xs:element name="Image" type="xs:base64Binary" minOccurs="0"/>
    <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
</xs:complexType>
```

#### 5.1.3.11 Face descriptor

Face descriptor is defined as another optional element of an Object Node. It is used to describe the other features of the face, e.g., facial shape, hair, eye. Annex F provides samples of several face elements.

**Table 7 Description of the Human Face detail**

| Feature | Sub feature | Type | Defined Value |
|---|---|---|---|
| Age | N/A | tt:IntRange | N/A |

| | | | |
|---|---|---|---|
| Gender | N/A | xs:string | fc:Gender |
| Temperature | N/A | xs:float | N/A |
| Complexion | N/A | xs:string | fc:Complexion |
| FacialShape | N/A | xs:string | fc:FacialShape |
| Hair | length | xs:string | fc:Length |
| | Style | xs:string | fc:HairStyle |
| | Color | xs:string | tt: ColorDescriptor |
| | Bangs | xs:boolean | N/A |
| Eyebrow | Width | xs:string | fc:EyebrowWidth |
| | Color | xs:string | tt: ColorDescriptor |
| | Space | xs:string | fc:EyebrowSpace |
| Eye | Shape | xs:string | fc:EyeShape |
| | Eyelid | xs:string | fc:Eyelid |
| | Eyeball | xs:string | fc:Eyeball |
| Ear | N/A | xs:string | fc:Ear |
| Nose | Length | xs:string | fc:NoseLength |
| | NoseBridge | xs:string | fc:NoseBridge |
| | NoseWing | xs:string | fc:NoseWing |
| | NoseEnd | xs:string | fc:NoseEnd |
| FacialHair | Mustache | xs:boolean | N/A |
| | Beard | xs:boolean | N/A |
| | Sideburn | xs:boolean | N/A |
| Lip | N/A | xs:string | fc:Lip |
| Chin | N/A | xs:string | fc:Chin |
| PoseAngle | PoseAngle | tt:GeoOrientation | N/A |
| | Uncertainty | tt:GeoOrientation | N/A |

| | | | |
|---|---|---|---|
| Accessory | Opticals | fc: AccessoryDescription | N/A |
| | Hat | fc: AccessoryDescription | N/A |
| | Mask | fc: AccessoryDescription | N/A |
| | Hijab | fc: AccessoryDescription | N/A |
| | Helmet | fc: AccessoryDescription | N/A |
| | Kerchief | fc: AccessoryDescription | N/A |
| | RightEyePatch | fc: AccessoryDescription | N/A |
| | LeftEyePatch | fc: AccessoryDescription | N/A |
| AdditionalFeatures | Scar | xs:boolean | N/A |
| | Mole | xs:boolean | N/A |
| | Tatoo | xs:boolean | N/A |
| | Freckles | xs:string | fc:FrecklesType |

PoseAngle represents the estimated pose of the face, two elements are included in PoseAngle field. One is PoseAngles which includes three angles yaw, pitch, roll, the other one is the pose angle uncertainty, uncertainty describes the expected degree of uncertainty of each pose angle.

Yaw angle: rotation about the vertical(y) axis.

Pitch angle: rotation about the horizontal side-to-side(x) horizontal axis.

Roll angle: rotation about the horizontal back to front(z) axis.

The range of yaw, pitch and roll is between -180 and +180 degrees. Pose angles are defined relative to the frontal view of the subject. Angle(0,0,0) is shown in Figure 3.

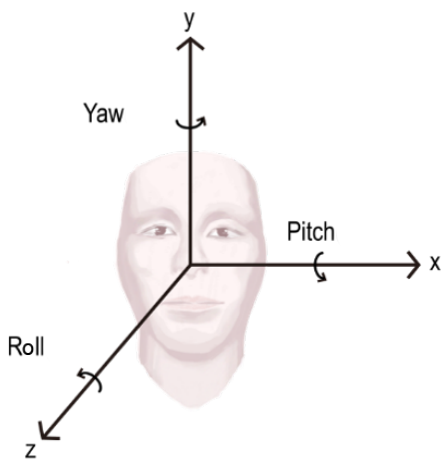



**Figure 3: Pose angles with respect to the frontal view of the subject**

#### 5.1.3.12 Human body descriptor

A human body descriptor is defined as an optional element of the Appearance Node of an Object Node. It is exploited to describle some features of the humanbody, such as bodymetric, clothing, belonging and behaviour. All features are listed in the below table.

**Table 8 Description of the Human Body detail**

| Feature | Sub Feature | Element | Type | Defined Value |
|---|---|---|---|---|
| BodyMetric | Height | N/A | xs:int | N/A |
| | BodyShape | N/A | xs:string | bd:BodyShape |
| Clothing | Scarf | Color | tt:ColorDescriptor | N/A |
| | | Wear | xs:boolean | N/A |
| | Gloves | Color | tt:ColorDescriptor | N/A |
| | | Wear | xs:boolean | N/A |
| | Tops | Category | xs:string | bd:TopsCategory |
| | | Color | tt:ColorDescriptor | N/A |
| | | Grain | xs:string | bd:Grain |
| | | Style | xs:string | bd:TopsStyle |
| | Bottoms | Category | xs:string | bd:BottomsCategory |
| | | Color | xs:string | N/A |
| | | Grain | xs:string | bd:Grain |
| | | Style | xs:string | bd:BottomsStyle |
| | Shoes | Category | xs:string | bd:ShoesCategory |
| | | Color | xs:string | N/A |
| Belonging | Bag | Category | xs:string | bd:KnapsackCategory |
| | | Color | xs:string | N/A |
| | Umbrella | Color | xs:string | N/A |

| | | | | |
|---|---|---|---|---|
| | | Open | xs:boolean | N/A |
| | LiftSomething | N/A | xs:boolean | N/A |
| | Box | Color | xs:string | N/A |
| | | Lug | xs:boolean | N/A |
| | Cart | Category | xs:string | bd:CartCategory |
| | | Color | tt:ColorDescriptor | N/A |
| | Weapon | N/A | xs:boolean | N/A |
| Behaviour | Smoking | N/A | xs:string | bd:Smoking |
| | UsingMobile | N/A | xs:string | bd:UsingMobile |
| | Activity | Walking | N/A | xs:string |
| | | Running | N/A | xs:string |
| | | Fallen | N/A | xs:string |
| | | Squatting | N/A | xs:string |
| | | Sitting | N/A | xs:string |
| | | Standing | N/A | xs:string |
| | | Driving | N/A | xs:string |

## 5.2 Configuration description language

This specification introduces a description language which is used to configure rules and analytics modules. Beside configuration parameters the config description additionally contains an event description according to the OASIS topic notification.

Implementations can define their own configuration types defined using the configuration descriptions as specified in section 5.2.2. The provided APIs allow devices and clients to instantiate one or more instances of the types available on a device. Each of the instances can be configured with the parameters given in 5.2.1 and their options defined in 5.2.3.

### 5.2.1 Configuration parameters

Each configuration has two required attributes: one specifies the name and the other specifies the type of the rule. The type refers to the name property of a description.

The different configuration parameters are listed below the parameters element of the rule element. Each parameter is either a SimpleItem or an ElementItem. The name attribute of each item shall be unique within the parameter list. SimpleItems have an additional Value

attribute containing the value of the parameter. The value of ElementItems is given by the child element of the ElementItem. It is recommended to represent as many parameters as possible with SimpleItems.

In case of the ElementItem, the type attribute shall reference a global element declaration of an XML schema.

### 5.2.2 Configuration description

A description has a name attribute with a name provided by the device vendor.

The description of a configuration contains the type information for all parameters belonging to a configuration as well as the description of the output produced by such a rule. The output of the Rule Engine are events which can either be used in an Event Engine or be subscribed to by a client.

All parameters are defind as SimpleItems or ElementItems and are respectibely described by SimpleItemDescription or ElementItemDescription. Both item descriptions contain a name attribute to identify the parameter and a type attribute to reference a specific XML schema type. In case of the SimpleItemDescription, the type attribute shall reference a SimpleType schema definition. In case of the ElementItemDescription, the type attribute shall reference a global element declaration of an XML schema.

Below, the definitions are included for convenience[4]:

```
<xs:complexType name="ConfigDescription">
  <xs:sequence>
    <xs:element name="Parameters"
                type="tt:ItemListDescription"/>
    <xs:element name="Messages" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
       <xs:complexContent>
          <xs:extension base="tt:MessageDescription">
            <xs:sequence>
              <xs:element name="ParentTopic" type="xs:string"/>
            </xs:sequence>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
      </xs:element>
    ...
  </xs:sequence>
  <xs:attribute name="Name" type="xs:QName" use="required"/>
</xs:complexType>

<xs:complexType name="ItemListDescription">
  <xs:sequence>
    <xs:element name="SimpleItemDescription" minOccurs="0"
                maxOccurs="unbounded">
      <xs:complexType>
        <xs:attribute name="Name" type="xs:string" use="required"/>
        <xs:attribute name="Type" type="xs:Qname" use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="ElementItemDescription" minOccurs="0"
                maxOccurs="unbounded">
      <xs:complexType>
        <xs:attribute name="Name" type="xs:string" use="required"/>
        <xs:attribute name="Type" type="xs:Qname" use="required"/>
```

[4] Please note that the schema is included here for *information only.* [ONVIF Schema] contains the normative schema definition.

```
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

### 5.2.3 Configuration options

#### 5.2.3.1 Overview

This section defines a generic mechanism for signaling valid parameter configuration values for simple element items via the GetRuleOptions and GetAnalyticsModuleOptions methods.

#### 5.2.3.2 Option type definition

Table 9 defines how to describe any static or dynamic limitations for generic types through the GetRuleOptions and GetAnalyticsModule options.

**Table 9 Generic constraint definitions for parameter data types**

| Parameter Type | Option Type | Limitation |
|---|---|---|
| xs:string | tt:StringList | Specified string options (no whitespace allowed) |
| xs:string | tt:StringItems | Specific string options (whitespace allowed) |
| xs:int | tt:IntRange | Min and Max value |
| xs:int | tt:IntList | Specific values |
| xs:float | tt:FloatRange | Min and Max value |
| xs:float | tt:FloatList | Specific values |
| xs:duration | tt:DurationRange | Min and Max value |
| tt:IntRectangle | tt:IntRectangleRange | Min and Max values for each x, y, width and height attribute |
| tt:StringList | tt:StringList | Parameter and options without whitespace |
| tt:StringItems | tt:StringItems | Parameter and options may contain whitespace |
| tt:Polyline | tt:IntRange | Number of points supported Min and Max value |
| tt:Polygon | tt: PolygonOptions | Specifies polygon support and number of points supported |

Note that the option type in Table 9 refers to the element definition. In case parameters occur with differing constraints in multiple rules or analytics modules the device shall provide the optional parameters RuleType or AnalyticsModule.

#### 5.2.3.3 Example

This examples explains the options for an analytics module called myModule in namesapce nn with four simple parameters:

```
<tan:AnalyticsModuleDescription Name="nn:myModule" fixed="true"
maxInstances="1">
    <tt:Parameters>
        <tt:SimpleItemDescription Name="Mode" Type="xs:string"/>
        <tt:SimpleItemDescription Name="Targets" Type="xs:string"/>
        <tt:SimpleItemDescription Name="Speed" Type="xs:int"/>
        <tt:SimpleItemDescription Name="Latency" Type="xs:duration"/>
    </tt:Parameters>
</tan:AnalyticsModuleDescription>
```

The implementation has the following restrictions for the parameters:

- Mode is either ‘Motion Detection’ or ‘Object Tracking’
- Targets is either 'Car', 'Bike' or 'Pedestrian'
- Speed supports the range between zero and 120
- Latency supports the range between one second and oneminute

The following code snipped shows the options returned by GetAnalyticsModuleOptions for myModule:

```
<tan:Options AnalyticsModule ="nn:myModule" Name="Mode">
  <tt:StringItems>
    <tt:Item>Motion Detection</tt:Item>
    <tt:Item>Object Tracking</tt:Item>
  </tt:StringItems>
</tan:Options>
<tan:Options AnalyticsModule="nn:myModule" Name="Targets">
  <tt:StringList>Car Bike Pedestrian</tt:StringList>
</tan:Options>
<tan:Options AnalyticsModule ="nn:myModule" Name="Speed">
  <tt:IntRange>
    <tt:Min>0</tt:Min>
    <tt:Max>120</tt:Max>
  </tt:IntRange >
</tan:Options>
<tan:Options AnalyticsModule ="nn:myModule" Name="Latency">
  <tt:DurationRange>
    <tt:Min>PT1S</tt:Min>
    <tt:Max>PT1M</tt:Max>
  </tt:DurationRange>
</tan:Options>
```

A device only supporting three concrete values for the speed parameter may instead return for the respective option:

```
<tan:Options AnalyticsModule ="nn:myModule" Name="Speed">
  <tt:IntList>15 30 80</tt:IntList>
</tan:Options>
```

## 5.3 Rule interface

An XML structure is introduced in Section 5.3.1 to communicate the configuration of rules. Section 5.3.2 specifies a language to describe the configuration of a specific rule type. A device implementing a rule engine can support rules described in Appendix A. Section 5.3.3 introduces operations to manage rules. If the device supports a Rule Engine, it shall implement the complete rule interface.

