ONVIF™
Video Analytics Service Specification

Version 2.1
June, 2011

CONTENTS

1 Scope

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

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

2 Normative references

ONVIF Core Specification
<http://www.onvif.org/specs/core/ONVIF-Core-Spec-v210.pdf>
ONVIF Media Service Specification
<http://www.onvif.org/specs/srv/media/ONVIF-Media-Service-Spec-v210.pdf>
ONVIF Streaming Specification
<http://www.onvif.org/specs/stream/ONVIF-Streaming-Spec-v210.pdf>

3 Terms and Definitions

3.1 Definitions

Video Analytics	Algorithms used to evaluate video data for meaning of content

3.2 Abbreviations

PTZ	Pan Tilt Zoom

4 Overview

Video analytic applications are divided into image 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 analytics engine and as the rule engine, together with the events and actions, form the video analytics architecture according to this specification as illustrated in Figure 1.

The video analytics architecture consists of elements and interfaces. Each element provides a functionality corresponding to a semantically unique entity of the complete video 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.

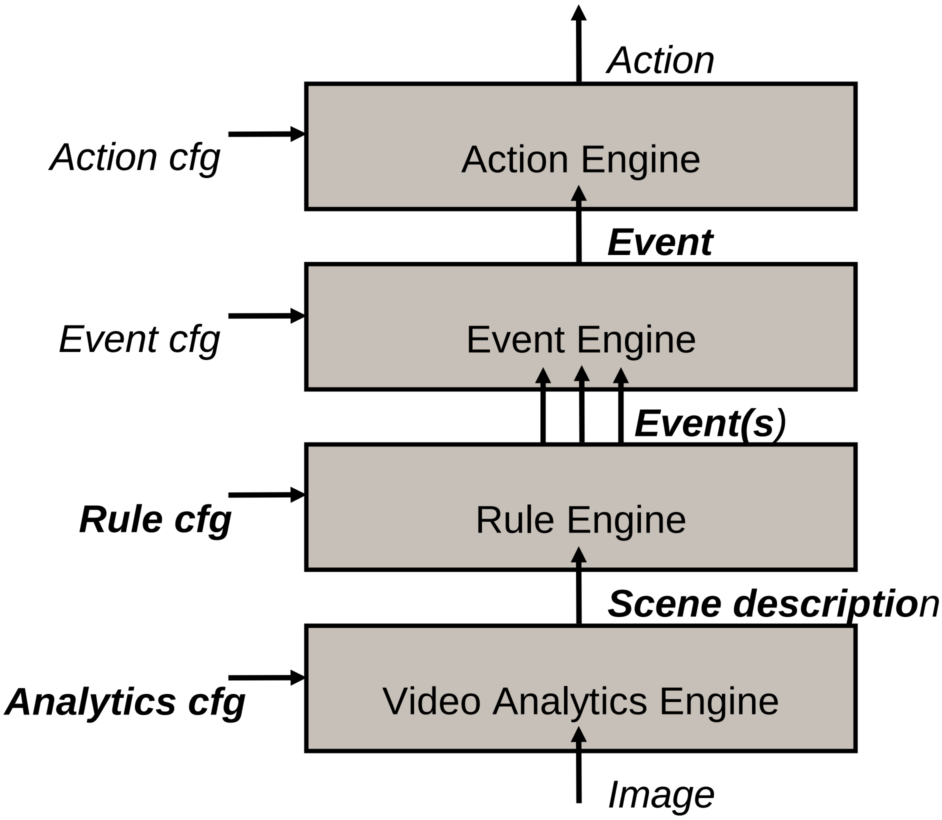

Figure 1: Video analytics architecture

The following interfaces are defined in this standard:

- Analytics Configuration Interface
- Scene Description
- Rule Configuration Interface

This specification defines a configuration framework for the video analytics engine called ONVIF Video Analytics Service. This framework enables a client to contact a device implementing the ONVIF Video 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 Video Analytics Modules in parallel if supported by the device.

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

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 these rules from the application to the device.

A device supporting ONVIF Video 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.

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

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

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

5 Service

This section covers the following main areas of this architecture:

- Analytics Module interface
- Scene description
- Rules interface

The analytics service allows fine-grained configuration of individual rules and individual analytics modules (see 5.2 and 5.3). 5.1 introduces the XML-based scene description, which can be streamed as metadata to clients via RTP as defined in the ONVIF Streaming Specification.

