ONVIF™
Analytics Service Specification

Version 23.12

December, 2023

CONTENTS

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>

RFC 7159 The JavaScript Object Notation (JSON) Data Interchange Format
<https://www.rfc-editor.org/rfc/rfc7159.txt>

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

Pre-fix	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

Prefix	Namespace URI	Description
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.

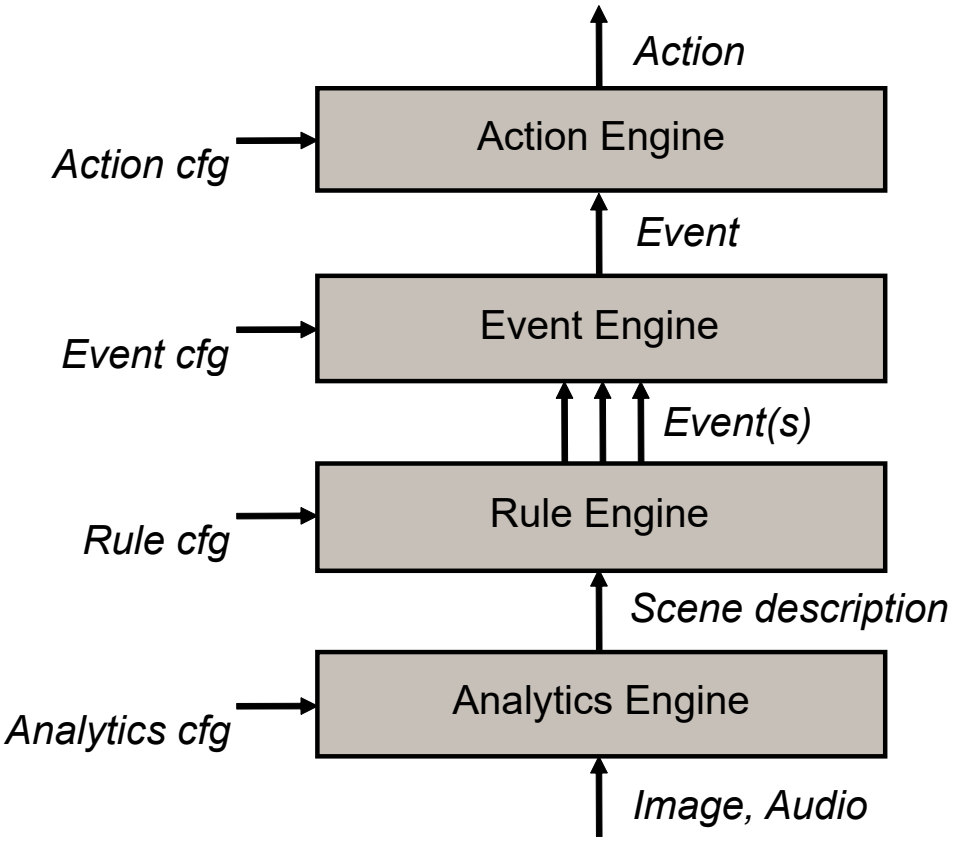


Figure 1: Analytics architecture

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. For easy consumption by non-traditional clients (e.g. cloud vendors) and to bridge gap between traditional surveillance systems and IoT ecosystems metadata including scene description can be published in JSON format over the MQTT protocol.

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 Scene Description

5.1 Overview

This chapter defines the XML-based scene description, which can be streamed as metadata to clients via RTP as defined in the ONVIF Streaming Specification.

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.

Section 5.4 defines an alternate JSON based representation.

5.2 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 5.2.1 and 5.2.2. The appearance and behaviour of tracked objects is discussed in section 5.3.1. Interactions between objects like splits and merges are described in section 5.3.2.

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"/>
```

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

```
    <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 5.2.1 describes how frames processed by the analytics algorithm are referenced within the analytics stream.

5.2.1 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.

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>
```

5.2.2 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 normalized coordinate system is shown in Figure 2.

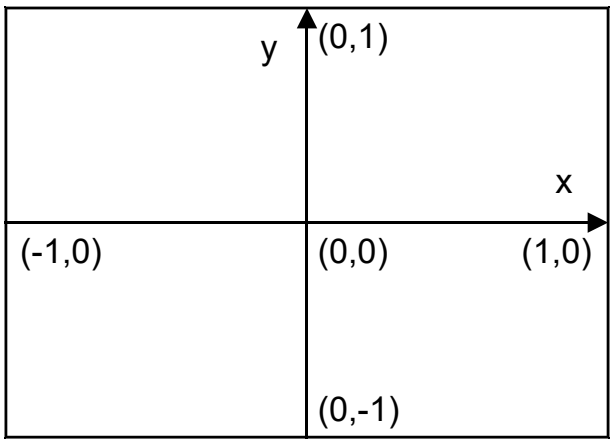


Figure 2: Normalized frame coordinate system

This specification allows modification of the coordinate system for individual nodes of the XML tree. As a result, each frame node starts with the normalized 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 $p = \begin{pmatrix} p_x \\ p_y \end{pmatrix}$ given with respect to this coordinate system is transformed into the corresponding point $q = \begin{pmatrix} q_x \\ q_y \end{pmatrix}$ of the normalized 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 normalized 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:

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

$\begin{pmatrix} t'_x \\ t'_y \end{pmatrix} = \begin{pmatrix} v_x \cdot s_x + t_x \\ v_y \cdot s_y + t_y \end{pmatrix}$, $\begin{pmatrix} s'_x \\ s'_y \end{pmatrix} = \begin{pmatrix} u_x \cdot s_x \\ u_y \cdot s_y \end{pmatrix}$, 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>
```

5.3 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.

5.3.1 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

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

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 5.3.3), color (see 5.3.4) and object class (see 5.3.5) 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>
      <tt:Shape>
        <tt:BoundingBox left="20.0" top="80.0" right="100.0" bottom="30.0"/>
        <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="80.0" right="100.0" bottom="30.0"/>
        <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="80.0" right="105.0" bottom="30.0"/>
        <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="80.0" right="100.0" bottom="30.0"/>
        <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="180.0" right="100.0" bottom="30.0"/>
        <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="150.0" right="70.0" bottom="100.0" />
          <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>
```

5.3.2 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>
```

5.3.3 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="80.0" right="100.0" bottom="30.0"/>
        <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>
```

5.3.4 Color descriptor

The optional color descriptor allows to describe the representative colors of the detected object or region. Each color is represented as a three-dimensional vector based on a color space. Its weight denotes the fraction of pixels assigned to the representative color in the range [0 .. 1]. Color covariance describes the variation of color values around the representative color value in color space thus representative color denotes a region in color space.

Note that the color descriptor does not specify, how color clusters are created. They can represent bins of a color histogram or the result of a clustering algorithm.

The color space defaults to the YCbCr color space which 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 Colorspace URI for the YCbCr color space is http://www.onvif.org/ver10/colorspace/YCbCr. Alternate color spaces as defined in Table 3 can be defined on frame level and individually for each object.

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

```
<tt:Frame UtcTime="2010-09-10T12:24:57.721" Colorspace="RGB">
  <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="80.0" right="100.0" bottom="30.0"/>
        <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>0.5</tt:Weight>
        </tt:ColorCluster>
        <tt:ColorCluster>
          <tt:Color X="165" Y="44" Z="139"/>
          <tt:Covariance XX="4" YY="4" ZZ="4"/>
          <tt:Weight>0.3</tt:Weight>
        </tt:ColorCluster>
      </tt:Color>
    </tt:Appearance>
  </tt:Object>
</tt:Frame>
```

An alternative representation of color can be used by adding a likelihood attribute to the color element.

Its likelihood denotes how confident the algorithm is that the color is correct. The range of likelihood is [0 .. 1].

When using the likelihood attribute the weight and covariance should not be used as they are missing a meaningful definition in this case.

When using the likelihood attribute the meaning of multiple color clusters in the same color descriptor changes. Multiple color clusters with likelihood should be interpreted as different color candidates with different likelihoods, not as different regions of colors on the object.

An example metadata containing color information of the detected object using a color cluster with a likelihood attribute is given below.

```
<tt:Frame UtcTime="2010-09-10T12:24:57.721" Colorspace="RGB">
  <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="80.0" right="100.0" bottom="30.0"/>
        <tt:CenterOfGravity x="60.0" y="50.0"/>
      </tt:Shape>
      <tt:Color>
        <tt:ColorCluster>
          <tt:Color X="0" Y="0" Z="255" Likelihood="0.93"/>
        </tt:ColorCluster>
        <tt:ColorCluster>
          <tt:Color X="255" Y="255" Z="255" Likelihood="0.87"/>
        </tt:ColorCluster>
        <tt:ColorCluster>
          <tt:Color X="0" Y="0" Z="0" Likelihood="0.12"/>
        </tt:ColorCluster>
        <tt:ColorCluster>
          <tt:Color X="0" Y="128" Z="0" Likelihood="0.08"/>
        </tt:ColorCluster>
        <tt:ColorCluster>
          <tt:Color X="255" Y="0" Z="0" Likelihood="0.04"/>
        </tt:ColorCluster>
      </tt:Color>
    </tt:Appearance>
  </tt:Object>
</tt:Frame>
```

The following table lists the acceptable values for colorspace attribute

Table 3: Colorspace namespace values

Color-space	Namespace URI	Description
YCbCr	http://www.onvif.org/ver10/colorspace/YCbCr	YCbCr color space
RGB	http://www.onvif.org/ver10/colorspace/RGB	RGB color space
CIELUV	http://www.onvif.org/ver10/colorspace/CIELUV	**Deprecated** CIE LUV
CIELAB	http://www.onvif.org/ver10/colorspace/CIELAB	**Deprecated** CIE 1976 (L*a*b*)
HSV	http://www.onvif.org/ver10/colorspace/HSV	**Deprecated** HSV color space

Table 4 lists the acceptable value ranges for prominent color spaces.

Table 4: Colorspace value ranges

Colorspace	Range
YCbCr	Y [16 to 235] and Cb/Cr[16 to 240]
RGB	For all component values [0...255]

5.3.5 Object Class descriptor

A Class Descriptor is defined as an optional element of the appearance node of an object 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 range of likelihood is [0..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="80.0" right="100.0" bottom="30.0"/>
        <tt:CenterOfGravity x="60.0" y="50.0"/>
      </tt:Shape>
      <tt:Class>
        <tt:Type Likelihood=”0.8”>Vehicle</tt:Type>
        <tt:Type Likelihood=”0.75”>Car</tt:Type>
        <tt:Type Likelihood=”0.3”>Truck</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.

5.3.6 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 5.2.2. 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 5: 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. 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). 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 12639 (TIFF, Revision 6.0).

5.3.7 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 6. 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 6: 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.

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