### 5.3.1 Rule representation

The rule representation uses the configuration language as defined in section 5.2.1. The following example shows a complete analytics configuration containing two rules:

```
<tt:RuleEngineConfiguration>
  <tt:Rule Name="MyLineDetector" Type="tt:LineDetector">
    <tt:Parameters>
      <tt:SimpleItem Name="Direction" Value="Any"/>
      <tt:ElementItem Name="Segments">
        <tt:Polyline>
          <tt:Point x="10.0"  y="50.0"/>
          <tt:Point x="100.0" y="50.0"/>
        </tt:Polyline>
      </tt:ElementItem>
    </tt:Parameters>
  </tt:Rule>
  <tt:Rule Name="MyFieldDetector" Type="tt:FieldDetector">
    <tt:Parameters>
      <tt:ElementItem Name="Field">
        <tt:Polygon>
          <tt:Point x="10.0"  y="50.0"/>
          <tt:Point x="100.0" y="50.0"/>
```

```
            <tt:Point x="100.0" y="150.0"/>
          </tt:Polygon>
        </tt:ElementItem>
      </tt:Parameters>
    </tt:Rule>
  </tt:RuleEngineConfiguration>
```

### 5.3.2 Rule description language

Rules are described using the rule description language defined in section 5.2.2.

The output produced by this rule type is described in multiple Messages elements. Each Messages contains a description of the message payload according to the Message Description Language detailed in the ONVIF Core specification. Additionally, the Messages shall contain a ParentTopic element naming the Topic a client has to subscribe to in order to receive this specific output. The topic shall be specified as a Concrete Topic Expression.

### 5.3.3 Operations on rules

The Create/Delete/Modify operations are atomic, meaning that either all modifications can be processed or the complete operation shall fail.

#### 5.3.3.1 GetSupportedRules

A device signaling support for rules via the RuleSupport capability shall support this operation. It returns a list of rule descriptions according to the Rule Description Language described in Section 5.3.2. Additionally, it contains a list of URLs that provide the location of the schema files. These schema files describe the types and elements used in the rule descriptions. Rule descriptions that reference types or elements imported from any ONVIF defined schema files need not explicitly list those schema files. The device shall indicate its limit for maximum number of rules through the maxInstances attribute.

REQUEST:

- **ConfigurationToken [tt:ReferenceToken]**
  Token of an existing analytics configuration.

RESPONSE:

- **SupportedRules [tt:SupportedRules]**
  Rules supported for the analytics configuration

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoConfig**
  The requested analytics configuration does not exist.

ACCESS CLASS:

**READ_MEDIA**

#### 5.3.3.2 GetRules

A device signaling support for rules via the RuleSupport capability shall support this operation to retrieve the currently associated rules with a video analytics configuration.

REQUEST:

- **ConfigurationToken [tt:ReferenceToken]**
  Token of an existing analytics configuration.

RESPONSE:

- **Rule - optional, unbounded [tt:Config]**
  List of rules associated with the specified configuration.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoConfig**
  The requested analytics configuration does not exist.

ACCESS CLASS:

**READ_MEDIA**

#### 5.3.3.3 CreateRules

A device signaling support for rules via the RuleSupport capability shall support this operation to add rules to an AnalyticsConfiguration. If all rules can not be created as requested, the device responds with a fault message.

REQUEST:

- **ConfigurationToken [tt:ReferenceToken]**
  Analytics configuration for which the rules should be created.

- **Rule - unbounded [tt:Config]**
  Rules to be added to the specified configuration.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoConfig**
  The requested analytics configuration does not exist.

- **env:Sender - ter:InvalidArgVal - ter:InvalidRule**
  The suggested rule configuration is not valid on the device.

- **env:Sender - ter:InvalidArgVal - ter:RuleAlreadyExistent**
  A rule with the same name already exists in the configuration.

- **env:Receiver - ter:Action - ter:TooManyRules**
  There is not enough space in the device to add the rules to the configuration.

- **env:Receiver - ter:Action - ter:ConfigurationConflict**
  The device cannot create the rules without creating a conflicting configuration.

ACCESS CLASS:

**ACTUATE**

#### 5.3.3.4 ModifyRules

A device signaling support for modifying rules via the RuleOptionsSupported capability shall support this operation. If all rules can not be modified as requested, the device responds with a fault message. A device may reject a request to change the rule type.

A device shall interpret parameters not present in the payload as unchanged. Note that this does not always result in unchanged values of such parameters, since some parameters may change due to dependency on others.

REQUEST:

- **ConfigurationToken [tt:ReferenceToken]**
  Token of an existing analytics configuration.

- **Rule - unbounded [tt:Config]**
  List of rules to be modified for the specified configuration.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoConfig**
  The requested analytics configuration does not exist.

- **env:Sender - ter:InvalidArgVal - ter:InvalidRule**
  The suggested rule configuration is not valid on the device.

- **env:Sender - ter:InvalidArgVal - ter:RuleNotExistent**
  The rule name or names do not exist.

- **env:Receiver - ter:Action - ter:ConfigurationConflict**
  The device cannot modify the rules without creating a conflicting configuration.

- **env:Receiver - ter:Action - ter:TypeChangeProhibited**
  The device cannot modify the rule configuration type without creating a conflicting configuration.

ACCESS CLASS:

**ACTUATE**

#### 5.3.3.5 DeleteRules

A device signaling support for rules via the RuleSupport capability shall support this operation to delete one or more rules. If all rules can not be deleted as requested, the device responds with a fault message.

REQUEST:

- **ConfigurationToken [tt:ReferenceToken]**
  Token of an existing analytics configuration.

- **RuleName - unbounded [xs:string]**
  List of rules to be removed for the specified configuration.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoConfig**
  The requested analytics configuration does not exist.
- **env:Sender - ter:InvalidArgVal - ter:RuleNotExistent**
  The rule name or names do not exist.
- **env:Sender - ter:Action - ter:FixedRules**
  At least one of the requested rules is fixed and cannot be deleted.
- **env:Receiver - ter:Action - ter:ConfigurationConflict**
  The device cannot delete the rules without creating a conflicting configuration.

ACCESS CLASS:

**ACTUATE**

#### 5.3.3.6 GetRuleOptions

A device signaling support for modifying rules via the RuleOptionsSupported capability shall support this operation to retrieve the options for the supported rules that specify an Option attribute.

REQUEST:

- **RuleType - optional [xs:QName]**
  Optional rule type for which the options shall be retrieved. If omitted all rule option types with all rule options supported by the device shall be retuned (as appropriate).
- **ConfigurationToken [tt:ReferenceToken]**
  Token of an existing analytics configuration.

RESPONSE:

- **RuleOptions - optional, unbounded [tan:ConfigOptions]**
  List of options for the specified rule. A device shall provide respective ConfigOptions.RuleType for each RuleOption if the request does not specify RuleType. The response Options shall not contain any AnalyticsModule attribute.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoConfig**
  The requested analytics configuration does not exist.
- **env:Sender - ter:InvalidArgVal - ter:RuleNotExistent**
  The rule option type does not exist.

ACCESS CLASS:

**READ_MEDIA**

## 5.4 Analytics modules interface

Section 5.4.1 defines an XML structure that communicates the configuration of analytics modules. Section 5.4.2 defines the language that describes the configuration of a specific analytics module. Section 5.4.3 defines the operations required by the analytics modules interface. If the device supports an analytics engine as defined by ONVIF, it shall implement the complete analytics modules Interface.

### 5.4.1 Analytics module configuration

The analytics module configuration uses the configuration description language as described in section 5.2.1. The following example shows a possible configuration of a vendor-specific ObjectTracker. This tracker allows configuration of the minimum and maximum object size with respect to the processed frame geometry.

```
<tt:AnalyticsEngineConfig>
  <tt:AnalyticsModule Name="MyObjectTracker" Type="nn:ObjectTracker">
    <tt:Parameters>
      <tt:SimpleItem Name="MinObjectWidth" Value="0.01"/>
      <tt:SimpleItem Name="MinObjectHeight" Value="0.01"/>
      <tt:SimpleItem Name="MaxObjectWidth" Value="0.5"/>
      <tt:SimpleItem Name="MaxObjectHeight" Value="0.5"/>
    </tt:Parameters>
  </tt:AnalyticsModule>
</tt:AnalyticsEngineConfig>
```

### 5.4.2 Analytics module description language

The Analytics Module reuses the configuration description language, described in section 5.2.2. The following AnalyticsModuleDescription element replaces the RuleDescription element:

```
<xs:element name="AnalyticsModuleDescription"
            type="tt:ConfigDescription"/>
```

Similar to rules, analytics modules produce events and shall be listed within the analytics module description. The subsequent description corresponds to the example of the previous section. The example module produces a SceneTooCrowded event when the scene becomes too complex for the module.

```
<tt:AnalyticsModuleDescription Name="nn:ObjectTracker">
  <tt:Parameters>
    <tt:SimpleItemDescription Name="MinObjectWidth"  Type="xs:float"/>
    <tt:SimpleItemDescription Name="MinObjectHeight" Type="xs:float"/>
    <tt:SimpleItemDescription Name="MaxObjectWidth"  Type="xs:float"/>
    <tt:SimpleItemDescription Name="MaxObjectHeight" Type="xs:float"/>
  </tt:Parameters>
  <tt:Messages>
    <tt:Source>
      <tt:SimpleItemDescription Name="VideoSourceConfiguration"
                                Type="tt:ReferenceToken"/>
      <tt:SimpleItemDescription Name="AnalyticsConfiguration"
                                Type="tt:ReferenceToken"/>
      <tt:SimpleItemDescription Name="AnalyticsModule" Type="xs:string"/>
    </tt:Source>
    <tt:ParentTopic>
      tns1:VideoAnalytics/nn:ObjectTracker/SceneTooCrowded
    </tt:ParentTopic>
  </tt:Messages>
</tt:RuleDescription>
```

### 5.4.3 Operations on analytics modules

The Create/Delete/Modify operations shall be atomic, all modifications can be processed or the complete operation shall fail.

#### 5.4.3.1 GetSupportedAnalyticsModules

A device signaling support for analytics modules via the AnalyticsModuleSupport capability shall support retrieving a list of analytics modules for an analytics configuration. The description shall conform to the configuration description language as described in section 5.2.

The optional AnalyticsModuleContentSchemaLocation parameter allows to list schema file locations that provide reference to types or elements which are not defined by ONVIF.

The device shall indicate its limit for maximum number of analytics modules through the maxInstances attribute.

REQUEST:

- **ConfigurationToken [tt:ReferenceToken]**
  Token of an existing analytics configuration.

RESPONSE:

- **SupportedAnalyticsModules[tt:SupportedModules]**
  Analytics modules supported for the analytics configuration

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoConfig**
  The requested analytics configuration does not exist.

ACCESS CLASS:

**READ_MEDIA**

#### 5.4.3.2 GetAnalyticsModules

A device signaling support for analytics modules via the AnalyticsModuleSupport capability shall support this method to retrieve currently associated analytics modules for an analytics configuration.

REQUEST:

- **ConfigurationToken [tt:ReferenceToken]**
  Token of an existing analytics configuration.

RESPONSE:

- **AnalyticsModule - optional, unbounded [tt:Config]**
  List of modules associated with the specified configuration.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoConfig**
  The requested analytics configuration does not exist.

ACCESS CLASS:

**READ_MEDIA**

#### 5.4.3.3 CreateAnalyticsModules

A device signaling support for analytics modules via the AnalyticsModuleSupport capability shall support this method to add analytics modules to an analytics configuration.

The device shall accept adding of analytics modules with an empty Parameter definition. Note that the resulted configuration may include a set of default parameter values.

REQUEST:

- **ConfigurationToken [tt:ReferenceToken]**
  Token of an existing analytics configuration.

- **AnalyticsModule - unbounded [tt:Config]**
  Modules to be added to the specified configuration.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoConfig**
  The requested analytics configuration does not exist.

- **env:Sender - ter:InvalidArgVal - ter:InvalidModule**
  The suggested module configuration is not valid on the device.

- **env:Sender - ter:InvalidArgVal - ter:NameAlreadyExistent**
  The same analytics module name exists already in the configuration.

- **env:Sender - ter:Action - ter:FixedModules**
  At least one of the requested modules is fixed which cannot be created.

- **env:Receiver - ter:Action - ter:TooManyModules**
  There are not enough resources in the device to add the analytics modules to the configuration.

- **env:Receiver - ter:Action - ter:ConfigurationConflict**
  The device cannot create the analytics modules without creating a conflicting configuration.

ACCESS CLASS:

**ACTUATE**

#### 5.4.3.4 ModifyAnalyticsModules

A device signaling support for analytics modules via the AnalyticsModuleOptionsSupported capability shall support this method to modify analytics module configurations. A device may reject a request to change the module type.