5.1 Scene Description Interface

5.1.1 Overview

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

5.1.2 Frame Related Content

The input of the Video 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 Video Analytics Engine (information such as video input line, video resolution, frame cropping, frame rate etc.), the temporal frame association within the input stream, and the spatial positioning of elements within a frame.

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

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

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

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

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

5.1.2.1 Temporal Relation

Since multiple scene elements can be extracted from the same image, scene elements are listed below a frame node which 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.

Example:

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

5.1.2.2 Spatial Relation

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

---

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

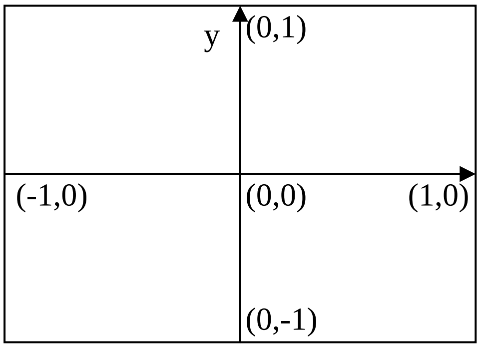

Figure 2: Default frame coordinate system

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

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

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

It follows a mathematical description of coordinate systems and transformations. A coordinate system 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 default coordinate system by the following formula: $\begin{pmatrix} q_x \\ q_y \end{pmatrix} = \begin{pmatrix} p_x \cdot s_x + t_x \\ p_y \cdot s_y + t_y \end{pmatrix}$. Similarly, a vector ***v*** given with respect to the coordinate system is transformed into the corresponding vector ***w*** of the default coordinate system by: $\begin{pmatrix} w_x \\ w_y \end{pmatrix} = \begin{pmatrix} v_x \cdot s_x \\ v_y \cdot s_y \end{pmatrix}$.

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

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>
```

5.1.3 Scene Elements

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

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

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

5.1.3.1 Objects

Objects are identified via their ObjectID. Features relating to one particular object are collected in an object node with the corresponding ObjectID as an attribute. Associations of objects, like 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:extension>
  </xs:complexContent>
</xs:complexType>
```

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

The object node has two placeholders for Appearance and Behaviour information. The appearance node starts with an optional transformation node which can be used to change from a frame-centric coordinate system to an object-centric coordinate system. Next, the Shape of an object can be specified. If an object is detected in a frame, the shape information should be present in the appearance description. The video 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.

Other object features like colour and object class can be added to the appearance node. This specification focuses on the shape descriptors (see section 5.1.3.3). The definition of colour and object class can be found in Appendix B.1.

This specification defines two standard behaviours for objects. When an object stops moving, it can be marked as either 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 although 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="30.0" right="100.0" bottom="80.0"/>
        <tt:CenterOfGravity x="60.0" y="50.0"/>
      </tt:Shape>
    </tt:Appearance>
  </tt:Object>
</tt:Frame>
...
<tt:Frame UtcTime="2008-10-10T12:24:57.421">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1.0"/>
    <tt:Scale x="0.003125" y="0.00416667"/>
  </tt:Transformation>
  <tt:Object ObjectId="12">
    <tt:Appearance>
      <tt:Shape>
        <tt:BoundingBox left="20.0" top="30.0" right="100.0" bottom="80.0"/>
        <tt:CenterOfGravity x="60.0" y="50.0"/>
      </tt:Shape>
    </tt:Appearance>
    <tt:Behaviour>
      <tt:Idle/>
    </tt:Behaviour>
  </tt:Object>
</tt:Frame>
...
<tt:Frame UtcTime="2008-10-10T12:24:57.521">
```

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

5.1.3.2 Object Tree

When two objects come too close to each other, such that the video 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 video 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 video 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 video 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 video analytics algorithm, however, could not determine where the split object came from.

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

A deleted ObjectId shall not be reused within the Scene Description until the ObjectId container has wrapped around.

Example:

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

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

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

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

5.1.3.3 Shape descriptor

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

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

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

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

Example:

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

5.2 Rule interface

A XML structure is introduced in Section 5.2.1 to communicate the configuration of rules. Section 5.2.2 specifies a language to describe the configuration of a specific rule type. In section 5.2.3 two rules are specified that should be supported by a device implementing a Rule Engine. Section 5.2.4 introduces operations to manage rules. If the device supports a Rule Engine, it shall implement the complete rule Interface.

5.2.1 Rule representation

The configuration of a rule has two required attributes: one specifies the name and the other specifies the type of the rule. 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 by SimpleItems.

The following example shows a complete video analytics configuration containing two rules:

```
<tt:VideoAnalyticsConfiguration>
  <tt:AnalyticsEngineConfiguration>
    ...
  </tt:AnalyticsEngineConfiguration>
  <tt:RuleEngineConfiguration>
    <tt:Rule Name="MyLineDetector" Type="tt:LineDetector">
      <tt:Parameters>
        <tt:SimpleItem Name="Direction" Value="Any"/>
        <tt:ElementItem Name="Segments">
          <tt:Polyline>
            <tt:Point x="10.0"  y="50.0"/>
            <tt:Point x="100.0" y="50.0"/>
          </tt:Polyline>
        </tt:ElementItem>
      </tt:Parameters>
    </tt:Rule>
    <tt:Rule Name="MyFieldDetector" Type="tt:FieldDetector">
      <tt:Parameters>
        <tt:ElementItem Name="Field">
          <tt:Polygon>
            <tt:Point x="10.0"  y="50.0"/>
            <tt:Point x="100.0" y="50.0"/>
            <tt:Point x="100.0" y="150.0"/>
          </tt:Polygon>
        </tt:ElementItem>
      </tt:Parameters>
    </tt:Rule>
  </tt:RuleEngineConfiguration>
</tt:VideoAnalyticsConfiguration>
```

5.2.2 Rule description language

The description of a rule contains the type information of all parameters belonging to a certain rule type and the description of the output produced by such a rule. The output of the Rule Engine is Events which can either be used in an Event Engine or be subscribed to by a client.

The parameters of a certain rule type are listed below the ParameterDescription element. All parameters are either SimpleItems or ElementItems and can be described by either a SimpleItemDescription or an ElementItemDescription. Both ItemDescriptions 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.

The output produced by this rule type is described in multiple MessageDescription elements. Each MessageDescription contains a description of the message payload according to the Message Description Language detailed in the ONVIF Core specification. Additionally, the MessageDescription 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.

Section 5.2.3 demonstrates the usage of the Rule Description Language. Below, the definitions are included for convenience[4]:

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

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

5.2.3 Specified Rules

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

---

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

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

5.2.3.1 LineDetector

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

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

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

5.2.3.2 FieldDetector

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.

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

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