```
<tt:VideoAnalytics>
  <tt:Frame UtcTime="2020-11-05T12:24:57.321">
    <tt:Object ObjectId="45">
      <tt:Appearance>
        <tt:VehicleInfo>
          <tt:Type Likelihood="0.9">Car</tt:Type>
          <tt:Brand Likelihood="0.7">Volvo</tt:Brand>
          <tt:Model Likelihood="0.6">XC60</tt:Model>
        </tt:VehicleInfo>
      </tt:Appearance>
    </tt:Object>
  </tt:Frame>
</tt:VideoAnalytics>
```

5.3.8 Speed descriptor

The Speed element 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>
```

5.3.9 Spherical coordinate descriptor

The spherical coordinates gives the position of the object *relative* to the device as three coordinates: distance (r), elevation angle (θ) and azimuth angle (φ). The origin is the device itself and the x axis is rightwards, the y axis is away from the device and the z axis is upwards. The spherical coordinates are described according to Figure 3, where the distance (r) is the number of meters from the origin, elevation angle (θ) is the elevation from the x-y plane and the azimuth angle (φ) is the rotation around the z axis, counter clockwise. The range for the elevation angle is -90 to 90 degrees and for the azimuth angle -180 to 180 degrees.

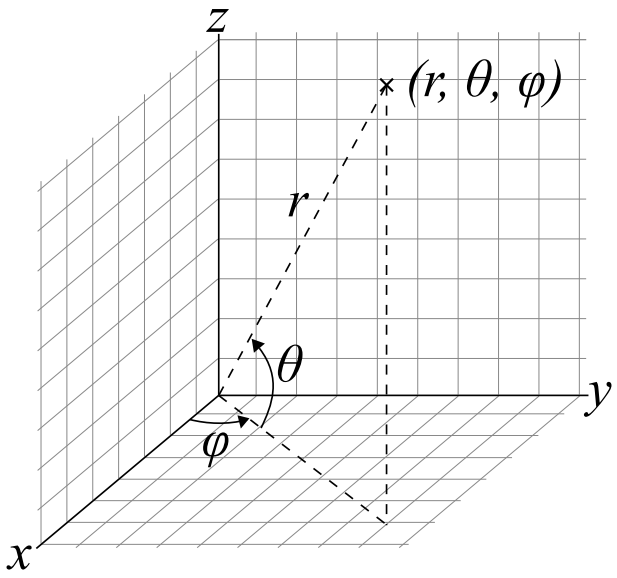


Figure 3: Spherical coordinates

An example metadata containing spherical coordinates of a detected object.

```
<tt:VideoAnalytics>
  <tt:Frame UtcTime="2016-9-10T12:24:57.321">
    <tt:Object ObjectId="23">
      <tt:SphericalCoordinate Distance="15.5" ElevationAngle="70.6" AzimuthAngle="50.7"/>
    </tt:Object>
  </tt:Frame>
</tt:VideoAnalytics>
```

5.3.10 Direction descriptor

The Direction element describes the movement direction of the object as a GeoOrientation type with the angles yaw (ψ) and pitch (θ). The origin is the device itself and the x axis is rightwards, the y axis is away from the device and the z axis is upwards, i.e. "body frame" coordinate system according to Figure 4. The range for yaw

is between -180 and +180 degrees, where 0 is rightwards and 90 is away from the device. The range for pitch is between 90 and -90 degrees where 90 is upwards.

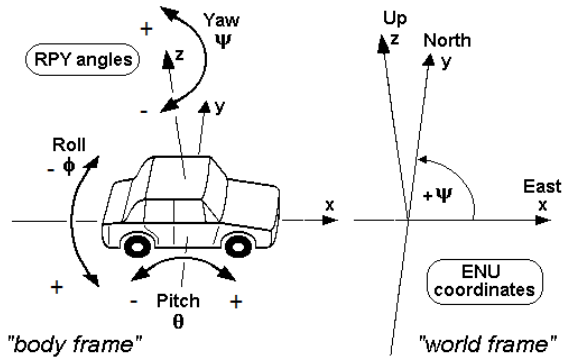


Figure 4: Orientation on earth surface[4]

An example metadata containing direction 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:Direction yaw=38.8 pitch=15.5/>
    </tt:Behaviour>
  </tt:Object>
</tt:Frame>
</tt:VideoAnalytics>
```

5.3.11 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 7. For each element, there is a class, which is associated 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 7: 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”, ”Diplomat”.
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">
```

[4]Source: By Qniemiec, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=10893168

```
      <tt:Appearance>
        <tt:LicensePlateInfo>
          <tt:PlateNumber Likelihood="0.9">6DZG261</tt:PlateNumber>
          <tt:PlateType Likelihood="0.8">Normal</tt:PlateType>
          <tt:CountryCode Likelihood="0.7">US</tt:CountryCode>
          <tt:IssuingEntity Likelihood="0.6">California</tt:IssuingEntity>
        </tt:LicensePlateInfo>
      </tt:Appearance>
    </tt:Object>
  </tt:Frame>
</tt:VideoAnalytics>
```

5.3.12 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. When embedding Image Data as base64Binary, image format shall be either JPG or PNG (On receiving end, start bytes may be used to identify the image type). 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>
```

5.3.13 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 8: 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

Feature	Sub feature	Type	Defined Value
	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 5.

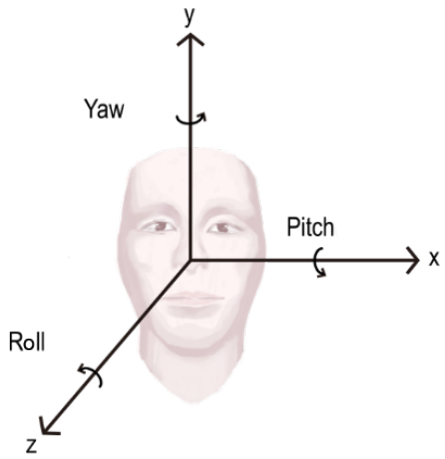


Figure 5: Pose angles with respect to the frontal view of the subject

5.3.14 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 describe some features of the human body, such as body metric, clothing, belonging and behavior. All features are listed in the below table.

Table 9: 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	tt:ColorDescriptor	N/A
		Grain	xs:string	bd:Grain
		Style	xs:string	bd:BottomsStyle
	Shoes	Category	xs:string	bd:ShoesCategory
		Color	tt:ColorDescriptor	N/A
Belonging	Bag	Category	xs:string	bd:KnapsackCategory
		Color	tt:ColorDescriptor	N/A
	Umbrella	Color	tt:ColorDescriptor	N/A
		Open	xs:boolean	N/A

Feature	Sub Feature	Element	Type	Defined Value
	LiftSomething	N/A	xs:boolean	N/A
	Box	Color	tt:ColorDescriptor	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	N/A	xs:string	bd:HumanActivity

5.3.15 Barcode information descriptor

A barcode information descriptor is defined as an optional element of the Appearance Node of an Object Node. It is used to give details of the barcode, like barcode data, type and pixel per module. The details are shown in Table 10

Table 10: Description of the barcode information

BarcodeInfo	Description
Data	Information encoded in barcode
Type	Barcode type Code-128, PDF417, QRCode etc., Acceptable values are defined in tt:BarcodeType.
PPM	Pixel per module(PPM) refers to how many pixels it will take to cover one cell or module of the code.

An example metadata containing the Barcode information is given below