A device shall interpret parameters not present in the payload as unchanged. Note that this does not always result in unchanged values of such parameters, since some parameters may change due to dependency on others.

REQUEST:

- **ConfigurationToken [tt:ReferenceToken]**
  Token of an existing analytics configuration.

- **AnalyticsModule - unbounded [tt:Config]**
  List of modules to be modified for the specified configuration.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoConfig**
  The requested analytics configuration does not exist.

- **env:Sender - ter:InvalidArgVal - ter:InvalidModule**
  The suggested module configuration is not valid on the device.

- **env:Sender - ter:InvalidArgVal - ter:NameNotExistent**
  The module name or names do not exist.

- **env:Receiver - ter:Action - ter:ConfigurationConflict**
  The device cannot modify the modules without creating a conflicting configuration.

- **env:Receiver - ter:Action - ter:TypeChangeProhibited**
  The device cannot modify the analytics module type without creating a conflicting configuration.

ACCESS CLASS:

**ACTUATE**

#### 5.4.3.5 DeleteAnalyticsModules

A device signaling support for analytics modules via the AnalyticsModuleSupport capability shall support this method for removing analytics module from an analytics configuration.

REQUEST:

- **ConfigurationToken [tt:ReferenceToken]**
  Token of an existing analytics configuration.

- **AnalyticsModuleName - unbounded [xs:string]**
  List of modules to be removed for the specified configuration.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoConfig**
  The requested analytics configuration does not exist.

- **env:Sender - ter:InvalidArgVal - ter:InvalidModule**
  The suggested module configuration is not valid on the device

- **env:Sender - ter:InvalidArgVal - ter:NameNotExistent**
  The module name or names do not exist.

- **env:Sender - ter:Action - ter:FixedModules**
  At least one of the requested modules is fixed which cannot be deleted.

- **env:Receiver - ter:Action - ter:ConfigurationConflict**
  The device cannot delete the modules without creating a conflicting configuration.

#### 5.4.3.6 GetAnalyticsModuleOptions

The following operation returns the options for the supported analytics modules that specify an Option attribute. A device signaling support for the AnalyticsModuleOptionsSupported capability shall support this method.

REQUEST:

- **Type - optional [xs:QName]**
  Optional module type for which the options shall be retrieved. If omitted all module option types with all module options supported by the device shall be retuned (as appropriate).

- **ConfigurationToken [tt:ReferenceToken]**
  Token of an existing analytics configuration.

RESPONSE:

- **Options - optional, unbounded [tan:ConfigOptions]**
  List of options for the specified analytics module. The response Options shall not contain any RuleType attribute.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoConfig**
  The requested analytics configuration does not exist.

- **env:Sender - ter:InvalidArgVal - ter:TypeNotExistent**
  The module option type does not exist.

ACCESS CLASS:

**READ_MEDIA**

#### 5.4.3.7 GetSupportedMetadata

This operation allows to query what metadata a device can generate. It shall be supported if the SupportedMetadata capability is set to true.

The response contains the following information

- **SampleFrame [tt:Frame]**
  An example Frame instance. The instance shall include all elements that the analytics module outputs to the frame of a metadtata stream. A device implementation should strive for including all supported enumeration values.

If the sample frame includes object bounding boxes or shapes, these shall be located in the top left quarter of the image.

REQUEST:

- **Type - optional [xs:QName]**
  Optional module type for which the metatdata information shall be retrieved. If omitted, the response of this method shall signal the information for all supported analytics modules.

RESPONSE:

- **AnalyticsModule – optional, unbounded [tan:MetadataInfo]**
  This parameter contains sample meta frames.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:TypeNotExistent**
  The module option type does not exist.

ACCESS CLASS:

**READ_MEDIA**

## 5.5 Capabilities

The capabilities reflect optional functions and functionality of a service. The information is static and does not change during device operation. The following capabilites are available:

**RuleSupport:** Indication that the device supports rules interface and rules syntax as specified in Section 5.3.

**AnalyticsModuleSupport:**
Indication that the device supports the scene analytics module interface as specified in Section 5.4.

**CellBasedSceneDescriptionSupported:**
Indication that the device produces the cell based scene description.

**RuleOptionsSupported:**
Indication that the device supports GetRuleOptions and ModifyRules.

**AnalyticsModuleOptionsSupported:**
Indication that the device supports GetAnalyticsModuleOptions and ModifyAnalyticsModules.

**SupportedMetadata**: Indication that the device supports the method GetSupportedMetadata.

**ImageSendingType:** Lists the supported methods for transmitting images.

REQUEST:

This message is empty

RESPONSE:

- **Capabilities [tan:Capabilities]**
  Set of indicators for function groups as described above.

FAULTS:

None

ACCESS CLASS:

**PRE_AUTH**

## 5.6 Events

### 5.6.1 Audio Detected

Analytics engine can send the following event for analyzed audio data.

```
<tt:MessageDescription IsProperty=“false">
  <tt:Source>
   <tt:SimpleItemDescription Name=“AudioSourceConfigurationToken"
     Type="tt:ReferenceToken"/>
   <tt:SimpleItemDescription Name=“AudioAnalyticsConfigurationToken"
     Type="tt:ReferenceToken"/>
   <tt:SimpleItemDescription Name=“Rule"
     Type="xs:string"/>
 </tt:Source>
  <tt:Data>
   <tt:SimpleItemDescription Name=“isSoundDetected" Type="xs:boolean"/>
   <tt:SimpleItemDescription Name=“UTCTime" Type="xs:dateTime"/>
   <tt:ElementItemDescription                      Name=“AudioClassifications"
Type=“AudioClassDescriptor"/>
</tt:Data >
  <tt:ParentTopic>tns1:AudioAnalytics/Audio/DetectedSound
  </tt:ParentTopic>
 </tt:MessageDescription>
```

The Rule field is optional and contains the Rule instance name that generated the audio detected event.

# Annex A. Specified Rules
(informative)

The following rules apply to static cameras. In case of a PTZ device, image-based rules should contain an additional ElementItem. The ElementItem identifies the position of the device for which the rule has been setup. The corresponding ElementItemDescription resembles the following:

```
<tt:ElementItemDescription Name="PTZStatus" Type="tt:PTZStatus">
```

## A.1 LineDetector

LineDetector is defined by a non-intersecting simple polyline. If an object crosses the polyline in the specified direction, the rule engine sends a Crossed event containing the name of the LineDetector and a reference to the object which has crossed the line. As directions, one can select between Left, Right, and Any, where directions Left and Right refer to the direction walking along the line from the first point to the second point and are the prohibited directions. The specific definition of the Left and Right direction can refer to the following Figure 4, A is the first point and B is the second point.

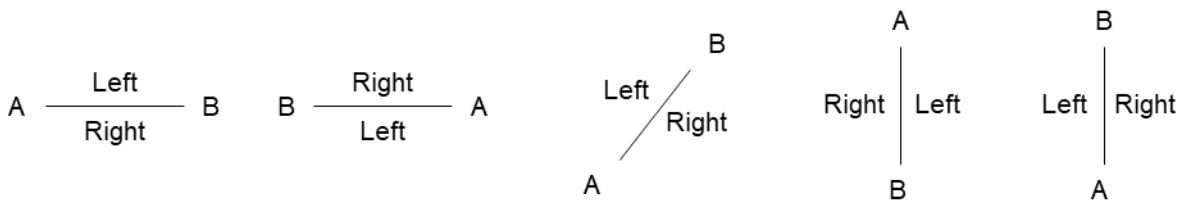



**Figure 4: Direction – Left and Right**

The LineDetector resembles the following code using the rule description language, detailed in the previous section:

```
<tt:RuleDescription Name="tt:LineDetector">
  <tt:Parameters>
    <tt:SimpleItemDescription Name="Direction" Type="tt:Direction"/>
    <tt:ElementItemDescription Name="Segments" Type="tt:Polyline"/>
  </tt:Parameters>
  <tt:Messages>
    <tt:Source>
      <tt:SimpleItemDescription Name="VideoSourceConfigurationToken"
                                Type="tt:ReferenceToken"/>
      <tt:SimpleItemDescription Name="VideoAnalyticsConfigurationToken"
                                Type="tt:ReferenceToken"/>
      <tt:SimpleItemDescription Name="Rule" Type="xs:string"/>
    </tt:Source>
    <tt:Data>
      <tt:SimpleItemDescription Name="ObjectId" Type="xs:integer"/>
    </tt:Data>
    <tt:ParentTopic>tns1:RuleEngine/LineDetector/Crossed</tt:ParentTopic>
  </tt:Messages>
</tt:RuleDescription>
```

The code above defines two parameters, Segments and Direction, and produces one event attached to the topic tns1:RuleEngine/LineDetector/Crossed.

## A.2 Field Detector

A FieldDetector is defined by a simple non-intersecting polygon. The FieldDetector determines if each object in the scene inside or outside the polygon. This information is put into a property.

FieldDetector resembles the following code, using the Rule Description Language detailed in the previous section:

```
<tt:RuleDescription Name="tt:FieldDetector">
  <tt:Parameters>
    <tt:ElementItemDescription Name="Field" Type="tt:Polygon"/>
  </tt:Parameters>
  <tt:Messages IsProperty="true">
    <tt:Source>
      <tt:SimpleItemDescription Name="VideoSourceConfigurationToken"
                                Type="tt:ReferenceToken"/>
      <tt:SimpleItemDescription Name="VideoAnalyticsConfigurationToken"
                                Type="tt:ReferenceToken"/>
      <tt:SimpleItemDescription Name="Rule" Type="xs:string"/>
    </tt:Source>
    <tt:Key>
      <tt:SimpleItemDescription Name="ObjectId" Type="xs:integer"/>
    </tt:Key>
    <tt:Data>
      <tt:SimpleItemDescription Name="IsInside" Type="xs:boolean"/>
    </tt:Data >
    <tt:ParentTopic>
      tns1:RuleEngine/FieldDetector/ObjectsInside
    </tt:ParentTopic>
  </tt:Messages>
</tt:RuleDescription>
```

From the Inside property, a client can derive the Entering and the Leaving parameters of the detector. A client can simulate Entering and Leaving events by adding a MessageContent Filter to the subscription, which lets only ObjectsInside messages pass, where the IsInside Item is set to true resp. false.

### A.3 Loitering Detector

A LoiteringDetector is defined by a simple non-intersecting polygon as an area of interest and threshold loitering interval. The LoiteringDetector determines if each object in the scene inside the polygon longer than the given time threshold value. It publishes when the object started loitering.

LoiteringDetector defined by the following code using the rule description language:

```
<tt:RuleDescription Name="tt:LoiteringDetector">

 <tt:Parameters>
    <tt:ElementItemDescription Name="Field"
                               Type="tt:PolygonConfiguration"/>
    <tt:SimpleItemDescription  Name="TimeThreshold"
                               Type="xs:nonNegativeInteger"/>
    <tt:SimpleItemDescription  Name="isPeriodicNotify" Type="xs:boolean"/>
    <tt:SimpleItemDescription  Name="TimeInterval"     Type="xs:duration"/>
  </tt:Parameters>

  <tt:Messages IsProperty="true">
    <tt:Source>
      <tt:SimpleItemDescription Name="VideoSourceConfigurationToken"
                                Type="tt:ReferenceToken"/>
      <tt:SimpleItemDescription Name="VideoAnalyticsConfigurationToken"
                                Type="tt:ReferenceToken"/>
      <tt:SimpleItemDescription Name="Rule" Type="xs:string"/>
    </tt:Source>
    <tt:Key>
      <tt:SimpleItemDescription Name="ObjectId" Type="xs:integer"/>
    </tt:Key>
    <tt:Data>

      <tt:SimpleItemDescription Name="Since"     Type="xs:dateTime"/>
      <tt:SimpleItemDescription Name="TimeStamp" Type="xs:dateTime"/>
    </tt:Data >
    <tt:ParentTopic>
      tns1:RuleEngine/LoiteringDetector/ObjectIsLoitering
    </tt:ParentTopic>
  </tt:Messages>
</tt:RuleDescription>
```

The above rule definition defines that a rule instance produces an event attached to the topic tns1:RuleEngine/LoiteringDetector/ObjectIsLoitering, and the configuration parameters are given in Table A-1.

**Table A-1 Loitering Detector rule configuration paramaters**

| Parameter Name | Description |
|---|---|
| Field | Area in camera field of view to detect object loitering |
| TimeThreshold | Maximum time threshold to judge that object is loitering |
| isPeriodicNotify | Whether continue to generate periodic events for loitering object |
| TimeInterval | Periodic time interval to continue to send loitering events for loitering object |

The event contains the fields given in Table A-2.

