Driving IP-based physical security through global standardization

ONVIF™

ONVIF Specification Version 19.12
Release Notes

1. Summary

The ONVIF 19.12 release incorporates a number of major enhancements and minor clarifications for better interoperability among ONVIF conformant clients and devices. The changes themselves are described in details in the list below chapters 2 and 3.

2. Additions

This release adds the following features.

2.1. Client Certificate User Authorization

The ONVIF Security Specification adds procedures using TLS client authentication as secure alternative to digest authentication for authorization.

2.2. Multi track streaming

The ONVIF Streaming Specification adds grouping of multiple streams to a single RTSP session for multiple image sensor devices. Additionally the ONVIF Media2 Service Specification now includes a mechanism for grouping media profiles.

2.3. Resource Query for Recordings

This enhances ONVIF resource query to support recording data in media storage.

2.4. Line and Area Rule Enhancement

A generic mechanism for combining object classification with detectors like line and field detector has been added to the ONVIF Analytics Specification.

2.5. Get Supported Metadata

A method has been added to the ONVIF Analytics Specification that enables clients to detect what metadata algorithms can produce.

2.6. Face and Human Body Metadata

The metadata streaming now may include human body and face descriptions. See the ONVIF Analytics Specification for a detailed description of the available features.

2.7. Image Transmission

Two mechanism for transmitting images in events and metadata have been defined allowing to convey images via external URL references or embedded in the stream.

2.8. Credential Service Extensions

The ONVIF Credential Service Specification now enables clients to retrieve or modify the device credential lists for white and black listing.

2.9. Application Management

This release adds a new specification to the ONVIF series for managing applications including installation and licensing.

3. Changes

Find below all errata from Version 19.06 to 19.12 in order to improve interoperability. The numbers correspond to the Change Request ticket numbers and are not necessarily continuously ascending.

If not noted otherwise the changes refer to the Core specification.

2470 Add pt:LPR as new identifier type in Credential Specification

Replace a description of *SupportedIdentifierType* at section *5.2.2.1 ServiceCapabilities* in Credential Service specification.

- SupportedIdentifierType

  A list of identifier types that the device supports. Identifier types starting with the prefix pt: are reserved to define ONVIF-specific types. For custom defined identifier types, free text can be used. The following types are defined by ONVIF:
  - pt:Card Supports Card identifier type (including fob, tag and similar)
  - pt:PIN Supports PIN identifier type
  - pt:Fingerprint Supports Fingerprint biometric identifier type
  - pt:Face Supports Face biometric identifier type
  - pt:Iris Supports Iris biometric identifier type
  - pt:Vein Supports Vein biometric identifier type
  - pt:Palm Supports Palm biometric identifier type
  - pt:Retina Supports Retina biometric identifier type

with

- SupportedIdentifierType

A list of identifier types that the device supports. Is of type text.
Identifier types starting with the prefix pt: are reserved to define ONVIF-specific types as defined in pt:RecognitionType. Please note that pt:REX is not an identifier type. For custom defined identifier types, free text can be used.

And replace the corresponding documentation in credential.wsdl.

```
<xs:element name="SupportedIdentifierType" type="pt:Name" minOccurs="1"
maxOccurs="unbounded">
  <xs:annotation>
    <xs:documentation>
      A list of identifier types that the device supports. Identifiers types starting with
      the prefix pt: are reserved to define ONVIF specific types. For custom defined identifier
types
      shall all share the "pt:&lt;Name&gt;" syntax.
    </xs:documentation>
```

With

```
<xs:element name="SupportedIdentifierType" type="pt:Name" minOccurs="1"
maxOccurs="unbounded">
  <xs:annotation>
    <xs:documentation>
      A list of identifier types that the device supports. Is of type text.
      Identifier types starting with the prefix pt: are reserved to define ONVIF-specific
      types as defined in pt:RecognitionType. Please note that pt:REX is not an identifier
      type. For custom defined identifier types, free text can be used.
    </xs:documentation>
```

2471 Add pt:LPR as new recognition type in Authentication Behavior Specification

Replace a description of *RecognitionType* at section *5.3.2.4 RecognitionMethod* in Authentication Behavior Service specification.

- RecognitionType

The requested type of recognition. Is of type text. Can be either one of the following reserved ONVIF types, or a custom type:

- o pt:Card Card is used as recognition method
- o pt:PIN PIN is used as recognition method
- o pt:Fingerprint A fingerprint scan is used as recognition method
- o pt:Face A facial scan is used as recognition method
- o pt:Iris An iris scan is used as recognition method

- pt:Vein A vein scan is used as recognition method
- pt:PalmA palm scan is used as recognition method
- pt:REX A request-to-exit button is used to be granted access

with

• RecognitionType

The requested type of recognition. Is of type text.

Recognition types starting with the prefix pt: are reserved to define ONVIF-specific types as defined in pt:RecognitionType. For custom defined identifier types, free text can be used.

And replace the corresponding documentation in authenticationbehavior.wsdl.

```
<xs:element name="RecognitionType" type="xs:string">
  <xs:annotation>
    <xs:documentation>The requested type of recognition.</xs:documentation>
```

With

```
<xs:element name="RecognitionType" type="xs:string">
  <xs:annotation>
    <xs:documentation>
      The requested type of recognition. Is of type text.
      Recognition types starting with the prefix pt: are reserved to define
      ONVIF-specific types as defined in pt:RecognitionType. For custom defined
      identifier types, free text can be used.
    </xs:documentation>
```

2474 Add capability for supported recognition types in Access Control Specification

Add some additional descriptions for *SupportedSecurityLevels* and add a new parameter *SupportedRecognitionTypes* at section 5.2.1.3 in Access Control Service specification. That is changed from

• SupportedSecurityLevels

A list of security level tokens that this access point supports. See [ONVIF Authentication Behavior Service Specification].

to

• SupportedSecurityLevels

A list of security level tokens that this access point supports. See [ONVIF Authentication Behavior Service Specification].

This field is optional, and if omitted, the device cannot support multi-factor authentication for this access point.

Please note that when an access point is updated, then any previously supported security levels are replaced with the new list.

• SupportedRecognitionTypes

A list of recognition types that the device supports. This field is only relevant for devices that are not aware of security levels (see [ONVIF Authentication Behavior Service Specification]).

Please note that when an access point is updated, then any previously supported recognition types are replaced with the new list.

Recognition types starting with the prefix pt: are reserved to define ONVIF-specific types as defined in pt:RecognitionType. For custom defined identifier types, free text can be used.

And add appropriate documentation and attribute in accesscontrol.wsdl. That is replaced

```
<xs:schema targetNamespace="http://www.onvif.org/ver10/accesscontrol/wsdl"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:pt="http://www.onvif.org/ver10/pacs"
  xmlns:tac="http://www.onvif.org/ver10/accesscontrol/wsdl"
  elementFormDefault="qualified"
  version="19.12">
  <xs:import namespace="http://www.onvif.org/ver10/pacs" schemaLocation="types.xsd"/>
```

With

```
<xs:schema targetNamespace="http://www.onvif.org/ver10/accesscontrol/wsdl"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:pt="http://www.onvif.org/ver10/pacs"
  xmlns:tac="http://www.onvif.org/ver10/accesscontrol/wsdl"
  elementFormDefault="qualified"
  version="19.12">
  <xs:import namespace="http://www.onvif.org/ver10/pacs" schemaLocation="types.xsd"/>
  <xs:import namespace="http://www.onvif.org/ver10/schema"
schemaLocation="../schema/onvif.xsd"/>
```

Replace

```
<xs:element name="SupportedSecurityLevels" type="pt:ReferenceToken" minOccurs="0"
maxOccurs="unbounded">
   <xs:annotation>
      <xs:documentation>A list of security level tokens that this access point supports.
         See [Authentication Behavior Service Specification].
      </xs:documentation>
```

With

```
<xs:element name="SupportedSecurityLevels" type="pt:ReferenceToken" minOccurs="0"
maxOccurs="unbounded">
```

<xs:annotation>
<xs:documentation>A list of security level tokens that this access point supports.
See [Authentication Behavior Service Specification].
This field is optional, and if omitted, the device cannot support multi-factor authentication for this access point.
Please note that when an access point is updated, then any previously supported security levels are replaced with the new list.
</xs:documentation>

Add following attribute definition as the last attribute of *AccessPointCapabilities* complexType.