```
<tt:VideoAnalytics>
  <tt:Frame UtcTime="2022-12-21T12:25:51.211">
    <tt:Object ObjectId="12">
      <tt:Appearance>
        <tt:Class>
          <tt:Type Likelihood="0.9">Barcode</tt:Type>
        </tt:Class>
        <tt:BarcodeInfo>
          <tt:Data Likelihood="0.9">Content</tt:Data>
          <tt:Type Likelihood="0.9">Code-128</tt:Type>
          <tt:PPM>1.4</tt:PPM>
        </tt:BarcodeInfo>
      </tt:Appearance>
    </tt:Object>
  </tt:Frame>
</tt:VideoAnalytics>
```

5.4 JSON over MQTT

5.4.1 Metadata publish sequence

MQTT is a pub/sub messaging protocol centered around topics. Consumers can subscribe to topics published by producers to get continuous updates on the data content. ONVIF metadata, like video analytics scene descriptions, can be published to an MQTT broker and subscribed to by a consumer.

This is sequence of operations to set up the metadata flow:

- An ONVIF client calls AddEventBroker in the device's Event service to set up:
    - Address and credentials to the MQTT broker to use
    - A unique TopicPrefix for the device
    - A MetadataFilter listing topics to which the device shall publish
- The producer (ONVIF device) connects to the MQTT broker and starts publishing data on the selected topics.
- The consumer connects to the MQTT broker, specifying which topics that it wants to subscribe to. A consumer can be any MQTT capable client, for example, an event engine like Node-RED, IoT cloud platforms, VSaaS platforms, etc. Multiple consumers using different topics are possible.
- The consumer receives updates on subscribed topics and can implement actions like statistics aggregation and complex analytics rules with data from multiple sources.

The following section describe how topics shall be structured by an ONVIF metadata producer when published to an MQTT broker. The payload shall be encoded as defined in Annex E of the ONVIF Core Specification.

5.4.2 MQTT topic structure

This section describes how an MQTT topic is structured based on metadata type and source expressed as ABNF rules according to RFC 5234.

Topic = TopicPrefix "/" PayloadPrefix "/" MetadataType "/" MetadataProducer

Note that special characters like, ‘/’, ‘#’ and ‘+’ shall be omitted from the topic.

with

- TopicPrefix - uniquely identifies the producer and is configurable through the AddEventBroker command in the Event service. It shall not be empty.
- PayloadPrefix - signals in what format the data is published. See Table 11 below for possible values.
- MetadataType - signals what type of metadata it is. See Table 12 below for possible values.
- MetadataProducer - specifies additional source information for the metadata. The format depends on the metadata type. See Table 12 below for definition.

Table 11: PayloadPrefix types

Prefix	Description
"onvif-mj"	ONVIF metadata with JSON payload

Table 12: MetadataType categories with corresponding MetadataProducer

MetadataType	MetadataProducer (template in ABNF)	Description
"VideoAnalytics"	ProfileToken "/" AnalyticsModuleName	ONVIF metadata from a VideoAnalytics module active via VideoAnalyticsConfiguration included in Profile configuration
"AudioAnalytics"	ProfileToken "/" AnalyticsModuleName	ONVIF metadata from an AudioAnalytics module active via AudioAnalyticsConfiguration included in Profile configuration

MetadataType	MetadataProducer (template in ABNF)	Description
"PTZ"	ProfileToken "/" PTZConfigurationToken	ONVIF metadata from a PTZ node included in Profile configuration

with

- ProfileToken = The media profile token
- AnalyticsModuleName = The name of the analytics module
- PTZConfigurationToken = The PTZ configuration token

5.4.3 Example

Sample metadata frame in XML format:

```
<?xml version="1.0" encoding="UTF-8"?>
<tt:MetadataStream xmlns:tt="http://www.onvif.org/ver10/schema"
xmlns:fc="http://www.onvif.org/ver20/analytics/humanface"
xmlns:bd="http://www.onvif.org/ver20/analytics/humanbody"
xmlns:acme="http://www.acme.com/schema">
  <tt:VideoAnalytics>
    <tt:Frame UtcTime="2021-10-05T15:13:27.321" Source="MyClassifier">
      <tt:Transformation>
        <tt:Translate x="-1.0" y="-1.0"/>
        <tt:Scale x="0.003125" y="0.00416667"/>
      </tt:Transformation>
      <tt:Object ObjectId="15">
        <tt:Appearance>
          <tt:Shape>
            <tt:BoundingBox left="20.0" top="80.0" right="100.0" bottom="30.0"/>
            <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:Color>
            <tt:ColorCluster>
              <tt:Color X="213" Y="135" Z="126" />
              <tt:Weight>0.9</tt:Weight>
              <acme:ColorName>White</acme:ColorName>
            </tt:ColorCluster>
            <tt:ColorCluster>
              <tt:Color X="102" Y="176" Z="119" />
              <tt:Weight>0.5</tt:Weight>
              <acme:ColorName>Blue</acme:ColorName>
            </tt:ColorCluster>
          </tt:Color>
          <tt:Class>
            <tt:Type Likelihood="0.8">Human</tt:Type>
          </tt:Class>
          <tt:HumanFace>
            <fc:Gender>Male</fc:Gender>
            <fc:Age>
              <tt:Min>20</tt:Min>
              <tt:Max>30</tt:Max>
            </fc:Age>
            <fc:Accessory>
```

```
              <fc:Opticals>
                <fc:Wear>True</fc:Wear>
              </fc:Opticals>
              <fc:Hat>
                <fc:Wear>True</fc:Wear>
              </fc:Hat>
            </fc:Accessory>
          </tt:HumanFace>
          <tt:HumanBody>
            <bd:Clothing>
              <bd:Tops>
                <bd:Category>LongSleeve</bd:Category>
              </bd:Tops>
              <bd:Bottoms>
                <bd:Category>Shorts</bd:Category>
              </bd:Bottoms>
            </bd:Clothing>
            <bd:Belonging>
              <bd:Bag>
                <bd:Category>Backpack</bd:Category>
              </bd:Bag>
            </bd:Belonging>
          </tt:HumanBody>
        </tt:Appearance>
      </tt:Object>
    </tt:Frame>
  </tt:VideoAnalytics>
</tt:MetadataStream>
```

This data will be published under this topic:

```
MyDevice/onvif-mj/VideoAnalytics/1/MyClassifier
```

Where "MyDevice" is the given TopicPrefix and "1" is the profile token. Shown below is how the sample metadata frame is formatted in JSON:

```
{
  "Frame": [{
    "@UtcTime": "2021-10-05T15:13:27.321",
    "@Source": "MyClassifier",
    "@context": {
        "acme": "http://www.acme.com/schema"
 },
    "Transformation": {
      "Translate": {
        "@x": -1.0, "@y": -1.0
      },
      "Scale": {
        "@x": 0.003125, "@y": 0.00416667
      }
    },
    "Object": [{
      "@ObjectId": 15,
      "Appearance": {
        "Shape": {
          "BoundingBox": {
            "@left": 20.0, "@top": 80.0, "@right": 100.0, "@bottom": 30.0
          },
          "CenterOfGravity": {
            "@x": 60.0, "@y": 50.0
          },
          "Polygon": {
            "Point": [{
              "@x": 20.0, "@y": 30.0
```

```
              }, {
              "@x": 100.0, "@y": 30.0
              }, {
              "@x": 100.0, "@y": 80.0
              }, {
              "@x": 20.0, "@y": 80.0
            }]
          }
        },
        "Color": {
          "ColorCluster": [{
            "Color": {
              "@X": 213, "@Y": 135, "@Z": 126
            },
            "Weight": 0.9,
            "acme:ColorName": "White"
          }, {
            "Color": {
              "@X": 102, "@Y": 176, "@Z": 119
            },
            "Weight": 0.5,
            "acme:ColorName": "Blue"
          }]
        },
        "Class": {
          "Type": [{
            "@Likelihood": 0.8,
            "#text": "Human"
          }]
        },
        "HumanFace": {
          "Gender": "Male",
          "Age": {
            "Min": 20, "Max": 30
          },
          "Accessory": {
            "Opticals": {
              "Wear": true
            },
            "Hat": {
              "Wear": true
            }
          }
        },
        "HumanBody": {
          "Clothing": {
            "Tops": {
              "Category": "LongSleeve"
            },
            "Bottoms": {
              "Category": "Shorts"
            }
          },
          "Belonging": {
            "Bag": {
              "Category": "Backpack"
            }
          }
        }
      }
    }]
  }]
}
```

6 Service

This section covers the following main areas of this architecture:

- Analytics Module interface
- Rules interface

The analytics service allows fine-grained configuration of individual rules and individual analytics modules (see 6.2 and 6.3).

6.1 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 6.1.2. 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 6.1.1 and their options defined in 6.1.3.

6.1.1 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.

6.1.2 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[1]:

```
<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">
```

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

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

6.1.3 Configuration options

6.1.3.1 Overview

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

6.1.3.2 Option type definition

Table 13 defines how to describe any static or dynamic limitations for generic types through the GetRuleOptions and GetAnalyticsModule options.

Table 13: 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

Parameter Type	Option Type	Limitation
tt:Polyline	tt:IntRange	Number of points supported Min and Max value
tt:Polygon	tt: PolygonOptions	Specifies polygon support and number of points supported

Both tt:Polyline and tt:Polygon shall use normalized coordinates as defined in 5.2.2

Note that the option type in Table 13 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.

The service shall provide options only for parameters with constraints. Parameters like boolean can be excluded in options.

6.1.3.3 Occurrence

Configuration parameters may signal their allowed minimal occurrence using the minOccurs attribute. Setting the value to zero signals that the parameter is optional.

The maxOccurs attribute allows to specify how often an element may occur in the same instance of a rule or module.

6.1.3.4 Example

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

```
<tt: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>
</tt:AnalyticsModuleDescription>
```

- 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>
```

6.2 Rule interface

An XML structure is introduced in Section 6.2.1 to communicate the configuration of rules. Section 6.2.2 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 6.2.3 introduces operations to manage rules. If the device supports a Rule Engine, it shall implement the complete rule interface.

Analytics rules typically operate on scene description generated by analytics models as depicted in Figure 1. Note that changes to associated analytics modules may affect rule configuration and cause differing device responses for GetSupportedRules and GetRuleOptions.