**Table A-2 Description of loitering event fields**

| Parameter Name | Description |
|---|---|
| ObjectId | Object Identifier |
| Since | Since when this object is loitering |
| TimeStamp | Timestamp of last object sample triggering generation of rule’s event |

### A.4 Declarative motion detector

Declarative motion detector detects the object motion against the specified motion attributes. The rule is active within an area (region of interest) and time interval. The condition is defined against the frame/object element in scene descriptor.

```
<tt:RuleDescription Name="tt:DeclarativeMotionDetector">

 <tt:Parameters>
  <tt:ElementItemDescription Name="Field"     Type="tt:PolygonConfiguration"/>
  <tt:ElementItemDescription Name="MotionExpression"
                             Type="tt:MotionExpressionConfiguration"/>
 </tt:Parameters>

 <tt:Messages IsProperty="true">
  <tt:Source>
   <tt:SimpleItemDescription Name="VideoSourceConfigurationToken"
                             Type="tt:ReferenceToken"/>
   <tt:SimpleItemDescription Name="VideoAnalyticsConfigurationToken"
                             Type="tt:ReferenceToken"/>
   <tt:SimpleItemDescription Name="Rule" Type="xs:string"/>
  </tt:Source>
  <tt:Key>
   <tt:SimpleItemDescription Name="ObjectId" Type="xs:integer"/>
  </tt:Key>
  <tt:Data>
    <tt:SimpleItemDescription Name="TimeStamp" Type="xs:dateTime"/>
  </tt:Data >
  <tt:ParentTopic>
   tns1:RuleEngine/DeclarativeMotionDetector/MotionMatched
  </tt:ParentTopic>
 </tt:Messages>
</tt:RuleDescription>
```

The above rule description defines that a rule instance produces event attached to the topic tns1:RuleEngine/DeclarativeMotionDetector/MotionMatched, and the configuration parameters given in Table A-3.

**Table A-3 Declarative Motion Detector rule configuration parameters**

| Parameter Name | Description |
|---|---|
| Field | Defines an area in camera field of view, the motion matching query is applied in this area for detected object. |
| MotionExpression | Motion Expression for frame/object elements in Scene Descriptor schema |

The event contains the fields given in Table A-4.

**Table A-4 Description of declarative motion event fields**

| Parameter Name | Description |
|---|---|
| ObjectId | Object Identifier |
| TimeStamp | Timestamp of detected motion |

## A.5 Counting rule

Counting rule counts the number of motion object passing through the set of line segments (barrier) and optional direction attribute. The configuration parameters also include the time interval to report the events and time interval to reset its counter.

```
<tt:RuleDescription Name="tt:CountAggregation">

 <tt:Parameters>
  <tt:ElementItemDescription Name="LineSegments"
                             Type="tt:PolylineArrayConfiguration"/>
  <tt:SimpleItemDescription  Name="ReportTimeInterval" Type="xs:duration"/>
  <tt:SimpleItemDescription  Name="ResetTimeInterval"  Type="xs:duration"/>
  <tt:SimpleItemDescription  Name="Direction"  Type="tt:Direction"/>
 </tt:Parameters>

 <tt:Messages IsProperty="true">
  <tt:Source>
   <tt:SimpleItemDescription Name="VideoSourceConfigurationToken"
                             Type="tt:ReferenceToken"/>
   <tt:SimpleItemDescription Name="VideoAnalyticsConfigurationToken"
                             Type="tt:ReferenceToken"/>
   <tt:SimpleItemDescription Name="Rule" Type="xs:string"/>
  </tt:Source>
  <tt:Key>
   <tt:SimpleItemDescription Name="ObjectId" Type="xs:integer"/>
  </tt:Key>
  <tt:Data>
   <tt:SimpleItemDescription Name="Count"    Type="xs:nonNegativeInteger"/>
  </tt:Data >
  <tt:ParentTopic>
    tns1:RuleEngine/CountAggregation/Counter
  </tt:ParentTopic>
 </tt:Messages>

</tt:RuleDescription>
```

The above rule description defines that a rule instance produces a count event of topic tns1:RuleEngine/CountAggregation/Counter. Configuration parameters are given in Table A-5.

**Table A-5 Counting rule configuration parameters**

| Parameter Name | Description |
|---|---|
| LineSegments | Array of lines used for detecting that object passed all of them for counting |
| ReportTimeInterval | Time interval to report count information |
| ResetTimeInterval | Periodic count reset time |
| Direction | Count direction |

The event contains the fields given in Table A-6.

**Table A-6 Description of counting event fields**

| Parameter Name | Description |
|---|---|
| ObjectId | Object Identifier of last counted object |
| Count | Value of counter |

## A.6 Object Detection

The object detection rule generates events when any of the configured object types are detected in the field of view. Whenever there is change in the detected object types, a new event is notified, if none of the objects from the configured list is detected, an event with empty object list is notified.

```
<tt:RuleDescription Name="tt:ObjectDetection">
  <tt:Parameters>
    <tt:SimpleItemDescription Name="ClassFilter" Type=" tt:StringList"/>
    <tt:SimpleItemDescription Name="ConfidenceLevel" Type="xs:float"/>
    <tt:SimpleItemDescription Name="DwellTime" Type="xs:duration"/>
  </tt:Parameters>
  <tt:Messages>
    <tt:Source>
      <tt:SimpleItemDescription Type="tt:ReferenceToken" Name="VideoSource"/>
      <tt:SimpleItemDescription Type="xs:string" Name=" RuleName" />
    </tt:Source>
    <tt:Data>
      <tt:SimpleItemDescription Type="tt:StringList" Name="ClassTypes"/>
    </tt:Data>
    <tt:ParentTopic>tns1:RuleEngine/ObjectDetection/Object</tt:ParentTopic>
  </tt:Messages>
</tt:RuleDescription>
```

The object types may differ between devices. See below for example rule options, based on the description language parameter options defined in 5.2.3.2.

```
<tan:GetRuleOptionsResponse>
  <tan:RuleOptions Name="ClassFilter" RuleType="tt:ObjectDetection">
    <tt:StringList>Person Vehicle Face LicensePlate</tt:StringList>
  </tan:RuleOptions>
  <tan:RuleOptions Name="ConfidenceLevel" RuleType="tt:ObjectDetection">
       <tt:FloatRange>
         <tt:Min>0.0</tt:Min>
         <tt:Max>100.0</tt:Max>
       </tt:FloatRange>
  </tan:RuleOptions>
  <tan:RuleOptions Name="DwellTime " RuleType="tt:ObjectDetection">
       <tt:DurationRange >
         <tt:Min>PT1S</tt:Min>
         <tt:Max>PT1M</tt:Max>
       </tt:DurationRange>
  </tan:RuleOptions>
</tan:GetRuleOptionsResponse>
```

## A.7 Object Classification

Object classification can be utizlied in rules to restrict generation of events to objects of one or more types. Items should be selected from the ObjectType enumeration. Examples for object types are

- Animal
- HumanFace
- Human
- Bicycle
- Vehicle
- LicensePlate

Note, that the actually supported object types can be notified using the description language parameter options defined in 5.2.3.2.

The following example shows a RuleDescription including ClassFilter in the parameter description.

```
<tt:RuleDescription Name="tt:XXXDetector">
  <tt:Parameters>
    <tt:SimpleItemDescription Name="Direction" Type="tt:Direction"/>
    <tt:ElementItemDescription Name="Segments" Type="tt:Polyline"/>
    <tt:SimpleItemDescription Name="ClassFilter" Type="tt:StringList"/>
  </tt:Parameters>
 ...
  </tt:Messages>
</tt:RuleDescription>
```

## A.8 Query rule

The query rule allows a client to define a query on streamed metadata of multimedia streams. The client subscribes to the topic associated with the rule to receive notifications from a device.

```
<tt:RuleDescription Name="tt:QueryRule">
 <tt:Parameters>
  <tt:ElementItemDescription Name="Query" Type="tt:MetadataFilter"/>
  <tt:SimpleItemDescription Name="VideoSource" Type="tt:ReferenceTokenList"/>
 </tt:Parameters>

 <tt:Messages>
  <tt:Source>
   <tt:SimpleItemDescription Name="VideoSourceConfigurationToken"
Type="tt:ReferenceToken"/>
   <tt:SimpleItemDescription Name="VideoAnalyticsConfigurationToken"
Type="tt:ReferenceToken"/>
   <tt:SimpleItemDescription Name="Rule" Type="xs:string"/>
  </tt:Source>
  <tt:Data>
   <tt:ElementItemDescription Name="Result" Type="xs:any"/>
  </tt:Data>
  <tt:ParentTopic>
   tns1:RuleEngine/QueryRule
  </tt:ParentTopic>
 </tt:Messages>

</tt:RuleDescription>
```

The topic is tns1:RuleEngine/QueryRule and the value of "Rule" in the notification event wll indicate which Query Rule instance generated the event. The client should subscribe the topic with the proper rule name to receive the events from the specific query rule. The configuration parameters are given in Table A-7.

**Table A-7 Query Rule configuration parameters**

| Parameter Name | Description |
|---|---|
| Query | Contains the query definition |
| VideoSource | Selects which video source or sources the query will execute on |

# Annex B. Cell motion detection
(informative)

## B.1 Cell motion detector

Cell motion detector rules process the output of cell motion analytics engine. The rule of type "tt:CellMotionDetector" consists of four parameters: "MinCount", "AlarmOnDelay", "AlarmOffDelay" and "ActiveCells" as listed in Table B-1.

The parameter "MinCount" describes the minimum number of adjacent cells that a moving object shall cover in order to generate an alarm event.

AlarmOnDelay and AlarmOffDelay are delay times in milliseconds. These delays should prevent alarm state changes within a short time. An alarm has to be active for at least AlarmOnDelay time to trigger an alarm. AlarmOffDelay will delay the transition from an activated alarm to an inactive alarm. An alarm has to be inactive for at least AlarmOffDelay time to become inactive again.

Each of the cells defined by the type "tt:CellLayout" in the AnalyticsModule, can be activated or deactivated individually. To exclude particular regions from being monitored owing to continuous movements (e.g. tree in the wind), the relevant cells can be deactivated. For a representation of the activated and deactivated cells a bitmask is used. For instance, if there are 1000 cells available, then 1000 bits need to be set. To efficiently encode this bitmask, the algorithm "Packbits" from ISO 12369 (TIFF, Version 6.0), which is a kind of run length encoder, is used. Afterwards the resulting output is stored as base64Binary.

The rule definition is as follows:

```
<tt:RuleDescription Name="tt:CellMotionDetector">
  <tt:Parameters>
    <tt:SimpleItemDescription Name="MinCount" Type="xs:integer"/>
    <tt:SimpleItemDescription Name="AlarmOnDelay" Type="xs:integer"/>
    <tt:SimpleItemDescription Name="AlarmOffDelay" Type="xs:integer"/>
    <tt:SimpleItemDescription Name="ActiveCells" Type="xs:base64Binary"/>
  </tt:Parameters>
  <tt:Messages IsProperty="true">
    <tt:Source>
      <tt:SimpleItemDescription
        Name="VideoSourceConfigurationToken"
        Type="tt:ReferenceToken"/>
      <tt:SimpleItemDescription
        Name="VideoAnalyticsConfigurationToken"
        Type="tt:ReferenceToken"/>
      <tt:SimpleItemDescription Name="Rule" Type="xs:string"/>
    </tt:Source>
    <tt:Data>
      <tt:SimpleItemDescription Name="IsMotion" Type="xs:boolean"/>
    </tt:Data>
    <tt:ParentTopic>
      tns1:RuleEngine/CellMotionDetector/Motion
    </tt:ParentTopic>
  </tt:Messages>
</tt:RuleDescription>
```

**Table B-1 Cell motion detector rule configuration parameters**

| Parameter Name | Description |
|---|---|
| MinCount | Minimum count of adjacent activated cells to trigger motion. This parameter allows suppressing false alarms. |

| | |
|---|---|
| AlarmOnDelay | Minimum time in milliseconds which is needed to change the alarm state from off to on. This delay is intended to prevent very brief alarm events from triggering. |
| AlarmOffDelay | Minimum time in milliseconds which is needed to change the alarm state from on to off. This delay is intended to prevent too many alarm changes. |
| ActiveCells | A "1" denotes an active cell and a "0" an inactive cell.<br><br>The first cell is in the upper left corner. Then the cell order goes first from left to right and then from up to down (see Figure B-1).<br><br>If the number of cells is not a multiple of 8 the last byte is filled with zeros. The information is run length encoded according to Packbit coding in ISO 12369 (TIFF, Revision 6.0). |

The event contains the following fields given in Table B-2.

**Table B-2 Description Cell Motion detected event fields**

| Parameter Name | Description |
|---|---|
| IsMotion | True if motion is detected |

The following code snippet is for cell motion detector rule configuration.

```
<tt:VideoAnalyticsConfiguration>
  <tt:RuleEngineConfiguration>
    <tt:Rule Name="MyMotionDetector" Type="tt:CellMotionDetector">
      <tt:Parameters>
        <tt:SimpleItem Name="MinCount" Value="4"/>
        <tt:SimpleItem Name="AlarmOnDelay" Value="1000"/>
        <tt:SimpleItem Name="AlarmOffDelay" Value="1000"/>
        <tt:SimpleItem Name="ActiveCells" Value="/v/+8A=="/>
      </tt:Parameters>
    </tt:Rule>
  </tt:RuleEngineConfiguration>
</tt:VideoAnalyticsConfiguration>
```

The activated cells are filled with gray and the deactivated cells are white. The active cells in hex coding are: "*ff ff ff f0 f0 f0*". This is encoded with the Packbit algorithm: "*fe ff fe f0*". Finally, this packed data is base64Binary encoded: *"/v/+8A=="*.