<xs:attribute name="SupportedRecognitionTypes" type="tt:StringAttrList">
<xs:annotation>
<xs:documentation>
A list of recognition types that the device supports. This field is only relevant for devices that are not aware of security levels (see [ONVIF Authentication Behavior Service Specification]).
Please note that when an access point is updated, then any previously supported recognition types
are replaced with the new list.
Recognition types starting with the prefix pt: are reserved to define ONVIF-specific types as defined in pt:RecognitionType. For custom defined identifier types, free text can be used.
</xs:documentation>
</xs:annotation>
</xs:attribute>

2480 Add capability for managing doors

Add following *DoorManagementSupported* parameter description at section *5.2.2.1 ServiceCapabilities* in Door Control Service specification.

• DoorManagementSupported
Indicates that the client can perform CRUD operations (create, read, update and delete) on doors. To enable the use of the commands GetDoors, GetDoorList, CreateDoor, ModifyDoor and DeleteDoor, the value must be set to true.

Insert next sentence into just before each tables listed below.

Sentence:

A device that signals support for the DoorManagementSupported capability shall implement this command.

Tables:

Table 5 GetDoors command
Table 6 GetDoorList command
Table 7 CreateDoor command
Table 9 ModifyDoor command
Table 10 DeleteDoor command

Add following attribute definition as the last attribute of *ServiceCapabilities* complexType in doorcontrol.wsdl.

```
<xs:attribute name="DoorManagementSupported" type="xs:boolean" default="false">
  <xs:annotation>
    <xs:documentation>
      Indicates that the client can perform CRUD operations (create, read, update and delete)
      on doors. To enable the use of the commands GetDoors, GetDoorList, CreateDoor,
ModifyDoor
      and DeleteDoor, the value must be set to true.
    </xs:documentation>
  </xs:annotation>
</xs:attribute>
```

2481 Add capability for managing access points and areas

Add following *AccessPointManagementSupported* and *AreaManagementSupported* parameter description at section *5.1.2.1 ServiceCapabilities* in Access Control Service specification.

• AccessPointManagementSupported
Indicates that the client can perform CRUD operations (create, read, update and delete) on access points. To enable the use of the commands GetAccessPoints, GetAccessPointList, CreateAccessPoint, ModifyAccessPoint, DeleteAccessPoint, SetAccessPointAuthenticationProfile and DeleteAccessPointAuthenticationProfile, the value must be set to true.
• AreaManagementSupported
Indicates that the client can perform CRUD operations (create, read, update and delete) on areas. To enable the use of the commands GetAreas, GetAreaList, CreateArea, ModifyArea and DeleteArea, the value must be set to true.

Insert next sentences into just before each tables listed below.

Sentence:

A device that signals support for the AccessPointManagementSupported capability shall implement this command.

Tables:

Table 5 GetAccessPoints command

Table 6 GetAccessPointList command

Table 7 CreateAccessPoint command

Table 9 ModifyAccessPoint command

Table 10 DeleteAccessPoint command

Table 11 SetAccessPointAuthenticationProfile command

Table 12 DeleteAccessPointAuthenticationProfile command

Sentence:

A device that signals support for the AreaManagementSupported capability shall implement this command.

Tables:

Table 15 GetAreas command

Table 16 GetAreaList command

Table 17 CreateArea command

Table 19 ModifyArea command

Table 20 DeleteArea command

Add following attribute definitions as the last attribute of *ServiceCapabilities* complexType in accesscontrol.wsdl.

```
<xs:attribute name="AccessPointManagementSupported" type="xs:boolean" default="false">
  <xs:annotation>
    <xs:documentation>
      Indicates that the client can perform CRUD operations (create, read, update and delete)
      on access points. To enable the use of the commands GetAccessPoints,
GetAccessPointList,
      CreateAccessPoint, ModifyAccessPoint, DeleteAccessPoint,
SetAccessPointAuthenticationProfile
      and DeleteAccessPointAuthenticationProfile, the value must be set to true.
    </xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="AreaManagementSupported" type="xs:boolean" default="false">
  <xs:annotation>
    <xs:documentation>
      Indicates that the client can perform CRUD operations (create, read, update and delete)
```

on areas. To enable the use of the commands GetAreas, GetAreaList, CreateArea, ModifyArea
and DeleteArea, the value must be set to true.
</xs:documentation>
</xs:annotation>
</xs:attribute>

2489 Improve descriptionin of AEConfig parameter in MetadataConfiguration

Add following annotation into *AnalyticsEngineConfiguration* element definition under *MetadataConfiguration* complexType in onvif.xsd, as replace

```
<xs:complexType name="MetadataConfiguration">
  <xs:complexContent>
    <xs:extension base="tt:ConfigurationEntity">
      <xs:sequence>
      …
      <xs:element name="AnalyticsEngineConfiguration" type="tt:AnalyticsEngineConfiguration" minOccurs="0"/>
```

With

```
<xs:complexType name="MetadataConfiguration">
  <xs:complexContent>
    <xs:extension base="tt:ConfigurationEntity">
      <xs:sequence>
      …
      <xs:element name="AnalyticsEngineConfiguration" type="tt:AnalyticsEngineConfiguration" minOccurs="0">
        <xs:annotation>
          <xs:documentation>Indication which AnalyticsModules shall output metadata.
          Note that the streaming behavior is undefined if the list includes items that are not part of the associated AnalyticsConfiguration.</xs:documentation>
        </xs:annotation>
      </xs:element>
```

Replace the first paragraph of section *5.10 Metadata configuration* in Media Service specification.

A MetadataConfiguration contains parameters for selecting the data to include in the metadata stream. The choices include PTZ status, PTZ position, events as defined by a subscription and analytics data . The event subscription data is described in the section “Event Handling” of the ONVIF Core Specification. The analytics parameters define which data to include from the analytics engine part of the profile, see Section 5.9.

With

For PTZ transmission of status and position change information can be enabled separately.
Event streaming can be enabled and controlled using topic filters. For topic filter configuration refer to section “Event Handling” of the ONVIF Core Specification.
Streaming of scene description can be enabled. Optionally the AnalyticsEngineConfiguration allows to restrict streaming of scene description to the provided list of AnalyticsModules. Note that analytics modules only generate scene description if they are configured in the AnalyticsConfiguration of the profile as defined in section 5.9.
A device shall ignore any analytics module parameters passed to the SetMetadataConfiguration command and should not list AnalyticsModule/Parameters.

Reaplace the second paragraph of section *5.2.8 Metadata Configuration* in Media2 Service specification.

A MetadataConfiguration contains parameters for selecting the data to include in the metadata stream. The choices include PTZ status, PTZ position, events as defined by a subscription and analytics data . The event subscription data is described in the section “Event Handling” of the ONVIF Core Specification. The analytics parameters define which data to include from the analytics engine part of the profile; see Section 5.2.7.