6.2.1 Rule representation

The rule representation uses the configuration language as defined in section 6.1.1. 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="0.10"  y="0.50"/>
          <tt:Point x="1.0" y="0.50"/>
        </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="0.1"  y="0.5"/>
          <tt:Point x="1.0" y="0.5"/>
          <tt:Point x="1.0" y="0.7"/>
        </tt:Polygon>
      </tt:ElementItem>
    </tt:Parameters>
  </tt:Rule>
</tt:RuleEngineConfiguration>
```

6.2.2 Rule description language

Rules are described using the rule description language defined in section 6.1.2.

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.

6.2.3 Operations on rules

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

6.2.3.1 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 6.2.2. 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

6.2.3.2 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

6.2.3.3 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.

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

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:Sender - ter:Action - ter:FixedRules**
  At least one of the requested rules is fixed and cannot be created.
- **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

6.2.3.4 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 should 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

6.2.3.5 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

6.2.3.6 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

6.3 Analytics modules interface

Section 6.3.1 defines an XML structure that communicates the configuration of analytics modules. Section 6.3.2 defines the language that describes the configuration of a specific analytics module. Section 6.3.3 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.

6.3.1 Analytics module configuration

The analytics module configuration uses the configuration description language as described in section 6.1.1. 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>
```

6.3.2 Analytics module description language

The Analytics Module reuses the configuration description language, described in section 6.1.2. 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>
```

6.3.3 Operations on analytics modules

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

6.3.3.1 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 6.1.

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

6.3.3.2 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

6.3.3.3 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 and 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

6.3.3.4 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 should 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

6.3.3.5 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:NameNotExistent**
  The module name or names do not exist.
- **env:Sender - ter:Action - ter:FixedModules**
  At least one of the requested modules is fixed and cannot be deleted.
- **env:Receiver - ter:Action - ter:ConfigurationConflict**
  The device cannot delete the modules without creating a conflicting configuration.

ACCESS CLASS:

ACTUATE

6.3.3.6 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

6.3.3.7 GetSupportedMetadata

This operation allows to query what metadata a device can generate. It shall be supported if the Supported-Metadata 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 metadata 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

6.4 GetServiceCapabilities

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 6.2.
AnalyticsModuleSupport	Indication that the device supports the scene analytics module interface as specified in Section 6.3.
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

6.5 Events

6.5.1 Audio Detected (deprecated)

Analytics engine can send the following event for analyzed audio data. This event is deprecated in favor of events described in Annex H.

```
<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 (normative)

A.1 Overview

This Annex defines a set of generic rules. Rules are mainly designed for static cameras since the graphical primitives are provided using bounds based normalized coordinates. For usage with PTZ devices see A.1.3.

A.1.1 Message format

Whenever a rule triggers it generates a message according to the core specification message definition. The topic identifies the rule type while the source items identify the originating input and rule. Most events signal their status via data values.

TOPIC:

Topic of the rule related event.

SOURCE:

- **VideoSource [tt:ReferenceToken]**
  The token of the video source.
- **AnalyticsConfiguration - optional [tt:ReferenceToken]**
  Optional token of the analytics configuration.
- **Rule [xs:string]**
  Name of the Rule.

DATA:

Data items are rule specific, except for the generic elements described here. For property events any change of the data values items must be signaled with a changed event. Non-property events may have no data items at all.

- **ObjectId - optional [tt:StringList]**
  Optional list of objects triggering this rule.
- **ClassTypes - optional [tt:StringList]**
  Optional list of class types of the detected objects, one for each object, in the same order as object ids are listed.
- **Object - optional, unbounded [tt:Object]**
  Optional description of objects triggering this rule.

The data section contains the rule specific state or event information. Vendors are allowed to add vendor specific extensions to the data section, but must take care to do this in a future proof way by adding a vendor specific namespace prefix to the name.

```
<tt:Messages>
  <tt:Source>
    <tt:SimpleItemDescription Name="VideoSource" Type="tt:ReferenceToken"/>
    <tt:SimpleItemDescription Name="AnalyticsConfiguration" Type="tt:ReferenceToken"/>
    <tt:SimpleItemDescription Name="Rule" Type="xs:string"/>
  </tt:Source>
  <tt:Data>
    ...
    <tt:SimpleItemDescription Name="ObjectId" Type="tt:StringList"/>
    <tt:SimpleItemDescription Name="ClassTypes" Type="tt:StringList"/>
    <tt:ElementItemDescription Name="Object" Type="tt:Object"/>
    </tt:Data>
```

```
  <tt:ParentTopic>tns1:RuleEngine/Xxxx</tt:ParentTopic>
</tt:Messages>
```

Note that if the message includes Object information there is no need to include ObjectIds, because each Object includes its ID.

A.1.2 Parameters

Generally parameters are rule specific. Implementations may extend the ONVIF defined rules by providing additional parameters for e.g. object or other constraints. Note that these additional parameters are optional. Vendors should check the availability of ONVIF defined parameters like the class filter before defining own items.

A.1.2.1 Generic parameters

The following section is reserved for ONVIF defined generic parameters.

A device signals via GetSupportedRules which parameters may be applied to its various rules. For all parameters a device should also signal their valid ranges via GetRuleOptions whether a rule supports these.

In case of setting up any region based detection rule for event trigger (e.g. Field Detector, Loitering Detector, Declarative motion detector, Object Detection) rule should contain a parameter 'Field' ElementItem. The ElementItem identifies the region usually defined by a polygon for which the rule has been setup. The corresponding ElementItemDescription resembles the following:

```
<tt:ElementItemDescription Name="Field" Type="tt:Polygon">
```

If the rule supports activation and deactivation, the rule should contain a parameter 'Armed' SimpleItem. The corresponding SimpleItemDescription resembles the following:

```
<<tt:SimpleItemDescription Name="Armed" Type="xs:boolean">
```

A.1.2.2 Object Classification

Object classification can be utilized 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

The following example shows how to apply the ClassFilter to the line detector rule description.

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

```
</tt:RuleDescription>
```

Note that the order of parameters is arbitrary as long as the simple items are listed before the complex element items.

A.1.3 Moving cameras

PTZ devices supporting these kinds of rules should remap the graphical primitives based on the current view port resulting from pan, title and zoom values so that these can be used independently from the view port. Of course rules with graphical elements located fully outside of the current view port will not be able to trigger any events.

To realize this, the rules may include an element item called "PTZVector" that acts as a filter for activating the rule only when the view port matches the given values. The ElementItem identifies the position of the device for which the rule has been setup. The corresponding ElementItemDescription resembles the following:

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

As alternative the rules may include a "PresetToken" that acts in the same way as the "PTZVector". The corresponding SimpleItemDescription resembles the following:

```
<tt:SimpleItemDescription Name="PresetToken" Type="tt:ReferenceToken">
```

Note that tolerances for matching are outside of the scope of this specification. However devices should strive to tolerate and compensate for small deviations.

A.2 Line Detector

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 A.1, A is the first point and B is the second point.

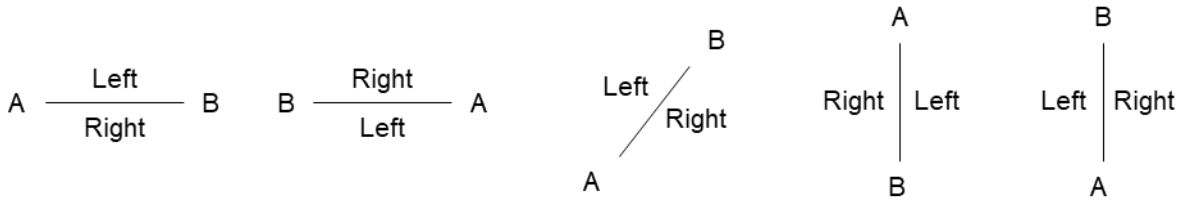


Figure A.1: Direction – Left and Right

PARAMETERS:

- **Direction - optional [tt:Direction]**
  Optional restriction of direction sensitivity. Default is any.
- **Segments [tt:Polyline]**
  Graphical line definition in normalized coordinates.

TOPIC:

- **tns1:RuleEngine/LineDetector/Crossed**

SOURCE:

See A.1.1

DATA:

See A.1.1

The following snippet shows the rule definition:

```
<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:Source>
    <tt:Data>
      ...
    </tt:Data>
    <tt:ParentTopic>tns1:RuleEngine/LineDetector/Crossed</tt:ParentTopic>
  </tt:Messages>
</tt:RuleDescription>
```

A.3 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.

PARAMETERS:

- **Field [tt:Polygon]**
  Graphical field definition in normalized coordinates.

TOPIC:

- **tns1:RuleEngine/FieldDetector/ObjectsInside**

SOURCE:

See A.1.1

DATA:

See A.1.1

- **IsInside [xs:boolean]**
  True if one or more objects matching the parameters are inside.

Note: a device signaling ObjectId shall emit changed events whenever the list of objects changes.

See below example for a FieldDetector definition:

```
<tt:RuleDescription Name="tt:FieldDetector">
  <tt:Parameters>
    <tt:ElementItemDescription Name="Field" Type="tt:Polygon"/>
  </tt:Parameters>
  <tt:Messages IsProperty="true">
    <tt:Source>
      ...
    </tt:Source>
    <tt:Data>
      ...
      <tt:SimpleItemDescription Name="IsInside" Type="xs:boolean"/>
    </tt:Data >
    <tt:ParentTopic>
      tns1:RuleEngine/FieldDetector/ObjectsInside
```

```
    </tt:ParentTopic>
  </tt:Messages>
</tt:RuleDescription>
```

A.4 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.

PARAMETERS:

- **Field - optional [tt:Polygon]**
  Optional graphical field restricting the detection area.
- **TimeThreshold - [xs:duration]**
  Maximum duration before an object is judged to be loitering.