## B.2 Cell motion analytics engine

A cell motion detector determines the motion in an image. For this, the image is subdivided into square or rectangle sensor fields, denoted as cells. A cell is based on a number of pixels, for example, an 5×5 cell is a cell of 25 pixels. The detector tries simply to estimate motion in each cell.

A cell motion detector consists of two: units AnalyticsModule and Rule. AnalyticsModule consists of two parameters: "CellLayout" and "Sensitivity" as listed in Table B-3. The latter can adjust the motion detector to the environment to which the camera is subjected. For lower sensitivity values higher variations of the luminance values are needed to be identified as motion. Hence, a low sensitivity can be used to suppress estimated motion due to noise.

The subdivision of the sensor fields depends on the analytics engine and cannot be modified by the user. The type "tt:CellLayout", which describes the cell structure, consists of the following three elements: "Columns", "Rows", and "Transformation". The first two parameters describe the number of cells in the X and Y directions. The first cell is located in the upper left -hand corner and the last cell in the lower right-hand corner. The parameter "Transformation" maps the frame coordinate system (see 5.1.2.2) to the cell grid. The origin of the new coordinate system is the upper left-hand corner of the upper left-hand cell. The unit dimension in X direction corresponds to the cell width and the unit dimension in Y direction corresponds to the cell height.

### B.2.1 Module configuration

Cell motion analytics engine module configuration is described by analytics module description language;

```
<tt:AnalyticsModuleDescription Name="tt:CellMotionEngine">
  <tt:Parameters>
    <tt:SimpleItemDescription Name="Sensitivity"
                              Type="xs:integer"/>
    <tt:ElementItemDescription Name="Layout" Type="tt:CellLayout"/>
  </tt:Parameters>
</tt:AnalyticsModuleDescription>
```

**Table B-3 Module configuration parameters**

| Parameter name | Description |
|---|---|
| Sensitivity | The lower the value of the sensitivity is set, the higher the variations of the luminance values need to be in order to be identified as motion. Hence, a low sensitivity can be used to suppress estimated motion due to noise.<br><br>Range is 0 to 100 |
| Layout | The layout is typically defined by the hardware of the motion detector.<br><br>It is fixed and cannot be altered. The layout structure defines the size of the cell array as well as their mapping to the video image. |

The definition uses tt:CellLayout data structure defined as;

```
<xs:complexType name="CellLayout">
 <xs:sequence>
  <xs:element name="Transformation" type="tt:Transformation"/>
  <xs:any namespace="##any" processContents="lax"
```

```
            minOccurs="0" maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attribute name="Columns"        type="xs:integer" use="required"/>
 <xs:attribute name="Rows"           type="xs:integer" use="required"/>
 <xs:anyAttribute processContents="lax"/>
</xs:complexType>
```

**Table B-4 Description of CellLayout fields**

| Parameter name | Description |
|---|---|
| Columns | Number of columns of the cell grid (x dimension) |
| Rows | Number of rows of the cell grid (y dimension). |
| Transformation | Mapping of the cell grid to the Video frame.<br><br>The cell grid is starting from the upper left corner and x dimension is going from left to right and the y dimension from up to down. |

An example of a configuration of a CellMotionEngine is shown below where an 8x6 cells motion detector is depicted. The video image is assumed to have the size 180×150 pixels. The cells have a dimension of 15×15 pixels and the size of the working area of the cell motion detector is 120×90 pixels

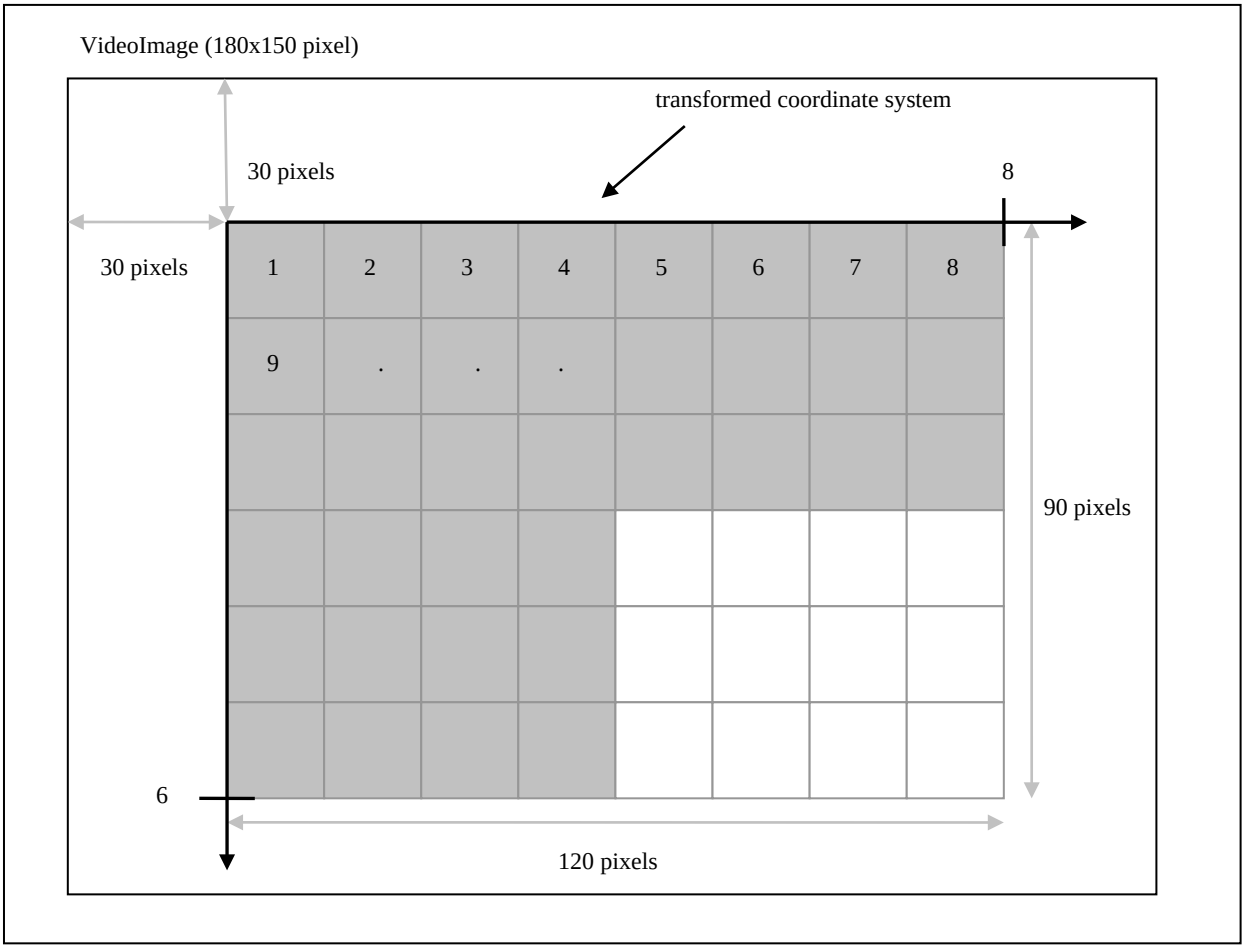



**Figure B-1: CellLayout of an 8×6 CellMotionEngine**

```
<tt:VideoAnalyticsConfiguration>
```

```
  <tt:AnalyticsEngineConfiguration>
    <tt:AnalyticsModule Name="MyCellMotion" Type="tt:CellMotionEngine">
      <tt:Parameters>
        <tt:SimpleItem Name="Sensitivity" Value="90"/>
        <tt:ElementItem Name=”Layout”>
         <tt:CellLayout Columns="8" Rows="6">
          <tt:Transformation>
            <tt:Translate x="-0.66666" y="0.6" />
            <tt:Scale x="0.1666666" y="-0.2" />
          </tt:Transformation>
         </tt:CellLayout>
        </tt:ElementItem>
      </tt:Parameters>
    </tt:AnalyticsModule>
</tt:AnalyticsEngineConfiguration>
```

# Annex C. Motion detection
(normative)

## C.1 Motion region detector

The region motion detector detects any motion against the specified motion region. The rule is configured for an area (region of interest on the image source) which can be armed or disarmed. The region shall be defined by a Polygon structure. Although this rule is defined within the context of the analytics service, it is intended to allow configuration of hardware motion detection as opposed to motion detection from scene description data. For a description of the parameters see Table C-1.

```
<tt:RuleDescription Name="tt:MotionRegionDetector">
  <tt:Parameters>
    <tt:ElementItemDescription Name="MotionRegion" Type="axt:MotionRegionConfig"/>
  </tt:Parameters>
  <tt:Messages IsProperty="true">
    <tt:Source>
       <tt:SimpleItemDescription Name="VideoSource" Type="tt:ReferenceToken"/>
       <tt:SimpleItemDescription Name="RuleName" Type="xs:string"/>
    </tt:Source>
    <tt:Key/>
    <tt:Data>
       <tt:SimpleItemDescription  Name="State" Type="xs:boolean"/>
    </tt:Data>
    <tt:ParentTopic>tns1:RuleEngine/MotionRegionDetector/Motion</tt:ParentTopic>
  </tt:Messages>
</tt:RuleDescription>
```

The above rule description defines that a rule instance produces an event attached to the topic tns1:RuleEngine/MotionRegionDetector/Motion with the state field defined in Table C-3. Calling the GetRuleOptions operation with the tt:MotionRegionDetector type will return a axt:MotionRegionConfigOptions with parameters defined in Table C-2.

**Table C-1 Motion region detector rule configuration parameters**

| Parameter Name | Description |
|---|---|
| Armed | Indicates if the Motion Region is armed (detecting motion) or disarmed (motion is not being detected). |
| Polygon | Provides the points of a polygon that is specified within a region defined by Bounds element of VideoSourceConfiguration. If the device does not support polygons, the points in the polygon must represent a rectangle. |
| Sensitivity | Indicates the sensitivity level of the motion detector for this region. The sensitivity value is normalized where 0 represents the lower sensitivity where significant motion is required to trigger an alarm and 1 represents the higher sensitivity where very little motion is required to trigger an alarm. |
| PresetToken | PTZ Preset position associated with the motion region defined by polygon. If no PresetToken is present the motion region shall behave as a screen element, and stay on the same screen coordinates as the PTZ moves (like a head up display mask). If PresetToken is present the motion region shall be active only when completely contained in the Field of View. |

**Table C-2 Motion region detector rule configuration options**

| Parameter Name | Description |
|---|---|
| MaxRegions | The total number of Motion Region Detector rules that can be created on the device. This element is deprecated. maxInstances in the GetSupportedRules shall be used instead. |
| DisarmSupport | True if the device supports disarming a Motion Region Detector rule. |
| PolygonSupport | True if the device supports defining a region using a Polygon instead of a rectangle. The rectangle points are still passed using a Polygon element if the device does not support polygon regions. In this case, the points provided in the Polygon element must represent a rectangle. |
| PolygonLimits | For devices that support Polygons with limitations on the number of points, provides the minimum and maximum number of points that can be defined in the Polygon. |
| RuleNotification | Indicates the device will supply the Name of the Rule that triggered the motion. |
| SingleSensitivitySupport | Indicates the device can only support one sensitivity level for all defined motion detection regions. Changing the sensitivity for one region would be applied to all regions. |
| PTZPresetMotionSupport | Indicates the support for PTZ preset based motion detection, if supported Preset token can be associated with a motion region. |

**Table C-3 Description of the motion region detector event fields**

| Parameter Name | Description |
|---|---|
| State | True (motion is detected) or False (motion is not detected) |

# Annex D. Radiometry
(normative)

This Annex describes Analytics Modules and Rules used for Radiometry. These are supported by thermal imaging Devices that have been radiometrically calibrated, and thus can provide the absolute temperature values for points on the image. The Radiometry capability is indicated in the Thermal Service, as well as in the Get Service Capabilities Response of the Analytics Service.

Radiometric video sources offer Analytics Configuration Modules for the Client to declare those points or areas on the image for which Temperature measurements shall be provided. These Temperature Measurements are provided by these Modules as streaming Metadata.

In order to calculate the temperature for a point in the image, a number of parameters are needed. These **Radiometry Parameters** include distance to the object being measured, emissivity of its surface, ambient temperature, and optional parameters such as relative humidity, atmospheric transmittance, etc.

To simplify the creation of new Analytics Modules, or the update of parameters that could be common to all the Active measurements, a set of **Global Radiometry Parameters** is defined for the Device, in the Thermal Service Configuration.

This specification covers the two most common measurement modules offered by radiometric devices today: Spots and Boxes.

A **Spot Measurement Module** shall provide the absolute temperature value for a point on the image (this could be a pixel, or small pixel cluster, depending on the lens/resolution of the device).