With

For PTZ transmission of status and position change information can be enabled separately.
Event streaming can be enabled and controlled using topic filters. For topic filter configuration refer to section “Event Handling” of the ONVIF Core Specification.
Streaming of scene description can be enabled. Optionally the AnalyticsEngineConfiguration allows to restrict streaming of scene description to the provided list of AnalyticsModules. Note that analytics modules only generate scene description if they are configured in the AnalyticsConfiguration of the profile as defined in section 5.2.7.
A device shall ignore any analytics module parameters passed to the SetMetadataConfiguration command and should not list AnalyticsModule/Parameters.

2504 Relationship between objects in metadata

Insert *Parent* attribute into *Object* complexType in metadata.xsd, as shown following underlined part.

```
<xs:complexType name="Object">
  <xs:complexContent>
    <xs:extension base="tt:ObjectId">
      <xs:sequence>
        <xs:element name="Appearance" type="tt:Appearance" minOccurs="0"/>
        <xs:element name="Behaviour" type="tt:Behaviour" minOccurs="0"/>
        <xs:element name="Extension" type="tt:ObjectExtension" minOccurs="0"/>
```

```
            <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>       <!-- reserved for ONVIF -->
         </xs:sequence>
         <xs:attribute name="Parent" type="xs:integer">
            <xs:annotation>
               <xs:documentation>Objectid of the parent object. eg: License plate object has Vehicle
object as parent.</xs:documentation>
            </xs:annotation>
         </xs:attribute>
         <xs:anyAttribute processContents="lax"/>
      </xs:extension>
```

Insert the following description about the *Parent* attribute and its example at the bottom of section *5.1.3.1 Objects* in Analytics Service specification.

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

Example:

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

<tt:Frame UtcTime="2019-06-10T12:24:57.721">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1.0"/>
    <tt:Scale x="0.003125" y="0.00416667"/>
```

```
  </tt:Transformation>
  <tt:Object ObjectId="14" Parent="12">
    <tt:Appearance>
      <tt:Shape>
        <tt:BoundingBox left="40.0" top="100.0" right="70.0" bottom="150.0" />
        <tt:CenterOfGravity x="57.0" y="130.0" />
      </tt:Shape>
      <tt:Class>
          <tt:Type Likelihood="0.6">LicensePlate</tt:Type>
      </tt:Class>
    </tt:Appearance>
  </tt:Object>
</tt:Frame>
```

2517 limitation for local token

Add annotation of limitation length of *ReferenceToken* as underlined part in following text in common.xsd.

```
<xs:simpleType name="ReferenceToken">
  <xs:annotation>
    <xs:documentation>Unique identifier for a physical or logical resource.
    Tokens should be assigned such that they are unique within a device. Tokens must be at least unique within its class.
    Length up to 64 characters. Token may be extended by intermediate terminal with adding prefix to make it global unique.
    The length should be within 36 characters for generating at local device. See "Remote Token" section in Resource Query specification.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:maxLength value="64"/>
  </xs:restriction>
</xs:simpleType>
```

2527 [Analytics Modules] Operations on analytics modules support condition

Remove the following generic requirement sentence from section *5.4.3 Operations on analytics modules* in Analytics Service specification.

> If the device supports an analytics engine as defined by ONVIF, it shall support the subsequent operations to manage analytics modules.

2528 [Analytics Modules] ConfigurationToken description for requests clarification

Replace the description of section *5.4.3.2 GetAnalyticsModules* in Analytics Service specification.

The following operation retrieves the currently installed analytics modules.

With

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

Replace the descriptions about Configuration Token under REQUEST parameter part at sections *5.4.3.2 GetAnalyticsModules,*

- **ConfigurationToken [tt:ReferenceToken]**
  Token of the analytics configuration for which the modules should be listed.

With

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

5.4.3.3 CreateAnalyticsModules,

- **ConfigurationToken [tt:ReferenceToken]**
  Token of the analytics configuration for which the modules should be installed.

With

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

5.4.3.4 ModifyAnalyticsModules,

- **ConfigurationToken [tt:ReferenceToken]**
  Token of the analytics configuration for which the modules should be modified.

With

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

5.4.3.5 DeleteAnalyticsModules,

- **ConfigurationToken [tt:ReferenceToken]**
  Token of the analytics configuration for which the modules should be removed.

With

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

And *5.4.3.6 GetAnalyticsModuleOptions*.

- **ConfigurationToken [tt:ReferenceToken]**
  Token of the analytics configuration for which the supported modules should be listed.

With

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

2529 Add RecognitionType structure to types.xsd

Add following "*RecognitionType"* simpleType definition in types.xsd.

```
<xs:simpleType name="RecognitionType">
  <xs:annotation>
    <xs:documentation>Recognition/identification types supported by ONVIF.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="pt:Card">
      <xs:annotation>
        <xs:documentation>A credential number is used for
recognition/identification.</xs:documentation>
      </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="pt:PIN"/>
    <xs:enumeration value="pt:Fingerprint"/>
    <xs:enumeration value="pt:Face"/>
    <xs:enumeration value="pt:Iris"/>
    <xs:enumeration value="pt:Vein"/>
    <xs:enumeration value="pt:Palm"/>
    <xs:enumeration value="pt:Retina"/>
    <xs:enumeration value="pt:LicensePlate"/>
    <xs:enumeration value="pt:REX">
      <xs:annotation>
        <xs:documentation>A request-to-exit button is used for anonymous recognition (but not for
identification).</xs:documentation>
      </xs:annotation>
    </xs:enumeration>
  </xs:restriction>
</xs:simpleType>
```

2535 Add AccessControl/Request/Identifier event to Access Control spec

Add following section in Access Control Service specification

6.8.4 Identifier

Whenever the device requests an external authorization, but the credential token is not known, the device shall send the following event:

Topic: tns1:AccessControl/Request/Identifier

```
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="AccessPointToken" Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="IdentifierType"      Type="xs:string"/>
    <tt:SimpleItemDescription Name="FormatType"          Type="xs:string"/>
    <tt:SimpleItemDescription Name="IdentifierValue"     Type="xs:hexbinary"/>
  </tt:Data>
</tt:MessageDescription>
```

The content of the identifier type string must be one of the ONVIF-specific types defined in pt:RecognitionType (except pt:REX) or a vendor-specific type.
The content of the format type string must be one of the BACnet formats referred to in the Credential Specification (e.g. WIEGAND26) or a vendor-specific format.
The content of the identifier value must contain the credential number, PIN or any other value identifying the credential that is requesting access.

2536 Add AccessControl/AccessGranted/Identifier event to Access Control spec

Add following section in Access Control Service specification

6.3.4 Identifier

Whenever a credential identifier is granted access, but the credential token is not known (e.g. when whitelisted), the device shall provide the following event:

Topic: tns1:AccessControl/AccessGranted/Identifier

```
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="AccessPointToken" Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
```

```
    <tt:SimpleItemDescription Name="IdentifierType"      Type="xs:string"/>
    <tt:SimpleItemDescription Name="FormatType"          Type="xs:string"/>
    <tt:SimpleItemDescription Name="IdentifierValue"     Type="xs:hexbinary"/>
  </tt:Data>
</tt:MessageDescription>
```

The content of the identifier type string must be one of the ONVIF-specific types defined in pt:RecognitionType (except pt:REX) or a vendor-specific type.
The content of the format type string must be one of the BACnet formats referred to in the Credential Specification (e.g. WIEGAND26) or a vendor-specific format.
The content of the identifier value must contain the credential number, PIN or any other value identifying the credential that has been granted access.

2537 Add AccessControl/Denied/Identifier event to Access Control spec

Add following section in Access Control Service specification

6.6.4 Identifier

Whenever a credential identifier is denied access, but the credential token is not known (e.g. when blacklisted), the device shall provide the following event:

```
Topic: tns1:AccessControl/Denied/Identifier
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="AccessPointToken"  Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="IdentifierType"      Type="xs:string"/>
    <tt:SimpleItemDescription Name="FormatType"          Type="xs:string"/>
    <tt:SimpleItemDescription Name="IdentifierValue"     Type="xs:hexbinary"/>
    <tt:SimpleItemDescription Name="Reason"              Type="xs:string"/>
  </tt:Data>
</tt:MessageDescription>
```

The content of the identifier type string must be one of the ONVIF-specific types defined in pt:RecognitionType (except pt:REX) or a vendor-specific type.
The content of the format type string must be one of the BACnet formats referred to in the Credential Specification (e.g. WIEGAND26) or a vendor-specific format.
The content of the identifier value must contain the credential number, PIN or any other value identifying the credential that was denied access.

2538 Add AccessControl/AccessTaken/Identifier event to Access Control spec

Add following section in Access Control Service specification

6.4.4 Identifier

When the device detects that access is taken, but the credential token is not known, it shall provide the following event:

Topic: tns1:AccessControl/AccessTaken/Identifier

```
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="AccessPointToken" Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="IdentifierType"      Type="xs:string"/>
    <tt:SimpleItemDescription Name="FormatType"          Type="xs:string"/>
    <tt:SimpleItemDescription Name="IdentifierValue"     Type="xs:hexbinary"/>
  </tt:Data>
</tt:MessageDescription>
```

The content of the identifier type string must be one of the ONVIF-specific types defined in pt:RecognitionType (except pt:REX) or a vendor-specific type.
The content of the format type string must be one of the BACnet formats referred to in the Credential Specification (e.g. WIEGAND26) or a vendor-specific format.
The content of the identifier value must contain the credential number, PIN or any other value identifying the credential that has taken access.

2539 Add AccessControl/AccessNotTaken/Identifier event to Access Control spec

Add following section in Access Control Service specification

6.5.4 Identifier

When the device detects that access is not taken, but the credential token is not known, it shall provide the following event:

Topic: tns1:AccessControl/AccessNotTaken/Identifier

```
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="AccessPointToken"  Type="pt:ReferenceToken"/>
  </tt:Source>
```

```
  <tt:Data>
    <tt:SimpleItemDescription Name="IdentifierType"      Type="xs:string"/>
    <tt:SimpleItemDescription Name="FormatType"          Type="xs:string"/>
    <tt:SimpleItemDescription Name="IdentifierValue"     Type="xs:hexbinary"/>
  </tt:Data>
</tt:MessageDescription>
```

The content of the identifier type string must be one of the ONVIF-specific types defined in pt:RecognitionType (except pt:REX) or a vendor-specific type.
The content of the format type string must be one of the BACnet formats referred to in the Credential Specification (e.g. WIEGAND26) or a vendor-specific format.
The content of the identifier value must contain the credential number, PIN or any other value identifying the credential that has not taken access.

2540 Remove obsolete section from Access Control spec

Remove a whole section *6.2.2 General transaction event layout* in Access Control Service specification.

2541 Add missing field in event and some editorial changes in Access Control spec

Replace an event example and the following description at section *6.6.5 CredentialNotFound* in Access Control Service specification.

```
Topic: tns1:AccessControl/Denied/CredentialNotFound
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="AccessPointToken"  Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="IdentifierType"      Type="xs:string"/>
    <tt:SimpleItemDescription Name="IdentifierValue"     Type="xs:hexBinary"/>
  </tt:Data>
</tt:MessageDescription>
```

The content of the Identifier type string must be one of the formats listed in the Credential Specification (e.g. pt:Card) or a vendor-specific string.
The content of the Identifier value string must the contain credential number, PIN or any other string identifying the credential that could not be found.

With