TOPIC:

- **tns1:RuleEngine/LoiteringDetector/ObjectIsLoitering**

SOURCE:

See A.1.1

DATA:

See A.1.1

- **Since [xs:dateTime]**
  Since when this object is loitering.

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

```
<tt:RuleDescription Name="tt:LoiteringDetector">
  <tt:Parameters>
    <tt:SimpleItemDescription  Name="TimeThreshold" Type="xs:duration"/>
    <tt:ElementItemDescription Name="Field" Type="tt:Polygon"/>
  </tt:Parameters>
  <tt:Messages IsProperty="true">
    <tt:Source>
      ...
    </tt:Source>
    <tt:Data>
      <tt:SimpleItemDescription Name="ObjectId" Type="tt:StringList"/>
      <tt:SimpleItemDescription Name="Since" Type="xs:dateTime"/>
    </tt:Data >
    <tt:ParentTopic>
      tns1:RuleEngine/LoiteringDetector/ObjectIsLoitering
    </tt:ParentTopic>
  </tt:Messages>
</tt:RuleDescription>
```

A.5 Line crossing counting rule

This rule counts the number of objects crossing a set of line segments. Optionally the line segments may only trigger counting when passed in a specific direction. The configuration parameters also include the time interval to report the events and time interval to reset its counter.

PARAMETERS:

- **Segments - unbounded [tt:Polyline]**
  One or more polylines used for detection.

- **PassAllPolylines - optional [xs:boolean]**
  Signals that an object must pass all polylines before being counted. By default any polyline passing counts separately.
- **Direction - optional [tt:Direction]**
  Optional restriction of direction sensitivity. Default is any. See definition in Figure A.1
- **ReportTimeInterval - optional [xs:duration]**
  Optional time interval to reduce number of reported changes by reporting aggregated values.
- **ResetTime - optional, unbounded [xs:time]**
  Time or times of the day when the counter should be reset. The time value shall be interpreted as localtime.

TOPIC:

- **tns1:RuleEngine/CountAggregation/Counter**

SOURCE:

See A.1.1

DATA:

- **Count [xs:int]**
  Number of objects counted since last reset.

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

```
<tt:RuleDescription Name="tt:LineCounting">
  <tt:Parameters>
    <tt:SimpleItemDescription Name="ReportTimeInterval" Type="xs:duration"/>
    <tt:SimpleItemDescription Name="ResetTime" Type="xs:time"/>
    <tt:SimpleItemDescription Name="Direction" Type="tt:Direction"/>
    <tt:SimpleItemDescription Name="PassAllPolylines" Type="xs:boolean"/>
    <tt:ElementItemDescription Name="Segments" Type="tt:Polyline"/>
  </tt:Parameters>
  <tt:Messages IsProperty="true">
    <tt:Source>
      ...
    </tt:Source>
    <tt:Data>
      <tt:SimpleItemDescription Name="Count" Type="xs:int"/>
    </tt:Data >
    <tt:ParentTopic>
      tns1:RuleEngine/CountAggregation/Counter
    </tt:ParentTopic>
  </tt:Messages>
</tt:RuleDescription>
```

A.6 Occupancy Counting rule

The occupancy counting rule counts the number of objects inside an enclosure by counting objects passing in and out of a set of line segments (barriers). The counter increments when objects pass from left to right and decrements when objects pass from right to left.

The configuration parameters also include the time interval to report the events and time interval to reset its counter.

PARAMETERS:

- **Segments - unbounded [tt:Polyline]**
  One or more polylines used for detection.

- **ReportTimeInterval - optional [xs:duration]**
  Optional time interval to reduce number of reported changes by reporting aggregated values.
- **ResetTime - optional, unbounded [xs:time]**
  Time or times of the day when the counter should be reset. The time value shall be interpreted as localtime.

TOPIC:

- **tns1:RuleEngine/CountAggregation/OccupancyCounter**

SOURCE:

See A.1.1

DATA:

- **Count [xs:int]**
  Number of objects counted since last reset.

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

```
<tt:RuleDescription Name="tt:OccupancyCounting">
  <tt:Parameters>
    <tt:ElementItemDescription Name="Segments" Type="tt:Polyline"/>
    <tt:SimpleItemDescription Name="ReportTimeInterval" Type="xs:duration"/>
    <tt:SimpleItemDescription Name="ResetTime" Type="xs:time"/>
  </tt:Parameters>
  <tt:Messages IsProperty="true">
    <tt:Source>
      ...
    </tt:Source>
    <tt:Data>
      <tt:SimpleItemDescription Name="Count" Type="xs:int"/>
    </tt:Data >
    <tt:ParentTopic>
      tns1:RuleEngine/CountAggregation/OccupancyCounter
    </tt:ParentTopic>
  </tt:Messages>
</tt:RuleDescription>
```

A.7 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.

PARAMETERS:

- **ClassFilter [tt:StringList]**
  List of classes to be detected.
- **ConfidenceLevel - optional [xs:float]**
  Minimum confidence level for accepting a classification.
- **DwellTime - optional [xs:duration]**
  Duration an object must be visible before accepting a classification.

TOPIC:

- **tns1:RuleEngine/ObjectDetection/Object**

SOURCE:

See A.1.1

DATA:

- **ClassTypes [tt:StringList]**
  Matching class associations of the detected objects.

Description of an object detector defined by the following code using the rule description language:

```
<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: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 6.1.3.2.

```
<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.8 Object Abandoned

The Object Abandoned rule is defined by a simple non-intersecting polygon as an area of interest and abandoned time threshold interval. The Object Abandoned detector continuously monitors an area to detect objects or items that have been abandoned in the scene beyond a certain threshold time. The Object Abandoned module looks for objects that are not part of the "normal" scene and issues real-time alerts upon detecting exceptions.

Object Abandoned can be deployed at airports, train stations, national infrastructure facilities, public places, and other secure areas to detect unattended baggage and suspicious looking objects left in the scene.

PARAMETERS:

- **Field [tt:Polygon]**
  Graphical field definition in normalized coordinates.
- **TimeThreshold [xs:duration]**
  Minimum duration the object should stay in the scene before it is judged to be abandoned.

- **Sensitivity [xs:float]**
  This parameter allows to change the ratio between True and False detections. For "High" sensitivity value there are more True positive events as well as False positives.

TOPIC:

- **tns1:RuleEngine/ObjectAbandoned**

SOURCE:

See A.1.1

DATA:

- **ObjectId [tt:StringList]**
  List of objects triggering this rule.

See below example for a Object Abandoned definition:

```
<tt:RuleDescription Name="tt:ObjectAbandoned">
  <tt:Parameters>
    <tt:ElementItemDescription Name="Field" Type="tt:Polygon"/>
    <tt:SimpleItemDescription Name="TimeThreshold" Type="xs:duration"/>
    <tt:SimpleItemDescription Name="Sensitivity" Type="xs:float"/>
  </tt:Parameters>
  <tt:Messages IsProperty="true">
    <tt:Source>
      ...
    </tt:Source>
    <tt:Data>
      <tt:SimpleItemDescription Name="ObjectId" Type="tt:StringList"/>
    </tt:Data >
    <tt:ParentTopic>
      tns1:RuleEngine/ObjectAbandoned
    </tt:ParentTopic>
  </tt:Messages>
</tt:RuleDescription>
```

A.9 Object Removed

The Object Removed rule is defined by a simple non-intersecting polygon as an area of interest and removed time threshold interval. Object Removed module continuously monitors a scene and detects and alarms when any valuable property is removed from the scene.

For example, object removed rule can be used in loss prevention and securing valuable items in government, industrial, retail, corporate facilities, and public places such as museums and national landmarks.

PARAMETERS:

- **Field [tt:Polygon]**
  Graphical field definition in normalized coordinates.
- **TimeThreshold [xs:duration]**
  Minimum duration the object should be out of the scene before it is judged to be removed.
- **Sensitivity [xs:float]**
  This parameter allows to change the ratio between True and False detections. For "High" sensitivity value there are more True positive events as well as False positives.

TOPIC:

- **tns1:RuleEngine/ObjectRemoved**

SOURCE:

See A.1.1

DATA:

- **ObjectId [tt:StringList]**
  List of objects triggering this rule.

See below example for a Object Removed definition:

```
<tt:RuleDescription Name="tt:ObjectRemoved">
  <tt:Parameters>
    <tt:ElementItemDescription Name="Field" Type="tt:Polygon"/>
    <tt:SimpleItemDescription Name="TimeThreshold" Type="xs:duration"/>
    <tt:SimpleItemDescription Name="Sensitivity" Type="xs:float"/>
  </tt:Parameters>
  <tt:Messages IsProperty="true">
    <tt:Source>
      ...
    </tt:Source>
    <tt:Data>
      <tt:SimpleItemDescription Name="ObjectId" Type="tt:StringList"/>
    </tt:Data >
    <tt:ParentTopic>
      tns1:RuleEngine/ObjectRemoved
    </tt:ParentTopic>
  </tt:Messages>
</tt:RuleDescription>
```

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.