A **Box Measurement Module** provides a number of different temperature values which are calculated based on the temperature values of the points contained in a rectangular area of the image. These **BoxTemperatureTypes** can be, for example, the Maximum temperature value for the points contained in the rectangular area, or the Minimum value, Average, Median, etc.

**Radiometry Rules** can be defined using the Rules Engine. These Rules trigger alarms based on **TemperatureConditions** of the output of their associated Analytics Module. Temperature Conditions supported by Devices can include, for example, LessThan (threshold), MoreThan, EqualTo, etc.

This specification describes a very simple rule, which is based on a single Measurement Module output, the **Temperature Rule**. These rules shall provide an Event when the TemperatureCondition is met by the selected output (Temperature Value) of the Analytics Module on which they are defined.

**Radiometry Analytics Module Options** are provided by the GetAnalyticsModuleOptions Response, as described in the Analytics Service. Options and Types are described in the **Radiometry Schema**, (radiometry.xsd), under /Analytics.

**Radiometry Rule Options** are provided by the GetRuleOptions Response, as described in the Analytics Service. Options and Types for Radiometry Rules are also described in radiometry.xsd.

The following diagram illustrates the different elements specified for Radiometry, and the relationships and data flows between the different Analytics Modules and Rules, Temperature Measurement Metadata and Temperature Alarm Events.

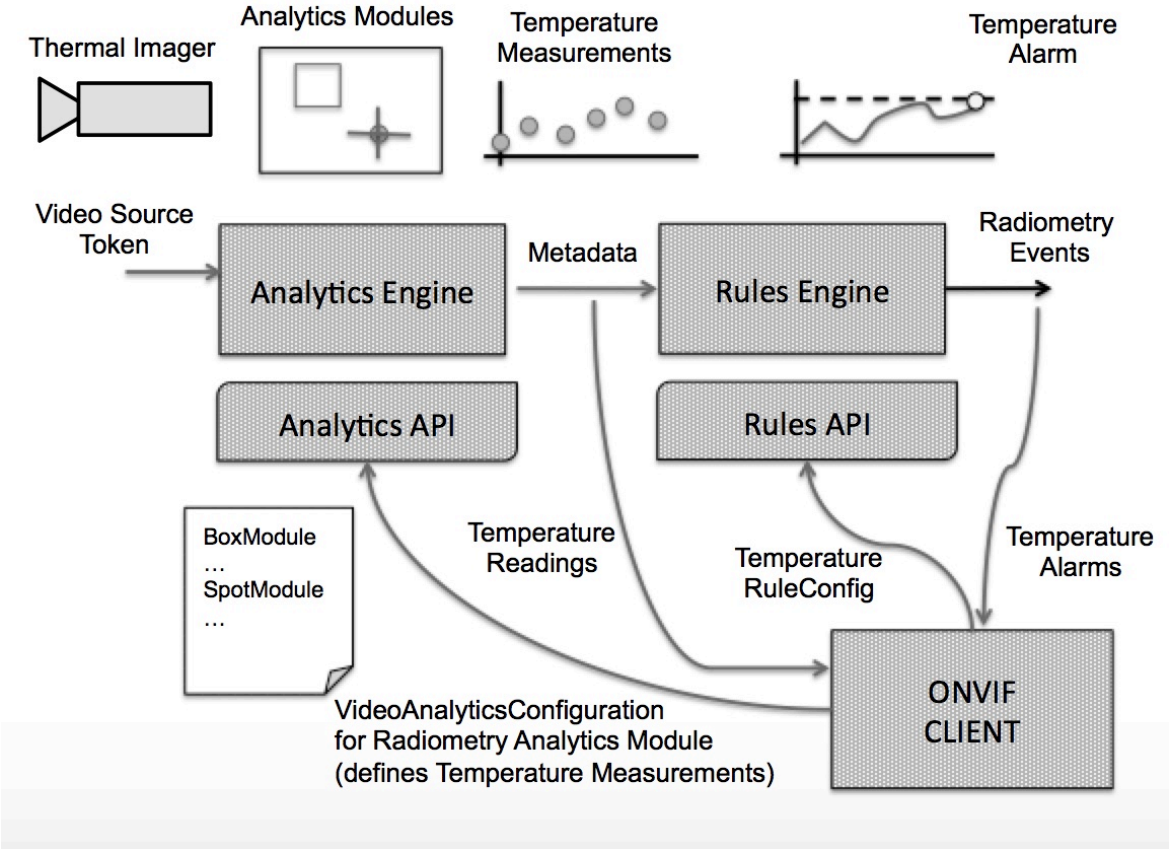



**Figure D-1: Radiometry Analytics Modules, Rules, Metadata and Events**

The current specification covers two different types of Analytics Modules for Radiometry: “Screen” Spots and Boxes, and “Geo-referenced” Spots and Boxes.

Spot and Box Measurement Modules can be defined as “Screen” elements, fixed points/area on the image, or as “Geo-referenced” elements, points/area in a specific location in the PTZ space around the Device.

“Screen” Spots are defined using an (x,y) tt:Vector, which describes a point on the image. Similarly, “Screen” Boxes are defined using a tt:Rectangle, which describes a rectangle on the image. Its coordinates assume a normalized screen [(-1.0,1.0), (-1.0,1.0)], with the center at (0.0,0.0). In PTZ Devices, when the PTZ moves, these screen elements shall behave as OSD masks, and remain on the same relative position of the image as the PTZ moves. Temperature measurements require the image to stabilize, so valid temperature values will normally only be produced and sent as Metadata a number of seconds after the PTZ stops. This acquisition time will vary among different Devices.

“Geo-referenced” Spots can be defined using an (x,y) tt:Vector, together with a tt:PTZVector, and the “Distance to Object” radiometric parameter of the Analytics Module. “Geo-referenced” Boxes share the same definition, with a tt:Rectangle to define their shape and absolute position on the screen.

For Geo-referenced Spot/Box creation, two use cases are contemplated:

(i) Spot/Box is created in the current field of view: the Client shall provide the tt:Vector, or tt:Rectangle for Boxes, and query PTZStatus to provide the current PTZVector as Parameter. Specifying the PTZVector shall indicate the Device that this Spot/Box is “geo-referenced” to that location. The Device can then normalize this PTZVector to store the tt:Vector, or tt:Rectangle, referenced to the center of the image.

(ii) Spot is created in a location outside the current field of view, or Box is not completely contained in the current field of view: the Client can either provide individually a tt:Vector,

or tt:Rectangle, and a PTZVector (as in previous case), or just provide the PTZVector (and Distance) , assuming the tt:Vector, or tt:Rectangle, to be centered in the image.

For Screen Spot or Box Module creation, the Client shall just provide the tt:Vector, or tt:Rectangle, which defines the position of the element on the image, independently of the PTZStatus.

## D.1 Radiometry Analytics Modules

Calling the GetAnalyticsModuleOptions operation with the ttr:RadiometryModule type shall return a ttr:RadiometryModuleConfigOptions, as well as ttr:RadiometrySpotModuleOptions and ttr:RadiometryBoxModuleOptions, if Spot and Box Modules are supported by the Device.

**Table D-1 Radiometry Module configuration options**

| Parameter Name | Description |
|---|---|
| MaxMeasurementModules | The total number of Radiometry Analytics Modules that can be created on the device (Spots, Boxes, Geo or Screen). |
| MaxScreenSpots | The total number of spot measurement modules that can be loaded simultaneously on the screen by the device. A value of 0 shall be used to indicate no support for Spots. |
| MaxScreenBoxes | The total number of box measurement modules that can be loaded simultaneously on the screen by the device. A value of 0 shall be used to indicate no support for Boxes. |
| RadiometryParameterOptions | Specifies valid ranges for the different radiometry parameters used for temperature calculation:<br>• ReflectedAmbientTemperature<br>• Emissivity<br>• DistanceToObject<br>• RelativeHumidity<br>• AtmosphericTemperature<br>• AtmosphericTransmittance<br>• ExtOpticsTemperature<br>• ExtOpticsTransmittance |

### D.1.1 Spot Measurement Module

A Spot Measurement Analytics Module provides the temperature value for a point on the image of its radiometrically calibrated Video Source.

As described in the introduction of this Annex, Spot Measurement Modules can be defined as “Screen” elements, points on the image, or as “Geo-referenced” elements, points in the PTZ space around the Device. The use of the “AbsoluteCoords” item will allow to differentiate this behaviour.

```
<tt:AnalyticsModuleDescription Name="ttr:RadiometrySpot">
    <tt:Parameters>
        <tt:ElementItemDescription Name="RadiometrySpot"
Type="ttr:RadiometrySpotModuleConfig"/>
    </tt:Parameters>
    <tt:MessageDescription>
        <tt:Source>
            <tt:SimpleItemDescription Name="VideoSourceToken"
Type="tt:ReferenceToken"/>
```

```
            <tt:SimpleItemDescription Name="VideoAnalyticsConfigurationToken"
Type="tt:ReferenceToken"/>
            <tt:SimpleItemDescription Name="AnalyticsModuleName" Type="xs:string"/>
        </tt:Source>
        <xs:Data>
            <xs:ElementItemDescription Name="Reading"
Type="ttr:SpotTemperatureReading"/>
            <tt:SimpleItemDescription Name="TimeStamp" Type="xs:dateTime"/>
        </xs:Data>
        <tt:ParentTopic>
            tns1:VideoAnalytics/Radiometry/SpotTemperatureReading
        </tt:ParentTopic>
    </tt:MessageDescription>
</tt:AnalyticsModuleDescription>
```

The above analytics module description defines that a Spot Module instance produces Measurement Events (when new temperature values are produced by the video source), attached to the topic **tns1: VideoAnalytics/Radiometry/SpotTemperatureReading**. These **Temperature Measurement Events** shall contain the following data payload:

```
<xs:complexType name="SpotTemperatureReading">
   <xs:sequence>
      <xs:element name="RadiometryParameters" type="ttr:RadiometryParameters"
   minOccurs="0">
         <xs:annotation>
            <xs:documentation>
               Not present means Global Parameters from Thermal Service are being
            used.
            </xs:documentation>
         </xs:annotation>
      </xs:element>
      <xs:any namespace="##any" processContents="lax" minOccurs="0"
   maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="ItemID" type="tt:ReferenceToken"/>
   <xs:attribute name="SpotTemperature" type="xs:float" use="required"/>
   <xs:anyAttribute processContents="lax"/>
</xs:complexType>
```

**Table D-2 RadiometrySpot Analytics Module configuration parameters in ttr:RadiometrySpotModuleConfig**

| Parameter Name | Description |
|---|---|
| ItemID | Unique identifier for this Temperature Measurement Analytics Module. |
| Active | Indicates if the Temperature Measurement Item is enabled to provide temperature readings. |
| ScreenCoords | Screen coordinates, if spot is currently on screen. Assumes normalized screen limits (-1.0, 1.0). |
| AbsoluteCoords | Absolute orientation of the PTZ Vector with the Spot on screen. If no PTZVector is present the spot shall behave as a screen element, and stay on the same screen coordinates as the PTZ moves (like a head up display mask). If PTZVector is present the Spot element shall appear on display only when contained in the Field of View. In this case SpotScreenCoords shall be reported as relative to PTZVector. |

| | |
|---|---|
| RadiometryParameters | Not present parameter means the Device shall use its value from Global Parameters in Thermal Service. |

**Table D-3 Description of the Radiometry Spot Module event fields**

| Parameter Name | Description |
|---|---|
| SpotTemperatureReading | Data structure with the different measurement parameters and the actual temperature measurement:<br><br>ItemID<br><br>SpotTemperature: Actual value of the measured Temperature.<br><br>RadiometryParameters: Not present means Global Parameters from Thermal Service are being used. |

### D.1.2 Box Measurement Module

A Box Measurement Analytics Module provides several temperature values which are based on the temperature of the points contained within a rectangular region, defined on the image of its radiometrically calibrated Video Source.