```
Topic: tns1:AccessControl/Denied/CredentialNotFound
```

<tt:MessageDescription IsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken" Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="IdentifierType" Type="xs:string"/>
<tt:SimpleItemDescription Name="FormatType" Type="xs:string"/>
<tt:SimpleItemDescription Name="IdentifierValue" Type="xs:hexBinary"/>
</tt:Data>
</tt:MessageDescription>

The content of the identifier type string must be one of the ONVIF-specific types defined in pt:RecognitionType (except pt:REX) or a vendor-specific type.
The content of the format type string must be one of the BACnet formats referred to in the Credential Specification (e.g. WIEGAND26) or a vendor-specific format. This field is optional.
The content of the identifier value must contain the credential number, PIN or any other value identifying the credential that could not be found.

2542 Minor editorial change in Access Control spec

Rephrase description of section *6.8.3 Credential* in Access Control Service specification.

Whenever the device requests external agent to authorize a person, the device shall send the following event:

With

Whenever the device requests an external authorization, and the credential token is specified, the device shall send the following event:

2543 Dependency on overall Video analytics configuration

Remove following annotation sentence from *GetSupportedRules* operation definition in analytics.wsdl.

The result of this method may depend on the overall Video analytics configuration of the device,which is available via the current set of profiles.

2552 Description for certificate / 802.1x methods in devicemgmt.wsdl misleading

Following 18 security operations had been deprecated. Aggregate the definitions and add annotation of the deprecation in devicemgmt.wsdl, as replace

<wsdl:operation name="CreateCertificate">
<wsdl:documentation>This operation generates a private/public key pair and also can create a self-signed device
certificate as a result of key pair generation. The certificate is created using a suitable onboard key pair generation mechanism.
If a device supports onboard key pair generation, the device that supports TLS shall support this certificate creation command. And also, if a device supports onboard key pair generation, the device that support IEEE 802.1X shall support this command for the purpose of key pair generation. Certificates and key pairs are identified using certificate IDs. These IDs are either chosen by the certificate generation requester or by the device (in case that no ID value is given).</wsdl:documentation>
<wsdl:input message="tds:CreateCertificateRequest"/>
<wsdl:output message="tds:CreateCertificateResponse"/>
</wsdl:operation>
<wsdl:operation name="GetCertificates">
<wsdl:documentation>This operation gets all device server certificates (including self-signed) for the purpose of TLS
authentication and all device client certificates for the purpose of IEEE 802.1X authentication. This command lists only the TLS server certificates and IEEE 802.1X client certificates for the device (neither trusted CA certificates nor trusted root certificates). The certificates are returned as binary data. A device that supports TLS shall support this command and the certificates shall be encoded using ASN.1 [X.681], [X.682], [X.683] DER [X.690] encoding rules.</wsdl:documentation>
<wsdl:input message="tds:GetCertificatesRequest"/>
<wsdl:output message="tds:GetCertificatesResponse"/>
</wsdl:operation>
<wsdl:operation name="GetCertificatesStatus">
<wsdl:documentation>This operation is specific to TLS functionality. This operation gets the status
(enabled/disabled) of the device TLS server certificates. A device that supports TLS shall support this command.</wsdl:documentation>
<wsdl:input message="tds:GetCertificatesStatusRequest"/>
<wsdl:output message="tds:GetCertificatesStatusResponse"/>
</wsdl:operation>
<wsdl:operation name="SetCertificatesStatus">
<wsdl:documentation>This operation is specific to TLS functionality. This operation sets the status (enable/disable)
of the device TLS server certificates. A device that supports TLS shall support this command. Typically only one device server certificate is allowed to be enabled at a time.</wsdl:documentation>

<wsdl:input message="tds:SetCertificatesStatusRequest"/>
<wsdl:output message="tds:SetCertificatesStatusResponse"/>
</wsdl:operation>
<wsdl:operation name="DeleteCertificates">
<wsdl:documentation>This operation deletes a certificate or multiple certificates. The device MAY also delete a
private/public key pair which is coupled with the certificate to be deleted. The device that support either TLS or IEEE 802.1X shall support the deletion of a certificate or multiple certificates through this command. Either all certificates are deleted successfully or a fault message shall be returned without deleting any certificate.</wsdl:documentation>
<wsdl:input message="tds:DeleteCertificatesRequest"/>
<wsdl:output message="tds:DeleteCertificatesResponse"/>
</wsdl:operation>
<wsdl:operation name="GetPkcs10Request">
<wsdl:documentation>This operation requests a PKCS #10 certificate signature request from the device. The
returned information field shall be either formatted exactly as specified in [PKCS#10] or PEM encoded [PKCS#10] format. In order for this command to work, the device must already have a private/public key pair. This key pair should be referred by CertificateID as specified in the input parameter description. This CertificateID refers to the key pair generated using CreateCertificate command.
A device that support onboard key pair generation that supports either TLS or IEEE 802.1X using client certificate shall support this command.</wsdl:documentation>
<wsdl:input message="tds:GetPkcs10RequestRequest"/>
<wsdl:output message="tds:GetPkcs10RequestResponse"/>
</wsdl:operation>
<wsdl:operation name="LoadCertificates">
<wsdl:documentation>TLS server certificate(s) or IEEE 802.1X client certificate(s) created using the PKCS#10
certificate request command can be loaded into the device using this command (see Section 8.4.13). The certificate ID in the request shall be present. The device may sort the received certificate(s) based on the public key and subject information in the certificate(s).
The certificate ID in the request will be the ID value the client wish to have. The device is supposed to scan the generated key pairs present in the device to identify which is the correspondent key pair with the loaded certificate and then make the link between the certificate and the key pair.
A device that supports onboard key pair generation that support either TLS or IEEE 802.1X shall support this command.
The certificates shall be encoded using ASN.1 [X.681], [X.682], [X.683] DER [X.690] encoding
rules.

This command is applicable to any device type, although the parameter name is called for historical reasons NVTCertificate.</wsdl:documentation>
<wsdl:input message="tds:LoadCertificatesRequest"/>
<wsdl:output message="tds:LoadCertificatesResponse"/>
</wsdl:operation>
<wsdl:operation name="GetClientCertificateMode">
<wsdl:documentation>This operation is specific to TLS functionality. This operation gets the status
(enabled/disabled) of the device TLS client authentication. A device that supports TLS shall support this command.</wsdl:documentation>
<wsdl:input message="tds:GetClientCertificateModeRequest"/>
<wsdl:output message="tds:GetClientCertificateModeResponse"/>
</wsdl:operation>
<wsdl:operation name="SetClientCertificateMode">
<wsdl:documentation>This operation is specific to TLS functionality. This operation sets the status
(enabled/disabled) of the device TLS client authentication. A device that supports TLS shall support this command.</wsdl:documentation>
<wsdl:input message="tds:SetClientCertificateModeRequest"/>
<wsdl:output message="tds:SetClientCertificateModeResponse"/>
</wsdl:operation>

<wsdl:operation name="GetCACertificates">
<wsdl:documentation>CA certificates will be loaded into a device and be used for the sake of following two cases.
The one is for the purpose of TLS client authentication in TLS server function. The other one is for the purpose of Authentication Server authentication in IEEE 802.1X function. This operation gets all CA certificates loaded into a device. A device that supports either TLS client
authentication or IEEE 802.1X shall support this command and the returned certificates shall be encoded using ASN.1 [X.681], [X.682], [X.683] DER [X.690] encoding rules.</wsdl:documentation>
<wsdl:input message="tds:GetCACertificatesRequest"/>
<wsdl:output message="tds:GetCACertificatesResponse"/>
</wsdl:operation>
<wsdl:operation name="LoadCertificateWithPrivateKey">
<wsdl:documentation>There might be some cases that a Certificate Authority or some other equivalent creates a
certificate without having PKCS#10 certificate signing request. In such cases, the certificate will be bundled in conjunction with its private key. This command will be used for such use case scenarios. The certificate ID in the request is optionally set to the ID value the client

wish to have. If the certificate ID is not specified in the request, device can choose the ID accordingly.
This operation imports a private/public key pair into the device.
The certificates shall be encoded using ASN.1 [X.681], [X.682], [X.683] DER [X.690] encoding
rules.
A device that does not support onboard key pair generation and support either TLS or IEEE 802.1X using client certificate shall support this command. A device that support onboard key pair generation MAY support this command. The security policy of a device that supports this operation should make sure that the private key is sufficiently protected.</wsdl:documentation>
<wsdl:input message="tds:LoadCertificateWithPrivateKeyRequest"/>
<wsdl:output message="tds:LoadCertificateWithPrivateKeyResponse"/>
</wsdl:operation>
<wsdl:operation name="GetCertificateInformation">
<wsdl:documentation>This operation requests the information of a certificate specified by certificate ID. The device
should respond with its “Issuer DN”, “Subject DN”, “Key usage”, "Extended key usage", “Key Length”, “Version”, “Serial Number”, “Signature Algorithm” and “Validity” data as the information of the certificate, as long as the device can retrieve such information from the specified certificate.
A device that supports either TLS or IEEE 802.1X should support this command.</wsdl:documentation>
<wsdl:input message="tds:GetCertificateInformationRequest"/>
<wsdl:output message="tds:GetCertificateInformationResponse"/>
</wsdl:operation>
<wsdl:operation name="LoadCACertificates">
<wsdl:documentation>This command is used when it is necessary to load trusted CA certificates or trusted root
certificates for the purpose of verification for its counterpart i.e. client certificate verification in TLS function or server certificate verification in IEEE 802.1X function.
A device that support either TLS or IEEE 802.1X shall support this command. As for the supported certificate format, either DER format or PEM format is possible to be used. But a device that support this command shall support at least DER format as supported format type.
The device may sort the received certificate(s) based on the public key and subject information in the certificate(s). Either all CA certificates are loaded successfully or a fault message shall be returned without loading any CA certificate.</wsdl:documentation>
<wsdl:input message="tds:LoadCACertificatesRequest"/>
<wsdl:output message="tds:LoadCACertificatesResponse"/>
</wsdl:operation>
<wsdl:operation name="CreateDot1XConfiguration">
<wsdl:documentation>This operation newly creates IEEE 802.1X configuration parameter set of

the device. The
device shall support this command if it supports IEEE 802.1X. If the device receives this request with already existing configuration token (Dot1XConfigurationToken) specification, the
device should respond with 'ter:ReferenceToken ' error to indicate there is some configuration
conflict.</wsdl:documentation>
<wsdl:input message="tds:CreateDot1XConfigurationRequest"/>
<wsdl:output message="tds:CreateDot1XConfigurationResponse"/>
</wsdl:operation>
<wsdl:operation name="SetDot1XConfiguration">
<wsdl:documentation>While the CreateDot1XConfiguration command is trying to create a new configuration
parameter set, this operation modifies existing IEEE 802.1X configuration parameter set of the device. A device that support IEEE 802.1X shall support this command.</wsdl:documentation>
<wsdl:input message="tds:SetDot1XConfigurationRequest"/>
<wsdl:output message="tds:SetDot1XConfigurationResponse"/>
</wsdl:operation>
<wsdl:operation name="GetDot1XConfiguration">
<wsdl:documentation>This operation gets one IEEE 802.1X configuration parameter set from the device by
specifying the configuration token (Dot1XConfigurationToken).
A device that supports IEEE 802.1X shall support this command.
Regardless of whether the 802.1X method in the retrieved configuration has a password or not, the device shall not include the Password element in the response.</wsdl:documentation>
<wsdl:input message="tds:GetDot1XConfigurationRequest"/>
<wsdl:output message="tds:GetDot1XConfigurationResponse"/>
</wsdl:operation>
<wsdl:operation name="GetDot1XConfigurations">
<wsdl:documentation>This operation gets all the existing IEEE 802.1X configuration parameter sets from the device.
The device shall respond with all the IEEE 802.1X configurations so that the client can get to know how many IEEE 802.1X configurations are existing and how they are configured.
A device that support IEEE 802.1X shall support this command.
Regardless of whether the 802.1X method in the retrieved configuration has a password or not, the device shall not include the Password element in the response.</wsdl:documentation>
<wsdl:input message="tds:GetDot1XConfigurationsRequest"/>
<wsdl:output message="tds:GetDot1XConfigurationsResponse"/>
</wsdl:operation>
<wsdl:operation name="DeleteDot1XConfiguration">

<wsdl:documentation>This operation deletes an IEEE 802.1X configuration parameter set from the device. Which
configuration should be deleted is specified by the 'Dot1XConfigurationToken' in the request.
A device that support IEEE 802.1X shall support this command.</wsdl:documentation>
<wsdl:input message="tds:DeleteDot1XConfigurationRequest"/>
<wsdl:output message="tds:DeleteDot1XConfigurationResponse"/>
</wsdl:operation>

With

<wsdl:documentation>The definition and interfaces for the Security have been deprecated with release 16.12.
The Security part was handed over to Security Configuration Service.
For backward compatibility the methods remains in the devicemgmt.wsdl file.
</wsdl:documentation>
<wsdl:operation name="CreateCertificate">
<wsdl:input message="tds:CreateCertificateRequest"/>
<wsdl:output message="tds:CreateCertificateResponse"/>
</wsdl:operation>
<wsdl:operation name="GetCertificates">
<wsdl:input message="tds:GetCertificatesRequest"/>
<wsdl:output message="tds:GetCertificatesResponse"/>
</wsdl:operation>
<wsdl:operation name="GetCertificatesStatus">
<wsdl:input message="tds:GetCertificatesStatusRequest"/>
<wsdl:output message="tds:GetCertificatesStatusResponse"/>
</wsdl:operation>
<wsdl:operation name="SetCertificatesStatus">
<wsdl:input message="tds:SetCertificatesStatusRequest"/>
<wsdl:output message="tds:SetCertificatesStatusResponse"/>
</wsdl:operation>
<wsdl:operation name="DeleteCertificates">
<wsdl:input message="tds:DeleteCertificatesRequest"/>
<wsdl:output message="tds:DeleteCertificatesResponse"/>
</wsdl:operation>
<wsdl:operation name="GetPkcs10Request">
<wsdl:input message="tds:GetPkcs10RequestRequest"/>
<wsdl:output message="tds:GetPkcs10RequestResponse"/>
</wsdl:operation>
<wsdl:operation name="LoadCertificates">
<wsdl:input message="tds:LoadCertificatesRequest"/>
<wsdl:output message="tds:LoadCertificatesResponse"/>
</wsdl:operation>