Parameter Name	Description
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. 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). 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 upperleft -hand corner and the last cell in the lower right-hand corner. The parameter "Transformation" maps the frame coordinate system (see5.2.2) 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. Range is 0 to 100
Layout	The layout is typically defined by the hardware of the motion detector. 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. 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

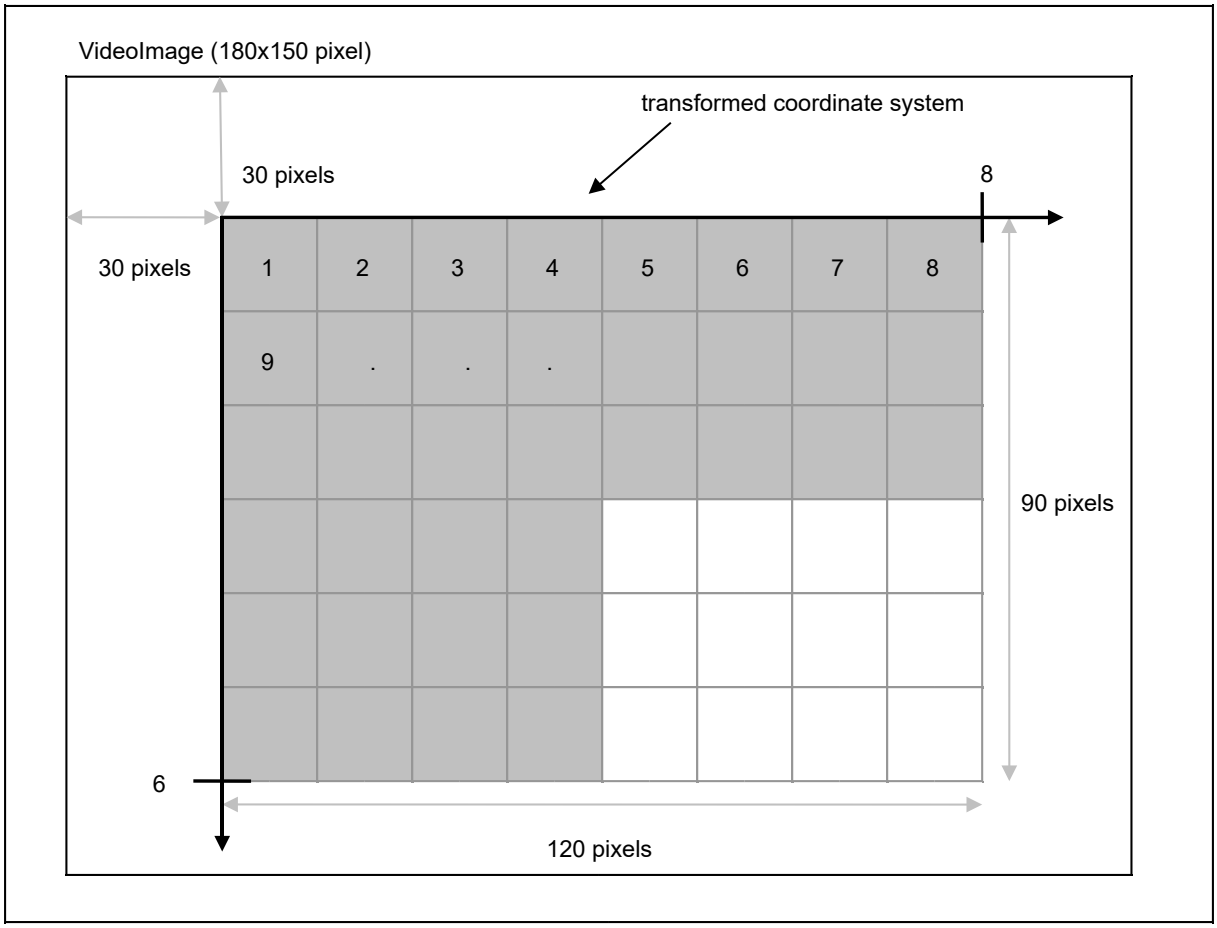


Figure B.1: CellLayout of an 8×6 CellMotionEngine

```
<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.

Parameter Name	Description
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.

A **Temperature Measurement Module** provides a generic configuration for Spot, Box and BodyTemperature measurement using the generic description language defined in section 5.3.12

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 three very simple rules, the **RadiometryTemperatureRule**, a generic measurement rule **TemperatureMeasurementRule** and the **BodyTemperatureRule**. All these three rules are based on a single Measurement Analytics Module output. 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.

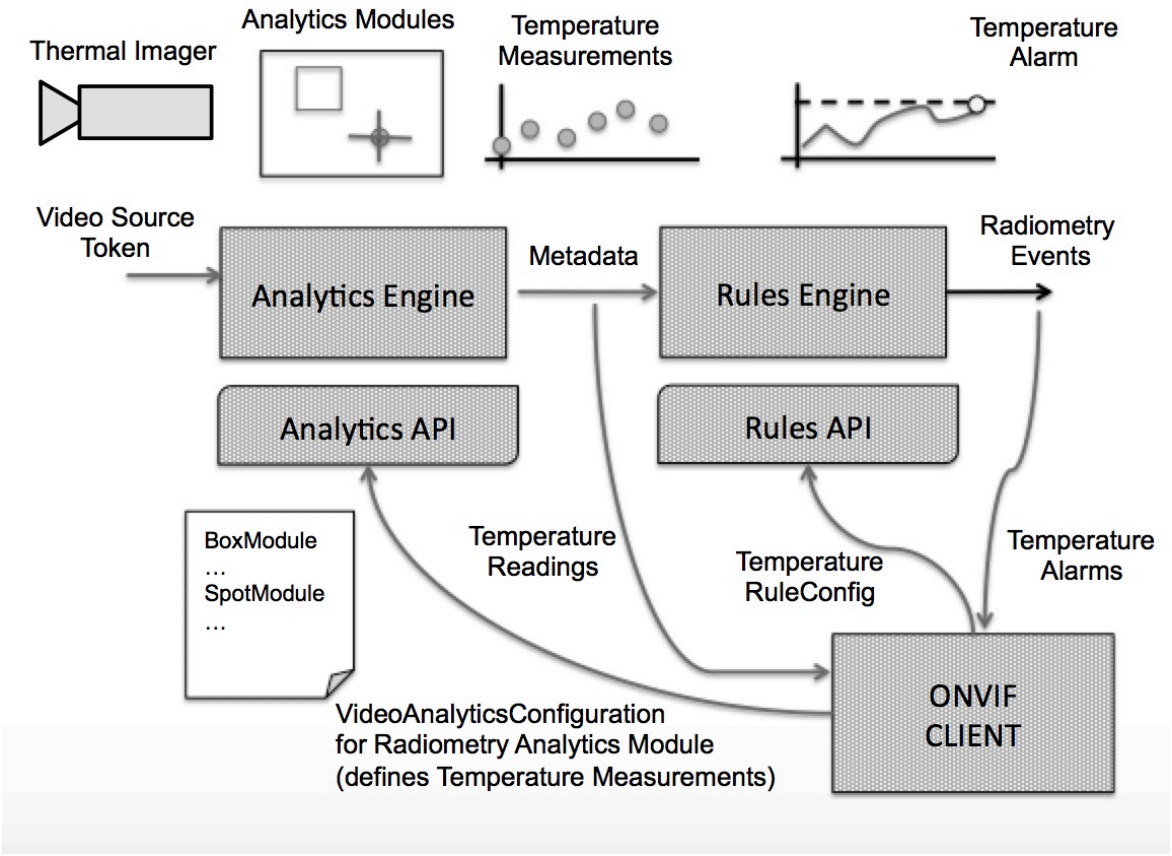


Figure D.1: Radiometry Analytics Modules, Rules, Metadata and Events

The current specification covers three different types of Analytics Modules for Radiometry: Spot, Box and Generic temperature measurement module.

All three 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: • ReflectedAmbientTemperature • Emissivity • DistanceToObject • RelativeHumidity • AtmosphericTemperature • AtmosphericTransmittance • ExtOpticsTemperature • 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>
    <tt:Data>
      <tt:SimpleItemDescription Name="TimeStamp" Type="xs:dateTime"/>
      <tt:ElementItemDescription Name="Reading" Type="ttr:SpotTemperatureReading"/>
    </tt: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: 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: ItemID

Parameter Name	Description
	SpotTemperature: Actual value of the measured Temperature. 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>
    <tt:Data>
      <tt:SimpleItemDescription Name="TimeStamp" Type="xs:dateTime"/>
      <tt:ElementItemDescription Name="Reading" Type="ttr:BoxTemperatureReading"/>
    </tt: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 Supported Temperature Values: MaxTemperature, MinTemperature, AverageTemperature, MedianTemperature. RadiometryParameters: Not present means Global Parameters from Thermal Service are being used.

D.1.3 Temperature Measurement Module

Generic Temperature measurement module defined below can be used to configure any of the following: spot, box or body temperature. For new implementations it is recommended to use this configuration, to utilize the generic description language.

For options this module refers to the generic description language defined in 5.3.12

```
<tt:AnalyticsModuleDescription Name="tt:TemperatureMeasurement">
<tt:Parameters>
  <tt:SimpleItemDescription Name="Enable" Type="xs:boolean"/>
  <tt:SimpleItemDescription Name="Mode" Type="xs:string"/>
  <tt:SimpleItemDescription Name="ReflectedAmbientTemperature" Type="xs:float"/>
  <tt:SimpleItemDescription name="Emissivity" type="xs:float"/>
  <tt:SimpleItemDescription name="DistanceToObject" type="xs:float"/>
  <tt:SimpleItemDescription name="RelativeHumidity" type="xs:float"/>
  <tt:SimpleItemDescription name="AtmosphericTemperature" type="xs:float"/>
  <tt:SimpleItemDescription name="AtmosphericTransmittance" type="xs:float"/>
  <tt:SimpleItemDescription name="ExtOpticsTemperature" type="xs:float"/>
  <tt:SimpleItemDescription name="ExtOpticsTransmittance" type="xs:float"/>
  <tt:SimpleItemDescription name="PresetToken" type="xs:string"/>
  <tt:ElementItemDescription Name="MeasurementPoint" Type="tt:Rectangle"/>
  <tt:ElementItemDescription Name="MeasurementRegion" Type="tt:Polygon"/>
</tt:Parameters>
```