```
      tns1:RuleEngine/FieldDetector/ObjectsInside
    </tt:ParentTopic>
  </tt:MessageDescription>
</tt:RuleDescription>
```

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

5.2.4 Operations on rules

If the device supports a Rule Engine as defined by ONVIF, then it shall implement the following operations to manage rules. The Create/Delete/Modify operations are atomic, meaning that either all modifications can be processed or the complete operation shall fail.

5.2.4.1 Get Supported rules

The device shall indicate the rules it supports by implementing the subsequent operation. It returns a list of rule descriptions according to the Rule Description Language described in Section 5.2.2. 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. If rule descriptions reference types or elements of the ONVIF schema file, the ONVIF schema file shall be explicitly listed.

Table 1: GetSupportedRules command

GetSupportedRules	Access Class: READ_MEDIA
Message name	**Description**
GetSupportedRulesRequest	*The request message contains the VideoAnalyticsConfigurationToken for which the supported rules should be listed.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetSupportedRulesResponse	*The response contains the supported rules.* tt: SupportedRules **SupportedRules** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*VideoAnalyticsConfiguration does not exist.*

5.2.4.2 Get Rules

The following operation retrieves the currently installed rules:

Table 2: GetRules command

GetRules	Access Class: READ_MEDIA
Message name	**Description**
GetRulesRequest	*The request message specifies the VideoAnalyticsConfigurationToken for which the rules should be reported.* tt:ReferenceToken **ConfigurationToken** [1][1]

GetRulesResponse	*The response is a list of installed rules for the specified configuration.* tt:Config **Rule** [0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The VideoAnalyticsConfiguration does not exist.*

5.2.4.3 Create rules

The following operation adds rules to a VideoAnalyticsConfiguration. If all rules can not be created as requested, the device responds with a fault message.

Table 3: CreateRules command

CreateRules	Access Class: ACTUATE
Message name	**Description**
CreateRulesRequest	*The request message specifies the VideoAnalyticsConfigurationToken to which the listed Rules should be added*. tt:ReferenceToken **ConfigurationToken** [1][1] tt:Config **Rule** [1][unbounded]
CreateRulesResponse	This is an empty message.
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The VideoAnalyticsConfiguration does not exist.*
env:Sender ter:InvalidArgVal ter:InvalidRule	*The suggested rules configuration is not valid on the device.*
env:Sender ter:InvalidArgVal ter:RuleAlreadyExistent	*The same rule name exists already in the configuration.*
enc: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.*

5.2.4.4 Modify Rules

The following operation modifies multiple rules. If all rules can not be modified as requested, the device responds with a fault message.

Table 4: ModifyRules command

ModifyRules	Access Class: ACTUATE
Message name	**Description**
ModifyRulesRequest	*The request message specifies the VideoAnalyticsConfigurationToken for which the listed Rules should be modified.*

	tt:ReferenceToken **ConfigurationToken** [1][1] tt:Config **Rule**[1][unbounded]
ModifyRulesResponse	This is an empty message.
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The VideoAnalyticsConfiguration does not exist.*
env:Sender ter:InvalidArgVal ter:InvalidRule	*The suggested rules configuration is not valid on the device.*
env:Sender ter:InvalidArgs ter:RuleNotExistent	*The rule name or names do not exist.*
enc: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:ConflictingConfig	*The device cannot modify the rules without creating a conflicting configuration.*

5.2.4.5 Delete Rules

The following operation deletes multiple rules. If all rules can not be deleted as requested, the device responds with a fault message.

Table 5: DeleteRules command

DeleteRules	Access Class: ACTUATE
Message name	**Description**
DeleteRulesRequest	*The request message specifies the VideoAnalyticsConfigurationToken from which the listed Rules should be removed.* tt:ReferenceToken **ConfigurationToken** [1][1] xs:string **RuleName** [1][unbounded]
DeleteRulesResponse	*The response is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The VideoAnalyticsConfiguration does not exist.*
env:Receiver ter:Action ter:ConflictingConfig	*The device cannot delete the rules without creating a conflicting configuration.*
env:Sender ter:InvalidArgs ter:RuleNotExistent	*The rule name or names do not exist.*

5.3 Analytics Modules Interface

Section 5.3.1 defines an XML structure that communicates the configuration of Analytics Modules. Section 5.3.2 defines the language that describes the configuration of a specific analytics module. Section 5.3.3 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.

5.3.1 Analytics module configuration

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