```
<wsdl:operation name="GetClientCertificateMode">
   <wsdl:input message="tds:GetClientCertificateModeRequest"/>
   <wsdl:output message="tds:GetClientCertificateModeResponse"/>
</wsdl:operation>
<wsdl:operation name="SetClientCertificateMode">
   <wsdl:input message="tds:SetClientCertificateModeRequest"/>
   <wsdl:output message="tds:SetClientCertificateModeResponse"/>
</wsdl:operation>

<wsdl:operation name="GetCACertificates">
   <wsdl:input message="tds:GetCACertificatesRequest"/>
   <wsdl:output message="tds:GetCACertificatesResponse"/>
</wsdl:operation>
<wsdl:operation name="LoadCertificateWithPrivateKey">
   <wsdl:input message="tds:LoadCertificateWithPrivateKeyRequest"/>
   <wsdl:output message="tds:LoadCertificateWithPrivateKeyResponse"/>
</wsdl:operation>
<wsdl:operation name="GetCertificateInformation">
   <wsdl:input message="tds:GetCertificateInformationRequest"/>
   <wsdl:output message="tds:GetCertificateInformationResponse"/>
</wsdl:operation>
<wsdl:operation name="LoadCACertificates">
   <wsdl:input message="tds:LoadCACertificatesRequest"/>
   <wsdl:output message="tds:LoadCACertificatesResponse"/>
</wsdl:operation>
<wsdl:operation name="CreateDot1XConfiguration">
   <wsdl:input message="tds:CreateDot1XConfigurationRequest"/>
   <wsdl:output message="tds:CreateDot1XConfigurationResponse"/>
</wsdl:operation>
<wsdl:operation name="SetDot1XConfiguration">
   <wsdl:input message="tds:SetDot1XConfigurationRequest"/>
   <wsdl:output message="tds:SetDot1XConfigurationResponse"/>
</wsdl:operation>
<wsdl:operation name="GetDot1XConfiguration">
   <wsdl:input message="tds:GetDot1XConfigurationRequest"/>
   <wsdl:output message="tds:GetDot1XConfigurationResponse"/>
</wsdl:operation>
<wsdl:operation name="GetDot1XConfigurations">
   <wsdl:input message="tds:GetDot1XConfigurationsRequest"/>
   <wsdl:output message="tds:GetDot1XConfigurationsResponse"/>
</wsdl:operation>
```

```
<wsdl:operation name="DeleteDot1XConfiguration">
  <wsdl:input message="tds:DeleteDot1XConfigurationRequest"/>
  <wsdl:output message="tds:DeleteDot1XConfigurationResponse"/>
</wsdl:operation>
```

2554 limitation for local token (PACS)

Add an annotation of limitation length for ReferenceToken in pacs.xsd as following underlined sentences.

```
<xs:simpleType name="ReferenceToken">
  <xs:annotation>
    <xs:documentation>Type used to reference logical and physical entities.
    Token may be extended by intermediate terminal with adding prefix to make it global unique.
    The length should be within 36 characters for generating as a local token.
    See "Remote Token" section in Resource Query specification.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:maxLength value="64"/>
    <xs:minLength value="0"/>
  </xs:restriction>
</xs:simpleType>
```

2555 Improve GetSupportedAnalyticsModules annotation

Replace the description of section *5.4.3.1 GetSupportedAnalyticsModules* in Analytics Service specification.

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 5.4.2. 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. Analytics module 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 analytics modules through the maxInstances attribute.

REQUEST:

- **ConfigurationToken [tt:ReferenceToken]**
  Token of the analytics configuration for which the supported modules should be listed.

With

A device signaling support for analytics modules via the AnalyticsModuleSupport capability shall support retrieving a list of analytics modules for an analytics configuration. The description shall conform to the configuration description language as described in section 5.2.
The optional AnalyticsModuleContentSchemaLocation parameter allows to list schema file locations that provide reference to types or elements which are not defined by ONVIF.
The device shall indicate its limit for maximum number of analytics modules through the maxInstances attribute.

REQUEST:

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

2556 GetOptions, Create and DeleteAnalyticsModules

Replace descriptions about analytics module APIs in Analytics Service specification. At section 5.4.3.3 CreateAnalytics Modules,

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

With

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

At section *5.4.3.5 DeleteAnalyticsModules*,

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

With

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

At section *5.4.3.6 GetAnalyticsModuleOptions*,

The following operation returns the options for the supported analytics modules that specify an Option attribute.

With

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.

2557 Active Connections not replaced or deprecated

Add following note at end of section *5.22.2 Active Connections* in Media Service specification, and at end of section *5.12.3 Active Connections* in Media2 Service specification.

NOTE: Active Connections Event is deprecated and its replaced by Active Sessions Event.

2565 Improve analytics module writable capability name

Rephrase following descriptions at section 5.5 Capabilities in Analytics Service specification.

RuleOptionsSupported:
Indication that the device supports the GetRuleOptions operation on the rules interface.
AnalyticsModuleOptionsSupported:
Indication that the device supports the GetAnalyticsModuleOptions operation on the analytics module interface.

With

RuleOptionsSupported:
Indication that the device supports GetRuleOptions and ModifyRules.
AnalyticsModuleOptionsSupported:
Indication that the device supports GetAnalyticsModuleOptions and ModifyAnalyticsModules.

And same rephrase in analytics.wsdl.