```
 <tt:MessageDescription>
 <tt:Source>
   <tt:SimpleItemDescription Name="VideoSource" Type="tt:ReferenceToken"/>
   <tt:SimpleItemDescription Name="AnalyticsModuleName" Type="xs:string"/>
 </tt:Source>
 <tt:Data>
   <tt:SimpleItemDescription Name="SpotReading" Type="xs:float"/>
   <tt:SimpleItemDescription Name="MinTemperature" Type="xs:float"/>
   <tt:SimpleItemDescription Name="MaxTemperature" Type="xs:float"/>
   <tt:SimpleItemDescription Name="AverageTemperature" Type="xs:float"/>
   <tt:SimpleItemDescription Name="MedianTemperature" Type="xs:float"/>
 </tt:Data>
 <tt:ParentTopic>
   tns1:VideoAnalytics/Radiometry/TemperatureReading
 </tt:ParentTopic>
 </tt:MessageDescription>
</tt:AnalyticsModuleDescription>
```

Temperature measurement module has following parametersTable D.6 and TemperatureReading Event has following parameters Table D.7 .

Table D.6: Temperature measurement analytics module fields

Field	Description	Requirement Level
Enable	To enable or disable the measurement module	Mandatory
Mode	Operating mode, eg., it can either operate as Spot, Box or BodyTemperature modes, actual supported modes would be listed in options	Mandatory
Radiometry parameters: ReflectedAmbientTemperature, Emissivity,DistanceToObject, RelativeHumidity, AtmosphericTemperature, AtmosphericTransmittance, ExtOpticsTemperature, ExtOpticsTransmittance	Radiometry parameters, if this is not supported global radiometry parameters are applicable	Optional
PresetToken	PTZ Preset position associated with spot, box or bodytemperature measurement location. If no PresetToken is present measurement location behaves as screen element, and stay on the same screen coordinates as the PTZ moves (like a head up display mask). If PresetToken is present the measurement location shall be active only when its completely contained in the Field of View	Optional
MeasurementPoint	Screen coordinates where temperature needs to be measured. Applicable when the mode is Spot temperature measurement.	Optional
MeasurementRegion	Polygon region where tempeature has to be measured. Applicable when the mode is box or body temperature measurement	Optional

Table D.7: Description of the TemperatureReading meta event fields

Parameter Name	Description
SpotReading	Provides the temperature of the configured screen coordinates in Kelvin, applicable only when the operating mode is set to Spot.
MinTemperature	Provides the minimum temperature within the configured region in Kelvin, applicable only when the operating mode is set to Box.
MaxTemperature	Provides the maximum temperature within the configured region in Kelvin, applicable only when the operating mode is set to Box.
AverageTemperature	Provides the average temperature within the configured region in Kelvin, applicable only when the operating mode is set to Box.
MedianTemperature	Provides the median temperature within the configured region in Kelvin, applicable only when the operating mode is set to Box.

D.2 Radiometry Rules

Following radiometry rules are defined, RadiometryTemperatureRule and TemperatureMeasurementRule.

D.2.1 RadiometryTemperatureRule

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.8: Radiometry Rule configuration options

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

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:RadiometryTemperatureRuleC
  </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.9: 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.10: 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)

D.2.2 TemperatureMeasurementRule

TemperatureMeasurementRule is a basic rule that is applicable for both Spot and Box Temperature rules. This rule works on the output of Temperature Measurement Module (D.1.3), when the mode is configured as Spot or Box.

TemperatureMeasurementRule and options follows the generic description language defined in 5.3.12. For description of parameters refer Table D.11.

```
<xs:RuleDescription Name="ttr:TemperatureMeasurementRule">
```

```
  <xs:Parameters>
 <tt:SimpleItemDescription Name="TemperatureType" Type="xs:string"/>
 <tt:SimpleItemDescription Name="RuleCondition" Type="xs:string"/>
 <tt:SimpleItemDescription Name="ThresholdTemperature" Type="xs:float"/>
 <tt:SimpleItemDescription Name="HysteresisTemperature" Type="xs:float"/>
 <tt:SimpleItemDescription Name="Duration" Type="xs:duration"/>
 <tt:SimpleItemDescription Name="TemperatureMeasurementModule" Type="xs:string"/>
  </xs:Parameters>
  <xs:MessageDescription IsProperty="true">
    <xs:Source>
      <xs:SimpleItemDescription Name="VideoSourceToken" Type="tt:ReferenceToken"/>
      <xs:SimpleItemDescription Name="RuleName" Type="xs:string"/>
    </xs:Source>
    <xs:Data>
      <xs:SimpleItemDescription Name="AlarmActive" Type="xs:boolean"/>
      <xs:ElementItemDescription Name="AlarmGeoLocation" Type="tt:GeoLocation" minOccurs=0/>
    </xs:Data>
    <xs:ParentTopic>
      tns1:RuleEngine/Radiometry/TemperatureMeasurementAlarm
    </xs:ParentTopic>
  </xs:MessageDescription>
</xs:RuleDescription>
```

Table D.11: TemperatureMeasurementRule configuration parameters

Parameter Name	Description
TemperatureType	Indicates which of the temperature values provided by the input Analytics Module shall be used by the rule. In the case of Analytics Modules providing a single Temperature Value (e.g. Spot) this parameter is ignored, and is therefore optional.
RuleCondition	Configures the condition to check eg, LessThan, EqualTo etc., Actual supported values would be provided in options.
ThresholdTemperature	Indicates the temperature reference value the rule shall be checked against. In Kelvin.
HysteresisTemperature	Indicates the width in Kelvin of the temerature hysteresis band to be considered by the rule.
Duration	How long the configured condition should stay in same state to generate the event.
TemperatureMeasurementModule	Name of the analytics module providing the temperature on which rule is defined.

Table D.12: Description of the TemperatureMeasurementAlarm event fields

Parameter Name	Description
AlarmActive	True (temperature condition is met) or False (condition not met)
AlarmGeoLocation	Geo Location of the measurement triggering the Alarm (if available)

D.2.3 BodyTemperatureRule

BodyTemperatureRule rule works on the output of Temperature Measurement Module (D.1.3), when the mode is configured as BodyTemperature.

BodyTemperatureRule and options follows the generic description language defined in 5.3.12. For description of parameters refer Table D.13.

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

```
<tt:Parameters>
  <tt:SimpleItemDescription Name="ThresholdTemperature" Type="xs:float"/>
  <tt:SimpleItemDescription Name="RuleCondition" Type="xs:string"/>
  <tt:SimpleItemDescription Name="Duration" Type="xs:duration"/>
  <tt:SimpleItemDescription Name="TemperatureMeasurementModule" Type="xs:string"/>
</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="xs:boolean" Name="State"/>
    <tt:SimpleItemDescription Type="xs:integer" Name="ObjectId"/>
    <tt:SimpleItemDescription Type="xs:float" Name="Temperature"/>
    <tt:SimpleItemDescription Type="xs:anyURI" Name="ImageUri"/>
    <tt:SimpleItemDescription Type="xs:anyURI" Name="SceneImageUri"/>
    <tt:ElementItemDescription Type="xs:base64Binary" Name="Image"/>
    <tt:ElementItemDescription Type="xs:base64Binary" Name="SceneImage"/>
    <tt:ElementItemDescription Type="tt:Rectangle" Name="BoundingBox"/>
  </tt:Data>
  <tt:ParentTopic>tns1:RuleEngine/Radiometry/BodyTemperatureAlarm</tt:ParentTopic>
</tt:Messages>
```

Table D.13: BodyTemperatureRule configuration parameters

Parameter Name	Description
ThresholdTemperature	Indicates the temperature reference value the rule shall be checked against. In Kelvin.
RuleCondition	Configures the condition to check eg, LessThan, EqualTo etc., Actual supported values would be provided in options.
Duration	How long the configured condition should stay in same state to generate the event.
TemperatureMeasurementModule	Name of the analytics module providing the temperature on which rule is defined.

Table D.14: Description of the BodyTemperature event fields

Parameter Name	Description
State	True (temperature condition is met) or False (condition not met)
ObjectId	Optional Object ID of the person, whose temperature met the condition
Temperature	Optional temperatue of the object in Kelvin
ImageUri	Optional , uri of the object image
SceneImageUri	Optional , uri of the full scene image
Image	Optional , base64encoded object image
SceneImage	Optional , base64encoded scene image
BoundingBox	Optional , bounding box information of the object as normalized screen coordinates (-1.0, 1.0)

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 5.3.12. 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: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/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 6.1.3.2.