As Spots, Box Measurement Modules can also be defined as “Screen” elements, or fixed rectangle on the image, or as “Geo-referenced” elements, representing an area in the PTZ space around the Device. Again, the use of the tt:PTZVector shall define if the Box shall behave as a screen element, or if it is assigned to a specific PTZStatus.

```
<tt:AnalyticsModuleDescription Name="ttr:RadiometryBox">
    <tt:Parameters>
       <tt:ElementItemDescription Name="RadiometryBox"
   Type="ttr:RadiometryBoxModuleConfig"/>
   </tt:Parameters>
    <tt:MessageDescription>
        <tt:Source>
            <tt:SimpleItemDescription Name="VideoSourceToken"
Type="tt:ReferenceToken"/>
            <tt:SimpleItemDescription Name="VideoAnalyticsConfigurationToken"
Type="tt:ReferenceToken"/>
            <tt:SimpleItemDescription Name="AnalyticsModuleName" Type="xs:string"/>
        </tt:Source>
        <xs:Data>
            <xs:ElementItemDescription Name="Reading"
Type="ttr:BoxTemperatureReading"/>
            <tt:SimpleItemDescription Name="TimeStamp" Type="xs:dateTime"/>
        </xs:Data>
        <tt:ParentTopic>
            tns1:VideoAnalytics/Radiometry/BoxTemperatureReading
        </tt:ParentTopic>
    </tt:MessageDescription>
</tt:AnalyticsModuleDescription>
```

The above analytics module description defines that a Box Module instance produces Measurement Events (when new temperature values are produced by the video source),

attached to the topic **tns1: VideoAnalytics/Radiometry/BoxTemperatureReading**. These **Temperature Measurement Events** shall contain the following data payload:

```
<xs:complexType name="BoxTemperatureReading">
   <xs:sequence>
      <xs:element name="RadiometryParameters" type="ttr:RadiometryParameters"
   minOccurs="0">
         <xs:annotation>
            <xs:documentation>
               Not present means Global Parameters from Thermal Service are being
            used.
            </xs:documentation>
         </xs:annotation>
      </xs:element>
      <xs:any namespace="##any" processContents="lax" minOccurs="0"
   maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="ItemID" type="tt:ReferenceToken" use="required"/>
   <xs:attribute name="MaxTemperature" type="xs:float" use="required"/>
   <xs:attribute name="MinTemperature" type="xs:float" use="required"/>
   <xs:attribute name="AverageTemperature" type="xs:float"/>
   <xs:atrribute name="MedianTemperature" type="xs:float"/>
   <xs:anyAttribute processContents="lax"/>
</xs:complexType>
```

**Table D-4 RadiometryBox Analytics Module configuration parameters in ttr:RadiometryBoxModuleConfig**

| Parameter Name | Description |
|---|---|
| ItemID | Unique identifier for this Temperature Measurement Analytics Module. |
| Active | Indicates if the Temperature Measurement Item is enabled to provide temperature readings. |
| ScreenCoords | Screen coordinates, if box is currently completely contained on the screen. Assumes normalized screen limits (-1.0, 1.0). |
| AbsoluteCoords | Absolute orientation of the PTZ Vector with the Box on screen. If no PTZVector is present the box shall behave as a screen element, and stay on the same screen coordinates as the PTZ moves (like a head up display mask). If PTZVector is present the Box element shall appear on display only when completely contained in the Field of View. In this case BoxScreenCoords shall be reported as relative to PTZVector. |
| RadiometryParameters | Not present parameter means the Device shall use its value from Global Parameters in Thermal Service. |

**Table D-5 Description of the Radiometry Box Module event fields**

| Parameter Name | Description |
|---|---|
| BoxTemperatureReading | Data structure with the different measurement parameters and the actual temperature measurement: |

| | |
|---|---|
| | ItemID<br><br>Supported Temperature Values: MaxTemperature, MinTemperature, AverageTemperature, MedianTemperature.<br><br>RadiometryParameters: Not present means Global Parameters from Thermal Service are being used. |

## D.2 Radiometry Rules

Calling the GetRuleOptions operation with the ttr:RadiometryRule type shall return an ttr:RadiometryRuleConfigOptions, as well as ttr:RadiometryTemperatureRuleConfig, if Radiometry Rules are supported by the Device.

**Table D-6 Radiometry Rule configuration options**

| Parameter Name | Description |
|---|---|
| RadiometryRuleOptions | Specifies valid ranges for the temperature condition parameters used for comparison in radiometric rules:<br><br>• Threshold Temperature<br>• ThresholdTime<br>• HysteresisTemperature |
| TemperatureConditionOptions | Specifies valid rule conditions for temperature comparisions in radiometric rules:<br><br>LessThan, MoreThan, EqualTo, Change |
| TemperatureTypeOptions | Specifies temperature measurement types provided by box measurement modules in the device:<br><br>MaxTemperature, MinTemperature, StdDeviation, MedianTemperature, ISOCoverage |

### D.2.1 Temperature Rule

A Temperature Rule is the most basic rule in Radiometry. It evaluates one of the output temperature values of a single Analytics Module, to trigger Temperature Alarm events when it meets a certain condition. The Rule shall only apply when its associated Analytics Module is Active, and providing valid readings.

```
<xs:RuleDescription Name="ttr:RadiometryTemperatureRule">
    <xs:Parameters>
        <xs:ElementItemDescription Name="RadiometryTemperatureRule"
Type="ttr:RadiometryTemperatureRuleConfig"/>
    </xs:Parameters>
    <xs:MessageDescription IsProperty="true">
        <xs:Source>
            <xs:SimpleItemDescription Name="VideoSourceToken"
Type="tt:ReferenceToken"/>
            <xs:SimpleItemDescription Name="VideoAnalyticsConfigurationToken"
Type="tt:ReferenceToken"/>
            <xs:SimpleItemDescription Name="RuleName" Type="xs:string"/>
```

```
        </xs:Source>
        <xs:Key>
            <xs:SimpleItemDescription Name="RadiometryModuleID"
Type="tt:ReferenceToken"/>
        </xs:Key>
        <xs:Data>
            <xs:SimpleItemDescription Name="AlarmActive" Type="xs:boolean"/>
                      <tt:SimpleItemDescription Name="TimeStamp"
Type="xs:dateTime"/>
            <xs:ElementItemDescription Name="AlarmGeoLocation"
Type="tt:GeoLocation" minOccurs=0/>
        </xs:Data>
        <xs:ParentTopic>
            tns1:RuleEngine/Radiometry/TemperatureAlarm
        </xs:ParentTopic>
    </xs:MessageDescription>
</xs:RuleDescription>
```

The above rule description defines that a Temperature Rule instance produces Temperature Alarm Events (when temperature conditions, as specified in the rule, are met), attached to the topic **tns1:RuleEngine/Radiometry/TemperatureAlarm**.

**Table D-7 RadiometryTemperatureRule configuration parameters in ttr:RadiometryTemperatureRuleConfig**

| Parameter Name | Description |
|---|---|
| RadiometryModuleID | Reference Token to the Measurement Analytics Module on which output the rule is defined. |
| TemperatureType | Indicates which of the temperature values provided by the Analytics Module shall be used by the rule. For Analitics Modules that provide a single Temperature output, this parameter shall be ignored, and therefore is optional. |
| RuleCondition | Indicates the type of temperature condition to check. |
| ThresholdTemperature | Indicates the temperature value the rule shall be checked against. |
| ThresholdTime | Indicates the time interval during which the rule condition shall be met to trigger an event. |
| HysteresisTemperature | Indicates the width in Kelvin of the temerature hysteresis band to be considered by the rule. |
| Enabled | Indicates if the Temperature Rule is enabled to provide temperature alarms. |

**Table D-8 Description of the Radiometry Temperature Rule event fields**

| Parameter Name | Description |
|---|---|
| AlarmActive | True (temperature condition is met) or False (condition not met) |
| TimeStamp | Date and time in which the rule's temperature condition was met. |

| | |
|---|---|
| AlarmGeoLocation | Geo Location of the measurement triggering the Alarm (if available) |

# Annex E. Tampering Detection
(normative)

## E.1 Tampering Detection

Tampering detection rule detects any kind of tampering to the image sensor, the Mode paramer defines the tampering detection type, tampering detection rule and options follows the generic description language defined in 5.1.3.10. For description of parameters refer Table E-1.

```
<tt:RuleDescription Name="tt:TamperingDetection">
   <tt:Parameters>
     <tt:SimpleItemDescription Name="Mode" Type="xs:string"/>
     <tt:SimpleItemDescription Name="Threshold" Type="xs:float"/>
     <tt:SimpleItemDescription Name="Duration" Type="xs:duration"/>
   </tt:Parameters>
</tt:RuleDescription>
```

It can generate any of the following events based on the Mode parameter,

```
<tt:Messages>
   <tt:Source>
      <tt:SimpleItemDescription Type="tt:ReferenceToken" Name="VideoSource"/>
      <tt:SimpleItemDescription Type="xs:string" Name=" RuleName" />
   </tt:Source>
   <tt:Data>
      <tt:SimpleItemDescription Type="xsd:boolean" Name="State"/>
   </tt:Data>
   <tt:ParentTopic>tns1:VideoSource/GlobalSceneChange/AnalyticsService</tt:ParentTo
pic>
</tt:Messages>
<tt:Messages>
   <tt:Source>
      <tt:SimpleItemDescription Type="tt:ReferenceToken" Name="VideoSource"/>
      <tt:SimpleItemDescription Type="xs:string" Name=" RuleName" />
   </tt:Source>
   <tt:Data>
      <tt:SimpleItemDescription Type="xsd:boolean" Name="State"/>
   </tt:Data>
   <tt:ParentTopic>tns1:VideoSource/ImageTooDark/AnalyticsService</tt:ParentTopic>
</tt:Messages>
<tt:Messages>
   <tt:Source>
      <tt:SimpleItemDescription Type="tt:ReferenceToken" Name="VideoSource"/>
      <tt:SimpleItemDescription Type="xs:string" Name=" RuleName" />
   </tt:Source>
   <tt:Data>
      <tt:SimpleItemDescription Type="xsd:boolean" Name="State"/>
   </tt:Data>
   <tt:ParentTopic>tns1:VideoSource/ImageTooBlurry/AnalyticsService</tt:ParentTopic
>
</tt:Messages>
<tt:Messages>
   <tt:Source>
      <tt:SimpleItemDescription Type="tt:ReferenceToken" Name="VideoSource"/>
      <tt:SimpleItemDescription Type="xs:string" Name=" RuleName" />
   </tt:Source>
   <tt:Data>
      <tt:SimpleItemDescription Type="xsd:boolean" Name="State"/>
   </tt:Data>
   <tt:ParentTopic>tns1:VideoSource/ImageTooBright/AnalyticsService</tt:ParentTopic
>
</tt:Messages>
<tt:Messages>
   <tt:Source>
      <tt:SimpleItemDescription Type="tt:ReferenceToken" Name="VideoSource"/>
      <tt:SimpleItemDescription Type="xs:string" Name=" RuleName" />
```

```
    </tt:Source>
    <tt:Data>
        <tt:SimpleItemDescription Type="xsd:boolean" Name="State"/>
    </tt:Data>
    <tt:ParentTopic>tns1:VideoSource/SignalLoss</tt:ParentTopic>
</tt:Messages>
```

**Table E-1 Tampering rule configuration parameters**

| Parameter Name | Description |
|---|---|
| Mode | Indicates any of the supported modes listed in the GetRuleOptionsResponse. |
| Threshold | Indicates the threshold level of the tampering detection algorithm, when the threshold crosses the specified value event is generated. The value is set according to the limits specified in the GetRuleOptionsResponse. |
| Duration | How long the detection values stays over the threshold to generate the event. This duration value is set according to the DurationRange specified in the GetRuleOptionsResponse. |

Sample options for the tampering detection, based on description language parameter options defined in 5.2.3.2.

```
<tan:GetRuleOptionsResponse>
    <tan:RuleOptions Name="Mode" RuleType="tt:TamperingDetection">
        <tt:StringItems>
            <tt:Item>GlobalSceneChange</tt:Item>
            <tt:Item>ImageTooDark</tt:Item>
            <tt:Item>ImageTooBright</tt:Item>
            <tt:Item>SignalLoss</tt:Item>
            <tt:Item>ImageTooBlurry</tt:Item>
        </tt:StringItems>
    </tan:RuleOptions>
    <tan:RuleOptions Name="Threshold" RuleType="tt:TamperingDetection">
        <tt:FloatRange>
            <tt:Min>0.0</tt:Min>
            <tt:Max>100.0</tt:Max>
        </tt:FloatRange>
    </tan:RuleOptions>
    <tan:RuleOptions Name="Duration" RuleType="tt:TamperingDetection">
        <tt:DurationRange >
            <tt:Min>PT1S</tt:Min>
            <tt:Max>PT1M</tt:Max>
        </tt:DurationRange>
    </tan:RuleOptions>
</tan:GetRuleOptionsResponse>
```

# Annex F. Face metadata values samples
(informative)

## F.1 Face metadata values samples

The main purpose for generating face descriptions is to be able to compare the output to manually generated characteristics. Note that these manually generated characteristics may differ between countries and are not calibrated by ONVIF.