```
<xs:complexType name="Capabilities">
  …
  <xs:attribute name="RuleOptionsSupported" type="xs:boolean">
    <xs:annotation>
      <xs:documentation>Indication that the device supports the GetRuleOptions operation on the rules interface</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="AnalyticsModuleOptionsSupported" type="xs:boolean">
    <xs:annotation>
      <xs:documentation>Indication that the device supports the GetAnalyticsModuleOptions operation on the analytics interface</xs:documentation>
    </xs:annotation>
  </xs:attribute>
```

With

```
<xs:complexType name="Capabilities">
  …
  <xs:attribute name="RuleOptionsSupported" type="xs:boolean">
```

```
      <xs:annotation>
        <xs:documentation>Indication that the device supports GetRuleOptions and ModifyRules
</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="AnalyticsModuleOptionsSupported" type="xs:boolean">
      <xs:annotation>
        <xs:documentation>Indication that the device supports GetAnalyticsModuleOptions and
ModifyAnalyticsModules</xs:documentation>
      </xs:annotation>
    </xs:attribute>
```

2566 Requirement of ModifyAnalyticsModules

Replace a description of section *5.4.3.4 ModifyAnalyticsModules* in Analytics Service specification.

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

With

A device signaling support for analytics modules via the AnalyticsModuleOptionsSupported capability shall support this method to modify analytics module configurations.

2567 Rule Operations on analytics modules support condition

Remove the first sentence from the section *5.3.3 Operations on rules* in Analytics Service specification.

If the device supports a Rule Engine as defined by ONVIF, then it shall implement the following operations to manage rules.

Replace the first sentence of the section *5.3.3.1 GetSupportedRules*

The device shall indicate the rules it supports by implementing the subsequent operation.

With

A device signaling support for rules via the RuleSupport capability shall support this operation.

Replace description of the section *5.3.3.2 Get Rules*

The following operation retrieves the currently installed rules:

With

A device signaling support for rules via the RuleSupport capability shall support this operation to retrieve the currently installed rules.

Replace the first sentence of the section *5.3.3.3 CreateRules*

The following operation adds rules to an AnalyticsConfiguration.

With

A device signaling support for rules via the RuleSupport capability shall support this operation to add rules to an AnalyticsConfiguration.

Replace the first sentence of the section *5.3.3.4 ModifyRules*

The following operation modifies multiple rules.

With

A device signaling support for modifying rules via the RuleOptionsSupported capability shall support this operation.

Replace the first sentence of the section *5.3.3.5 DeleteRules*

The following operation deletes multiple rules.

With

A device signaling support for rules via the RuleSupport capability shall support this operation to delete one or more rules.

Replace description of the section *5.3.3.6 GetRuleOptions*

The following operation returns the options for the supported rules that specify an Option attribute.

With

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.

2573 ProfileChanged event and ConfigurationChanged event in Media2 service

Replace descriptions at section *5.12.1 Profile Change* in Media2 Service specification

Whenever a profile is created, deleted or one or more of its configurations are added or removed the following event shall be generated.

With

Whenever a profile is created, deleted or one or more of its configurations are added or removed the following event should be generated.

And at section *5.12.2 Configuration Change*

> Whenever a Configuration of a device changes the device shall provide the following event. For the parameter Type pass the appropriate ConfigurationEnumeration value.

With

> Whenever a Configuration of a device changes the device should provide the following event. For the parameter Type pass the appropriate ConfigurationEnumeration value.

2578 DTT Clarification request regarding CR 2437

Remove a sentence from first paragraph of description at Annex C.1 Motion region detector in Analytics Service specification, as replace

> 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. Omission of the Polygon element indicates an empty region and not a default region. 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.

With

> 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.

2580 Remove unused TopicNamespaceLocation

Remove the following simpleType *TopicNamespaceLocation* definition from onvif.xsd.

```
<xs:simpleType name="TopicNamespaceLocation">
  <xs:restriction base="xs:anyURI"/>
</xs:simpleType>
```

2581 ElementItem in event Source and Key

Add following paragraph after second paragraph of section *9.4.1 Notification Message Format* in ONVIF Core specification.

ElementItem should not be used in the Source and Key elements.

2582 Change FloatList to FloatItems

Change a complexType name “FloatList” to “FloatItems”. It makes replace in onvif.xsd

```
<xs:complexType name="FloatList">
  <xs:sequence>
    <xs:element name="Items" type="xs:float" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

With

```
<xs:complexType name="FloatItems">
  <xs:sequence>
    <xs:element name="Items" type="xs:float" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

Replace in deviceio.wsdl

```
<xs:complexType name="SerialPortConfigurationOptions">
  <xs:annotation>
    <xs:documentation>The configuration options that relates to serial port.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    …
    <xs:element name="StopBitList" type="tt:FloatList">
      <xs:annotation>
        <xs:documentation>The list of configurable number of stop bits used to terminate each
character.</xs:documentation>
      </xs:annotation>
    </xs:element>
```

With

```
<xs:complexType name="SerialPortConfigurationOptions">
  <xs:annotation>
    <xs:documentation>The configuration options that relates to serial port.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    …
    <xs:element name="StopBitList" type="tt:FloatItems">
```

```
        <xs:annotation>
          <xs:documentation>The list of configurable number of stop bits used to terminate each
    character.</xs:documentation>
        </xs:annotation>
    </xs:element>
```

2598 Change IntList to IntItems

Change complexType name “IntList” to “IntItems”.

Replace complexType definition *IntList* in onvif.xsd.

```
    <xs:complexType name="IntList">
      <xs:annotation>
        <xs:documentation>List of values.</xs:documentation>
      </xs:annotation>
```

With

```
    <xs:complexType name="IntItems">
      <xs:annotation>
        <xs:documentation>List of values.</xs:documentation>
      </xs:annotation>
```

Replace complexType definition *RotateOptions*.

```
    <xs:complexType name="RotateOptions">
      <xs:sequence>
        <xs:element name="Mode" type="tt:RotateMode" maxOccurs="unbounded">
          …
        </xs:element>
        <xs:element name="DegreeList" type="tt:IntList" minOccurs="0">
```

With

```
    <xs:complexType name="RotateOptions">
      <xs:sequence>
        <xs:element name="Mode" type="tt:RotateMode" maxOccurs="unbounded">
          …
        </xs:element>
        <xs:element name="DegreeList" type="tt:IntItems" minOccurs="0">
```

Replace complexType definition *AudioEncoderConfigurationOption*.

```
    <xs:complexType name="AudioEncoderConfigurationOption">
```

```
      <xs:sequence>
        <xs:element name="Encoding" type="tt:AudioEncoding">
          …
        </xs:element>
        <xs:element name="BitrateList" type="tt:IntList">
          …
        </xs:element>
        <xs:element name="SampleRateList" type="tt:IntList">
```

With

```
    <xs:complexType name="AudioEncoderConfigurationOption">
      <xs:sequence>
        <xs:element name="Encoding" type="tt:AudioEncoding">
          …
        </xs:element>
        <xs:element name="BitrateList" type="tt:IntItems">
          …
        </xs:element>
        <xs:element name="SampleRateList" type="tt:IntItems">
```

Replace complexType definition *AudioEncoder2ConfigurationOptions*.