```
<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

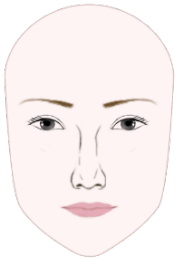
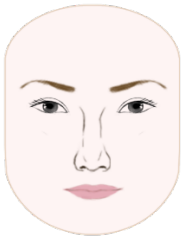
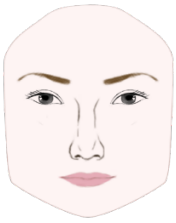
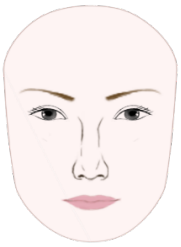
Feature	Value	Sample
FacialShape	Long	
	Round	
	Square	
	Oval	

Table F.2: Samples For HairStyle

Feature	Value	Sample
HairStyle	Straight	
	Wave	
	Curly	
	CrewCut	
	Bald	

Feature	Value	Sample
	Ponytail	
	Pigtail	

Table F.3: Samples For EyebrowSpace

Feature	Value	Sample
EyebrowSpace	Joint	
	Separate	

Table F.4: Samples For EyeShape

Feature	Value	Sample
EyeShape	Almond	
	Round	

Table F.5: Samples For Ear

Feature	Value	Sample
Ear	Round	
	Pointed	
	Narrow	
	BroadLobe	

Table F.6: Samples For NoseEnd

Feature	Value	Sample
NoseEnd	Snub	
	Turnedup	
	Flat	

Feature	Value	Sample
	Hooked	

Table F.7: Samples For FacialHair

Feature	Value	Sample
FacialHair	Mustache	
	Beard	
	Sideburn	

Table F.8: Samples For Chin

Feature	Value	Sample
Chin	Double	

Feature	Value	Sample
	Pointed	
	Round	

Table F.9: Samples For Accessory

Feature	Value	Sample
Accessory	Opticals	
	Hat	
	Helmet	
	Mask	

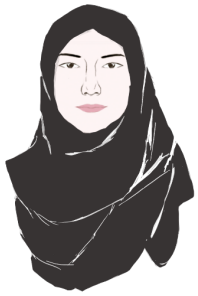
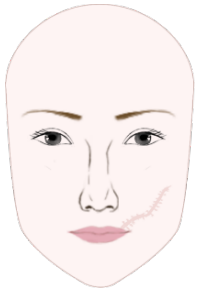
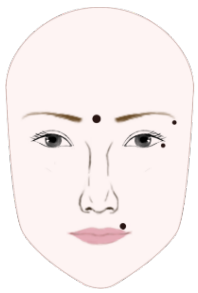
Feature	Value	Sample
	Hijab	
	RightEyePatch	
	LeftEyePatch	

Table F.10: Samples For AdditionalFeatures

Feature	Value	Sample
AdditionalFeatures	Scar	
	Mole	

Feature	Value	Sample
	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.

For the definition of the message source refer to A.1.1.

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. • Embedded: Embed a recognized image into the event payload. • LocalStorage: Include a local URI for the recognized image. • RemoteStorage: Include a remote URI for the recognized image.	Optional
Threshold	The level of confidence required for the detection/recognition.	Optional

Options for the rule parameters are described by the generic mechanism defined in section 6.1.3.

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. Area of recognized object in normalized screen coordinates (-1.0, 1.0).	Optional
Label	Information regarding matching list. Predefined values are: • Whitelisted • Blacklisted • 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=”VideoSource” Type=”tt:ReferenceToken”/>
      <tt:SimpleItemDescription Name=”Rule” 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 G.2 as well as the specific event fields listed in Table G.3 .

Table G.3: 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: SimpleItemDescription Name=”Country” Type=”xs:string”/>
 <tt: SimpleItemDescription Name=”Sensitivity” Type=”xs:float”/>
 <tt: SimpleItemDescription Name=”Threshold” Type=”xs:float”/>
 <tt: SimpleItemDescription Name=”PlateMinWidth” Type=”xs:float”/>
 <tt: SimpleItemDescription Name=”PlateMaxWidth” Type=”xs:float”/>
 <tt: SimpleItemDescription Name=”EventInterval” Type=”xs:int”/>
  </tt:Parameters>
  <tt:Messages>
    <tt:Source>
      <tt:SimpleItemDescription Name=”VideoSource” Type=”tt:ReferenceToken”/>
      <tt:SimpleItemDescription Name=”Rule” Type=”xs:string”/>
    </tt:Source>
    <tt:Data>
      <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 G.1 as well as the specific configuration parameters listed in Table G.4.

Table G.4: 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
Country	Specifies the country for which LPR should be processed.	Optional
Sensitivity	Detection sensitivity which provides compromise between detection rate and false alarms.Higher the sensitivity, more License plates are detected at the cost of increasing false alarms number.	Optional
PlateMinWidth	Minimum width of the license plate in percentage to be detected/recognized with respect to Scene Image.	Optional
PlateMaxWidth	Maximum width of the license plate in percentage to be detected/recognized with respect to Scene Image.	Optional
EventInterval	Minimum interval between event notification when same license plate is recognized.	Optional

Options for the rule parameters are described by the generic mechanism defined in section 6.1.3.

The License Plate Recognition event consists of the generic event fields from Table G.2 as well as the specific event fields listed in Table G.5.

Table G.5: 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.
Audio analytics (normative)

H.1 Audio Classification

A rule engine can send the following event for detected audio classes.

```
<tt:RuleDescription Name="tt:AudioClassDetector">
  <tt:Parameters>
    <tt:SimpleItemDescription Name="ClassFilter" Type="tt:StringList"/>
    <tt:SimpleItemDescription Name="ConfidenceLevel" Type="xs:float"/>
  </tt:Parameters>
  <tt:Messages>
    <tt:MessageDescription IsProperty="false">
      <tt:Source>
        <tt:SimpleItemDescription Name="AudioSource" Type="tt:ReferenceToken"/>
        <tt:SimpleItemDescription Name="AnalyticsConfiguration" Type="tt:ReferenceToken"/>
        <tt:SimpleItemDescription Name="Rule" Type="xs:string"/>
      </tt:Source>
      <tt:Data>
        <tt:SimpleItemDescription Name="Class" Type="tt:StringList"/>
        <tt:SimpleItemDescription Name="Score" Type="tt:FloatList"/>
        <tt:SimpleItemDescription Name="Loudness" Type="xs:float"/>
      </tt:Data >
      <tt:ParentTopic>tns1:RuleEngine/AudioDetector/Class</tt:ParentTopic>
    </tt:MessageDescription>
  <tt:Messages>
</tt:RuleDescription>
```

PARAMETERS:

- **ClassFilter - optional [tt:StringList]**
  List of classes that the rule should operate on chosen from tt:AudioClassification.
- **ConfidenceLevel - optional [xs:float]**
  Minimum confidence level for accepting a classification.

TOPIC:

tns1:RuleEngine/AudioDetector/Class

SOURCE:

- **AudioSource [tt:ReferenceToken]**
  The token of the audio source.
- **AnalyticsConfiguration - optional [tt:ReferenceToken]**
  Optional token of the analytics configuration this rule belongs to.
- **Rule - optional [xs:string]**
  Optional name of the Rule that generated the event.

DATA:

- **Class [tt:StringList]**
  List of detected audio classifications of type tt:AudioClassification.
- **Score - optional [tt:FloatList]**
  Optional list of scores of detected audio classifications.
- **Loudness - optional [xs:float]**
  Optional loudness of the detected sound in dB.

Annex I.
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, Michio Hirai	Added A6 Query Rule Change Request 1559
16.06	Jun-2016	Hiroyuki Sano	Change Request 1821, 1825, 1844
16.12	Dec-2016	Steven Dillingham, Hiroyuki Sano	Added Motion Region Detection Rules: 5.2.3.6, Annex C Change Request 1871
17.06	Jun-2017	Hans Busch, Hiroyuki Sano	Updated method layouts. Apply 2089 Change Request 1843, 1984, 2042, 2052, 2072, 2114, 2137 Added GetAnalyticsModuleOptions & Annex D
17.12	Dec-2017	Hiroyuki Sano	Change Request 2177, 2204
18.06	Jun-2018	Qian Zhou, Hiroyuki Sano	Added Vehicle information, Speed, License plate information descriptor 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, Hiroyuki Sano, Hans Busch	Added PTZ Motion Region Configuration, Tampering Configuration Change Request 2462, 2484 Change Request 2465
19.12	Dec-2019	Haina Han, Hiroyuki Sano	Added Image Sending, Object Classification, Face and Human body descriptor Change Request 2504, 2527, 2528, 2555, 2556, 2565, 2566, 2567, 2578, 2628
20.06	Jun-2020	Michio Hirai, Sujith Raman, Hainha Han, Sriram Bhetanabottla	Change Request 2695, 2640, 2641, 2669, 2671, 2678, 2682, 2685, 2690, 2696, 2698, 2686 Added PolyLine and Polygon options Added Recognition rule engines
20.12	Dec-2020	Hans Busch	Add occurrence section. Refactor Annex A. Remove Query Rule and Declaritive Motion Detector.
21.06	Jun-2021	Hans Busch	Update counting rule. Minor editorial fixes.
21.12	Dec-2021	Sujith Raman, Fredrik Svensson, Hans Busch	Added body temperature alarm rule. Clarifications for color.
22.06	Jun-2022	Fredrik Svensson, Hans Busch, Sriram	Move scene description to separate chapter and append section on JSON syntax. Clarify color value range. Add missing Access Class for DeleteAnalyticsModule. Update license plate rule configuration.

Rev.	Date	Editor	Changes
		Bhetanabottla, Venki Aravapalli	
22.12	Dec-2022	Fredrik Svensson, Hans Busch	Unify analytics and core mqtt attribute and value handling. Add Likelihood attribute to color type
23.06	Jun-2023	Fredrik Svensson, Sujith Raman, Venki Aravapalli	Addition of spherical coordinate descriptor, direction descriptor and barcode information. Rule additions for abondoned and removed objects.
23.12	Dec-2023	Sriram Bhetanabottla, Hans Busch, Sujith Raman	JSON Payload section moved to Core spec, fix EyeShape definition, clarify metadata image format and add optional event object description.