```
<tt:VideoAnalyticsConfig>
  <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>
  <tt:RuleEngineConfig>
    ...
  </tt:RuleEngineConfig>
</tt:VideoAnalyticsConfig>
```

5.3.2 Analytics Module Description Language

The Analytics Module reuses the Rule Description Language, described in Section 5.2.2. The following AnalyticsModuleDescription element replaces the RuleDescription element:

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

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

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

5.3.3 Operations on Analytics Modules

If the device supports an analytics engine as defined by ONVIF, it shall support the subsequent operations to manage analytics modules. The Create/Delete/Modify operations shall be atomic, all modifications can be processed or the complete operation shall fail.

5.3.3.1 GetSupportedAnalyticsModules

The device indicates the analytics modules it supports by implementing the GetSupportedAnalysticsModule operation. It returns a list of analytics modules according to the Analytics Module Description Language, described in section 5.2.2. 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 analytics module descriptions. If the analytics module descriptions reference types or elements of the ONVIF schema file, the ONVIFschema file shall be explicitly listed.

Table 6: GetSupportedAnalyticsModules command

GetSupportedAnalyticsModules	Access Class: READ_MEDIA
Message name	**Description**
GetSupportedAnalyticsModulesRequest	T*he request message contains the VideoAnalyticsConfigurationToken for which the supported analytics modules should be listed.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetSupportedAnalyticsModulesResponse	*The response contains the supported analytics modules.* **SupportedAnalyticsModules** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgs ter:NoConfig	*VideoAnalyticsConfiguration does not exist.*

5.3.3.2 GetAnalytics Modules

The following operation retrieves the currently installed analytics modules:

Table 7: GetAnalyticsModules command

GetAnalyticsModules	Access Class: READ_MEDIA
Message name	**Description**
GetAnalyticsModulesRequest	*The request message specifies the VideoAnalyticsConfigurationToken for which the analytics modules should be reported.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetAnalyticsModulesResponse	*The response is a list of installed analytics modules for the specified configuration.* tt:Config **AnalyticsModule** [0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgs ter:NoConfig	*The VideoAnalyticsConfiguration does not exist.*

5.3.3.3 CreateAnalytics Modules

The following operation adds analytics modules to a VideoAnalyticsConfiguration. If all analytics modules can not be created as requested, the device responds with a fault message.

Table 8: CreateAnalyticsModules command.

CreateAnalyticsModules	Access Class: ACTUATE
Message name	**Description**
CreateAnalyticsModulesRequest	*The request message specifies the VideoAnalyticsConfigurationToken to which the listed Analytics Modules should be added.* tt:ReferenceToken **ConfigurationToken** [1][1] tt:Config **AnalyticsModule** [1][unbounded]
CreateAnalyticsModulesResponse	This is an empty message.
Fault codes	**Description**
env:Sender ter:InvalidArgs ter:NoConfig	*The VideoAnalyticsConfiguration does not exist.*
env:Sender ter:InvalidArgs ter:NameAlreadyExistent	*The same analytics module name exists already in the configuration.*
enc:Receiver ter:Action ter:TooManyModules	*There is not enough space 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.*
env:Sender ter:InvalidArgVal ter:InvalidModule	*The suggested module configuration is not valid on the device.*

5.3.3.4 ModifyAnalytics Modules

The following operation modifies multiple analytics modules. If all analytics modules can not be modified as requested, the device respond with a fault message.

Table 9: ModifyAnalyticsModules command

ModifyAnalyticsModules	Access Class: ACTUATE
Message name	**Description**
ModifyAnalyticsModulesRequest	*The request message specifies the VideoAnalyticsConfigurationToken for which the listed analytics modules should be modified.* tt:ReferenceToken **ConfigurationToken** [1][1] tt:Config **AnalyticsModule** [1][unbounded]
ModifyAnalyticsModulesResponse	The response is an empty message.
Fault codes	**Description**

env:Sender ter:InvalidArgs ter:NoConfig	*The VideoAnalyticsConfiguration does not exist.*
env:Sender ter:InvalidArgs ter:NameNotExistent	*The analytics module with the requested name does not exist.*
enc:Receiver ter:Action ter:TooManyModules	*There is not enough space in the device to add the analytics modules to the configuration.*
env:Receiver ter:Action ter:ConfigurationConflict	*The device cannot modify the analytics modules without creating a conflicting configuration.*
env:Sender ter:InvalidArgVal ter:InvalidModule	*The suggested module configuration is not valid on the device.*

5.3.3.5 DeleteAnalytics Modules

The following operation deletes multiple analytics modules. If all analytics modules can not be deleted as requested, the device responds with a fault message.

Table 10: DeleteAnalyticsModules command

DeleteAnalyticsModules	Access Class: ACTUATE
Message name	**Description**
DeleteAnalyticsModulesRequest	*The request message specifies the VideoAnalyticsConfigurationToken from which the listed Analytics Modules should be removed.* tt:Reference**Token ConfigurationToken** [1][1] xs:string **AnalyticsModuleName** [1][unbounded]
DeleteAnalyticsModulesResponse	The response is an empty message.
Fault codes	**Description**
env:Sender ter:InvalidArgs ter:NoConfig	*The VideoAnalyticsConfiguration does not exist.*
env:Receiver ter:Action ter:ConfigurationConflict	*The device cannot delete the analytics modules without creating a conflicting configuration.*
env:Sender ter:InvalidArgs ter:NameNotExistent	*The analytics module with the requested name does not exist.*

5.4 Capabilities

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

RuleSupport: Indication that the device supports rules interface and rules syntax as specified in Section 5.2

AnalyticsModuleSupport: Indication that the device supports the scene analytics module interface as specified in Section 5.3.

Table 11: GetServiceCapabilities command

GetServiceCapabilities	Access Class: PRE_AUTH
Message name	**Description**
GetServiceCapabilitiesRequest	*This message contains a request for device capabilities.*
GetServiceCapabilitiesResponse	*The capability response message contains the requested service capabilities using a hierarchical XML capability structure.* tan:Capabilities **Capabilities** [1][1]
Fault codes	**Description**
	No command specific faults!

5.5 Service specific data types

This service does reuse the tt:Config data type defined in the ONVIF Event Service Specification. It does not introduce own data types for the service itself.

For the data types that describe the scene description refer to the onvif.xsd schema file.

5.6 Service-specific fault codes

Table 12 below lists the analytics service-specific fault codes. Each command can also generate a generic fault.

The specific faults are defined as subcode of a generic fault. The parent generic subcode is the subcode at the top of each row below and the specific fault subcode is at the bottom of the cell.

Table 12: The analytics-specific fault codes

Fault Code	Parent Subcode Subcode	Fault Reason	Description
env:Receiver	ter:Action ter:TooManyRules	No more space available.	There is not enough space in the device to add the rules to the configuration.
env:Receiver	ter:Action ter:TooManyModules	No more space available.	There is not enough space in the device to add the analytics modules to the configuration.
env:Receiver	ter:Action ter:ConfigurationConflict	Conflict when using new settings	The new settings result in an inconsistent configuration.
env:Sender	ter:InvalidArgVal	No such	The requested

	ter:NoConfig	configuration	VideoAnalyticsConfiguration does not exist.
env:Sender	ter:InvalidArgVal ter:InvalidRule	The rule is invalid.	The suggested rule configuration is not valid.
env:Sender	ter:InvalidArgVal ter:InvalidModule	The module is invalid	The suggested analytics module configuration is not valid on the device.
env:Sender	ter:InvalidArgVal ter:RuleAlreadyExistent	The rule exists	The same rule name exists already in the configuration.
env:Sender	ter:InvalidArgs ter:RuleNotExistent	The rule does not exist	The rule name or names do not exist.
env:Sender	ter:InvalidArgs ter:NameAlreadyExistent	The name exists	The same analytics module name exists already in the configuration.
env:Sender	ter:InvalidArgs ter:NameNotExistent	The name does not exist	The analytics module with the requested name does not exist.

Annex A
(informative)

Scene descriptions

A.1 Colour Descriptor

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

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