**Table F- 1 Samples For FacialShape**

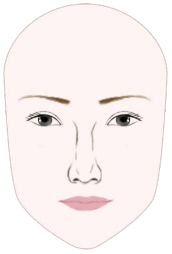
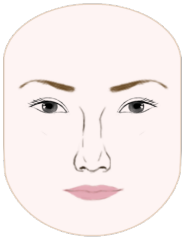
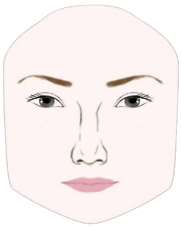
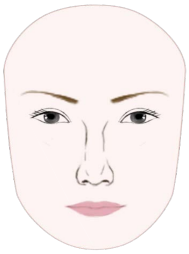
| Feature | values | sample |
|---|---|---|
| FacialShape | Long |  |
| | Round |  |
| | Square |  |
| | Oval |  |

**Table F- 2 Samples For HairStyle**

| Feature | Values | sample |
|---|---|---|
| HairStyle | Straight | |
| | Wave | |
| | Curly | |
| | CrewCut | |
| | Bald | |

| | | |
|---|---|---|
| | Ponytail | |
| | Pigtail | |

**Table F- 3 Samples For EyebrowSpace**

| Feature | values | sample |
|---|---|---|
| EyebrowSpace | Joint | |
| | Separate | |

**Table F- 4 Samples For EyeShape**

| Feature | values | sample |
|---|---|---|
| EyeShape | Almond | |
| | Slitty | |
| | Round | |

**Table F- 5 Samples For Ear**

| Feature | values | sample |
|---|---|---|
| Ear | Round | |
| | Pointed | |
| | Narrow | |
| | BroadLobe | |

**Table F- 6 Samples For NoseEnd**

| Feature | values | sample |
|---|---|---|
| NoseEnd | Snub | |

| | | |
|---|---|---|
| | Turnedup | |
| | Flat | |
| | Hooked | |

**Table F- 7 Samples For FacialHair**

| Feature | values | sample |
|---|---|---|
| FacialHair | Mustache | |
| | Beard | |

| | | |
|---|---|---|
| | Sideburn | |

**Table F- 8 Samples For Chin**

| Feature | values | sample |
|---|---|---|
| Chin | Double | |
| | Pointed | |
| | Round | |

**Table F- 9 Samples For Accessory**

| Feature | values | sample |
|---|---|---|
| Accessory | Opticals | |

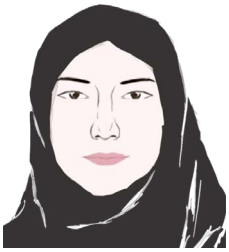
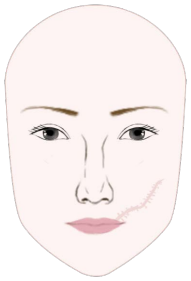
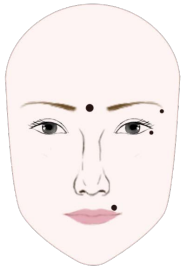
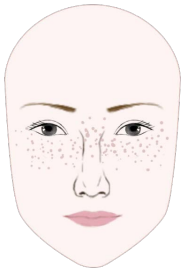
| | | |
|---|---|---|
| | Hat | |
| | Helmet | |
| | Mask | |
| | Hijab |  |
| | RightEyePatch | |
| | LeftEyePatch | |

**Table F- 10 Samples For AdditionalFeatures**

| Feature | values | sample |
|---|---|---|
| AdditionalFeatures | Scar |  |
| | Mole |  |
| | Tattoo | |
| | Freckles |  |

# Annex G. Recognition rule engines
(normative)

This annex describes rule engines related to recognition in video analytics, e.g. Face Recognition and License Plate Recognition.

## G.1 Generic parameters for recognition rule engines

This section describes generic parameters for configuration of a recognition rule engine. It also describes common event fields included in an event produced by a recogition rule engine. For concrete examples of how these parameters are used, please look at subsequent sections about Face Recognition and License Plate Recognition.

**Table G-1 Generic recognition rule configuration parameters**

| Parameter Name | Description | Requirement Level |
|---|---|---|
| IncludeImage | Configures how recognized images should be conveyed as type tt:ImageSendingType.<br>• Embedded: Embed a recognized image into the event payload.<br>• LocalStorage: Include a local URI for the recognized image.<br>• RemoteStorage: Include a remote URI for the recognized image. | Optional |

Options for the rule parameters are described by the generic mechanism defined in section 5.2.3.

**Table G-2 Generic recognition event fields**

| Event Field | Description | Requirement Level |
|---|---|---|
| Likelihood | Likelihood of accuracy of recognition between 0~1 | Mandatory |
| BoundingBox | Recognized object bounding box details | Optional |
| Label | Information regarding matching list.<br>Predefined values are:<br>• Whitelisted<br>• Blacklisted<br>• Temporary | Optional |
| Image | Base64 encoded binary image data of the recognized object | Optional |
| ImageUri | URI to the image of the recognized object | Optional |

If the IncludeImage configuration parameter is set to an empty string, device shall not include Image and ImageUri event fields in the event message.

If the IncludeImage configuration parameter is set to Embedded, device shall include the Image event field in the event message.

If the IncludeImage configuration parameter is set to LocalStorage or RemoteStorage, device shall include the ImageUri event field in the event message.

## G.2 Face Recognition

This section describes Face Recognition rule engine that produces an event with the topic “tns1:RuleEngine/Recognition/Face. The Face Recognition rule engine triggers the event when the identified face matches with a reference face image, which is described using the Rule Description Language as:

```
<tt:RuleDescription Name="tt:FaceRecognition">
  <tt:Parameters>
    <tt:SimpleItemDescription Name="IncludeImage" Type=”xs:string"/>
  </tt:Parameters>
  <tt:Messages>
    <tt:Source>
       <tt:SimpleItemDescription Name=”VideoSourceConfiguration”
       Type=”tt:ReferenceToken”/>
       <tt:SimpleItemDescription Name="VideoAnalyticsConfiguration"
       Type="tt:ReferenceToken"/>
       <tt:SimpleItemDescription Name=”RuleName” Type=”xs:string”/>
    </tt:Source>
    <tt:Data>
       <tt:SimpleItemDescription Name ="Likelihood" Type="xs:float"/>
       <tt:SimpleItemDescription Name ="Label" Type="xs:string "/>
       <tt:SimpleItemDescription Name ="ImageUri" Type="xs:anyURI"/>
       <tt:SimpleItemDescription Name="EnrollmentID" Type="xs:string"/>
       <tt:SimpleItemDescription Name ="RefImageUri" Type="xs:anyURI"/>
       <tt:ElementItemDescription Name ="Image" Type="xs:base64Binary"/>
       <tt:ElementItemDescription Name ="BoundingBox" Type="tt:Rectangle"/>
    </tt:Data>
    <tt:ParentTopic>
      tns1:RuleEngine/Recognition/Face
    </tt:ParentTopic>
  </tt:Messages>
</tt:RuleDescription>
```

The Face Recognition event consists of the generic event fields from Table F-1-3 as well as the specific event fields listed in table F-2-1.

**Table G3 Face Recognition event fields**

| Event Field | Description | Requirement Level |
|---|---|---|
| EnrollmentID | Unique ID to identify a person. | Mandatory |
| RefImageUri | URI of reference face image against which face recognition is successfully matched. | Optional |

## G.3 License Plate Recognition

This section describes License Plate Recognition rule engine that produces an event with the topic “tns1:RuleEngine/Recognition/LicensePlate”. License Plate Recognition detects vehicles’ features or behaviours against specified region or direction. The rule can be configured to detect vehicle in lane/region or driving direction of interest on the image source.

License Plate Recognition rule engine triggers an event which is described using the Rule Description Language as :

```
<tt:RuleDescription Name="tt:LicensePlateRecognition">
  <tt:Parameters>
    <tt:SimpleItemDescription Name="IncludeImage" Type=”xs:string"/>
    <tt:SimpleItemDescription Name="PlateLocation" Type="xs:string"/>
    <tt:ElementItemDescription Name=”Region” Type=”tt:Polygon”/>
    <tt:ElementItemDescription Name=”SnapLine” Type=”tt:Polyline”/>
  </tt:Parameters>
  <tt:Messages>
    <tt:Source>
       <tt:SimpleItemDescription Name=”VideoSourceConfiguration”
       Type=”tt:ReferenceToken”/>
       <tt:SimpleItemDescription Name="VideoAnalyticsConfiguration"
       Type="tt:ReferenceToken"/>
       <tt:SimpleItemDescription Name=”RuleName” Type=”xs:string”/>
    </tt:Source>
    <tt:Data>
       <tt:SimpleItemDescription name="Likelihood" Type="xs:float"/>
       <tt:SimpleItemDescription Name="Label" Type="xs:string"/>
       <tt:SimpleItemDescription Name=”ImageURI” Type=”xs:anyURI”/>
       <tt:SimpleItemDescription Name=”VehicleImageURI” Type=”xs:anyURI”/>
       <tt:ElementItemDescription Name=”BoundingBox” Type=”tt:Rectangle”/>
       <tt:ElementItemDescription Name=”Image” Type=”xs:base64Binary/>
       <tt:ElementItemDescription Name=”LicensePlateInfo”
       Type=”tt:LicensePlateInfo”/>
       <tt:ElementItemDescription Name=”VehicleInfo” Type=”tt:VehicleInfo”/>
       <tt:ElementItemDescription Name=”VehicleImage” Type=”xs:base64Binary/>
    </tt:Data>
    <tt:ParentTopic>
      tns1:RuleEngine/Recognition/LicensePlate
    </tt:ParentTopic>
  </tt:Messages>
</tt:RuleDescription>
```

License Plate Recognition configuration parameters consists of the generic configuration parameters listed in Table F-1-1 as well as the specific configuration parameters listed in Table F-3-1.

**Table G4 License Plate Recognition rule configuration parameters**

| Parameter Name | Description | Requirement Level |
|---|---|---|
| Region | Defines a region that should be analyzed. | Optional |
| SnapLine | Defines a polyline to indicate the best snap location in the camera’s field of view. | Optional |
| PlateLocation | License plate to be analyzed. Predefined values are 'Front' and 'Rear'. | Optional |

Options for the rule parameters are described by the generic mechanism defined in section 5.2.3.

The License Plate Recognition event consists of the generic event fields from Table F-1-3 as well as the specific event fields listed in Table F-3-2.

**Table G5 License Plate Recognition event fields**

| Event Field | Description | Requirement Level |
|---|---|---|
| LicensePlateInfo | Provides information of the recognized license plate. | Mandatory |

| | | |
|---|---|---|
| VehicleInfo | Provides information of the recognized vehicle. | Optional |
| VehicleImage | Image data of the recognized vehicle. | Optional |
| VehicleImageURI | URI of recognized vehicle image. | Optional |

If the IncludeImage configuration parameter is set to an empty string, device shall not include VehicleImage and VehicleImageURI event fields in the event message.

If the IncludeImage configuration parameter is set as Embedded, device shall include VehicleImage event field in the event message.

If the IncludeImage configuration parameter is set as LocalStorage or RemoteStorage, device shall include VehicleImageURI event field in the event message.

## Annex H. Revision History

| Rev. | Date | Editor | Changes |
|---|---|---|---|
| 2.1 | Jul-2011 | Hans Busch | Split from Core 2.0 without change of content. |
| 2.1.1 | Jan-2012 | Hans Busch | Change Requests 505, 535 |
| 2.2 | May-2012 | Hasan T. Ozdemir | Incorporated 2.2 enhancements |
| 2.2.1 | Dec-2012 | Hans Busch | Change Request 708 |
| 2.4 | Aug-2013 | Hasan T. Ozdemir | Generalized from VideoAnalytics. Added Audio events. |
| 2.4.1 | Dec-2013 | Michio Hirai | Change Request 1164, 1193 |
| 2.4.2 | Jun-2014 | Michio Hirai | Change Request 1305 |
| 2.5 | Dec-2014 | Hasan T. Ozdemir<br>Michio Hirai | Added A6 Query Rule<br>Change Request 1559 |
| 16.06 | Jun-2016 | Hiroyuki Sano | Change Request 1821, 1825, 1844 |
| 16.12 | Dec-2016 | Steven Dillingham<br>Hiroyuki Sano | Added Motion Region Detection Rules: 5.2.3.6, Annex C<br>Change Request 1871 |
| 17.06 | Jun-2017 | Hans Busch<br>Hiroyuki Sano | Updated method layouts. Apply 2089<br>Change Request 1843, 1984, 2042, 2052, 2072, 2114, 2137<br>Added GetAnalyticsModuleOptions & Annex D |
| 17.12 | Dec-2017 | Hiroyuki Sano | Change Request 2177, 2204 |
| 18.06 | Jun-2018 | Qian Zhou<br><br>Hiroyuki Sano | Added Vehicle information, Speed, License plate information descriptor<br>Change Request 2234, 2235, 2245, 2305, 2306 |
| 18.12 | Dec-2018 | Hiroyuki Sano | Change Request 2322, 2354, 2363, 2376, 2385, 2437 |
| 19.06 | Jun-2019 | Sujith Raman<br><br>Hiroyuki Sano<br>Hans Busch | Added PTZ Motion Region Configuration, Tampering Configuration<br>Change Request 2462, 2484<br>Change Request 2465 |
| 19.12 | Dec-2019 | Haina Han<br><br>Hiroyuki Sano | Added Image Sending, Object Classification, Face and Human body descriptor<br>Change Request 2504, 2527, 2528, 2555, 2556, 2565, 2566, 2567, 2578, 2628 |
| 20.06 | Jun-2020 | Michio Hirai<br><br>Sujith Raman<br>Hainha Han, Sriram Bhetanabottla | Change Request 2695, 2640, 2641, 2669, 2671, 2678, 2682, 2685, 2690, 2696, 2698, 2686<br>Added PolyLine and Polygon options<br>Added Recognition rule engines |