```
<xs:complexType name="ColorDescriptor">
  <xs:sequence>
    <xs:element name="ColorCluster" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Color" type="tt:Color"/>
          <xs:element name="Weight" type="xs:float" minOccurs="0"/>
          <xs:element name="Covariance" type="tt:ColorCovariance" minOccurs="0"/>
          ...
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Color">
  <xs:attribute name="X" type="xs:float" use="required"/>
  <xs:attribute name="Y" type="xs:float" use="required"/>
  <xs:attribute name="Z" type="xs:float" use="required" />
  <xs:attribute name="Colorspace" type="xs:anyURI"/>
</xs:complexType>

<xs:complexType name="ColorCovariance">
  <xs:attribute name="XX" type="xs:float" use="required"/>
  <xs:attribute name="YY" type="xs:float" use="required"/>
  <xs:attribute name="ZZ" type="xs:float" use="required" />
  <xs:attribute name="XY" type="xs:float"/>
  <xs:attribute name="XZ" type="xs:float"/>
  <xs:attribute name="YZ" type="xs:float" />
  <xs:attribute name="Colorspace" type="xs:anyURI"/>
</xs:complexType>
```

A.2 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 sum of the likelihoods shall NOT exceed 1.

```
<xs:simpleType name="ClassType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Animal"/>
    <xs:enumeration value="Face"/>
    <xs:enumeration value="Human"/>
    <xs:enumeration value="Vehicle"/>
    <xs:enumeration value="Other"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="ClassDescriptor">
  <xs:sequence>
    <xs:element name="ClassCandidate" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Type" type="tt:ClassType"/>
          <xs:element name="Likelihood" type="xs:float"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```