```
    <xs:complexType name="AudioEncoder2ConfigurationOptions">
      <xs:sequence>
        <xs:element name="Encoding" type="xs:string">
          …
        </xs:element>
        <xs:element name="BitrateList" type="tt:IntList">
          …
        </xs:element>
        <xs:element name="SampleRateList" type="tt:IntList">
```

With

```
    <xs:complexType name="AudioEncoder2ConfigurationOptions">
      <xs:sequence>
        <xs:element name="Encoding" type="xs:string">
          …
        </xs:element>
        <xs:element name="BitrateList" type="tt:IntItems">
          …
        </xs:element>
        <xs:element name="SampleRateList" type="tt:IntItems">
```

Replace complexType definition *G711DecOptions*.

```
<xs:complexType name="G711DecOptions">
  <xs:sequence>
    <xs:element name="Bitrate" type="tt:IntList">
      …
    </xs:element>
    <xs:element name="SampleRateRange" type="tt:IntList">
```

With

```
<xs:complexType name="G711DecOptions">
  <xs:sequence>
    <xs:element name="Bitrate" type="tt:IntItems">
      …
    </xs:element>
    <xs:element name="SampleRateRange" type="tt:IntItems">
```

Replace complexType definition *AACDecOptions*.

```
<xs:complexType name="AACDecOptions">
  <xs:sequence>
    <xs:element name="Bitrate" type="tt:IntList">
      …
    </xs:element>
    <xs:element name="SampleRateRange" type="tt:IntList">
```

With

```
<xs:complexType name="AACDecOptions">
  <xs:sequence>
    <xs:element name="Bitrate" type="tt:IntItems">
      …
    </xs:element>
    <xs:element name="SampleRateRange" type="tt:IntItems">
```

Replace complexType definition G726*DecOptions*.

```
<xs:complexType name="G726DecOptions">
  <xs:sequence>
    <xs:element name="Bitrate" type="tt:IntList">
      …
    </xs:element>
    <xs:element name="SampleRateRange" type="tt:IntList">
```

With

```
<xs:complexType name="G726DecOptions">
  <xs:sequence>
    <xs:element name="Bitrate" type="tt:IntItems">
      …
    </xs:element>
    <xs:element name="SampleRateRange" type="tt:IntItems">
```

Replace complexType definition *SerialPortConfigurationOptions* in deviceio.wsdl.

```
<xs:complexType name="SerialPortConfigurationOptions">
  <xs:annotation>
    <xs:documentation>The configuration options that relates to serial port.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="BaudRateList" type="tt:IntList">
      …
    </xs:element>
    <xs:element name="ParityBitList" type="tmd:ParityBitList">
      …
    </xs:element>
    <xs:element name="CharacterLengthList" type="tt:IntList">
```

With

```
<xs:complexType name="SerialPortConfigurationOptions">
  <xs:annotation>
    <xs:documentation>The configuration options that relates to serial port.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="BaudRateList" type="tt:IntItems">
      …
    </xs:element>
    <xs:element name="ParityBitList" type="tmd:ParityBitList">
      …
    </xs:element>
    <xs:element name="CharacterLengthList" type="tt:IntItems">
```

Replace data type description example at section *5.23.11 RotateOptions* in Media Service specification.

```
<xs:complexType name="RotateOptions">
  <xs:element name="Mode" type="tt:RotateMode" maxOccurs="unbounded"/>
  <xs:element name="DegreeList" type="tt:IntList" minOccurs="0"/>
</xs:complexType>
```

With

```
<xs:complexType name="RotateOptions">
   <xs:element name="Mode" type="tt:RotateMode" maxOccurs="unbounded"/>
   <xs:element name="DegreeList" type="tt:IntItems" minOccurs="0"/>
</xs:complexType>
```

Replace data type description example at section *5.23.29 AudioEncoderConfigurationOption*.

```
<xs:complexType name="AudioEncoderConfigurationOption">
  <xs:element name="Encoding" type="tt:AudioEncoding"/>
  <xs:element name="BitrateList" type= "tt:IntList"/>
  <xs:element name="SampleRateList" type= "tt:IntList"/>
</xs:complexType>
```

With

```
<xs:complexType name="AudioEncoderConfigurationOption">
  <xs:element name="Encoding" type="tt:AudioEncoding"/>
  <xs:element name="BitrateList" type= "tt:IntItems"/>
  <xs:element name="SampleRateList" type= "tt:IntItems"/>
</xs:complexType>
```

Replace data type description example at section *5.23.48 G711DecOptions*.

```
<xs:complexType name="G711DecOptions">
  <xs:element name="Bitrate" type= "tt:IntList"/>
  <xs:element name="SampleRateRange" type= "tt:IntList"/>
</xs:complexType>
```

With

```
<xs:complexType name="G711DecOptions">
  <xs:element name="Bitrate" type= "tt:IntItems"/>
  <xs:element name="SampleRateRange" type= "tt:IntItems"/>
</xs:complexType>
```

Replace data type description example at section *5.23.49 AACDecOptions*.

```
<xs:complexType name="AACDecOptions">
  <xs:element name="Bitrate" type= "tt:IntList"/>
  <xs:element name="SampleRateRange" type= "tt:IntList"/>
</xs:complexType>
```

With

```
<xs:complexType name="AACDecOptions">
  <xs:element name="Bitrate" type= "tt:IntItems"/>
```

```
  <xs:element name="SampleRateRange" type= "tt:IntItems"/>
</xs:complexType>
```

Replace data type description example at section *5.23.50 G726DecOptions*.

```
<xs:complexType name="G726DecOptions">
  <xs:element name="Bitrate" type= "tt:IntList"/>
  <xs:element name="SampleRateRange" type= "tt:IntList"/>
</xs:complexType>
```

With

```
<xs:complexType name="G726DecOptions">
  <xs:element name="Bitrate" type= "tt:IntItems"/>
  <xs:element name="SampleRateRange" type= "tt:IntItems"/>
</xs:complexType>
```

2603 Align AddMetadata error behavior

Remove the following second paragraph from the section *5.2.7 Analytics Configuration* in Media2 Service specification.

> A device shall return an error (ter:NoConfig) if a client attempts to add VideoAnalyticsConfiguration to the Profile when the required source configuration (e.g., VideoSourceConfiguration) is not present.

2607 Define a new SimpleType "OtherClassType" to specify a more generic "ClassType"

Add a new simpleType *ObjectType* in metadatastream.xsd.

```
<xs:simpleType name="ObjectType">
   <xs:restriction base="xs:string">
      <xs:enumeration value="Animal"/>
      <xs:enumeration value="HumanFace"/>
      <xs:enumeration value="Human"/>
      <xs:enumeration value="Bicycle"/>
      <xs:enumeration value="Vehicle"/>
      <xs:enumeration value="LicensePlate"/>
   </xs:restriction>
</xs:simpleType>
```

And add an annotation sentence under complexType *ClassDescriptor/Type*, as replace

```
<xs:complexType name="ClassDescriptor">
  <xs:sequence>
    <xs:element name="ClassCandidate" minOccurs="0" maxOccurs="unbounded">
      …
    </xs:element>
    <xs:element name="Extension" type="tt:ClassDescriptorExtension" minOccurs="0"/>
    <xs:element name="Type" type="tt:StringLikelihood" minOccurs="0"
maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>ONVIF recommends to use this 'Type' element instead of
'ClassCandidate' and 'Extension' above for new design.</xs:documentation>
      </xs:annotation>
```

With

```
<xs:complexType name="ClassDescriptor">
  <xs:sequence>
    <xs:element name="ClassCandidate" minOccurs="0" maxOccurs="unbounded">
      …
    </xs:element>
    <xs:element name="Extension" type="tt:ClassDescriptorExtension" minOccurs="0"/>
    <xs:element name="Type" type="tt:StringLikelihood" minOccurs="0"
maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>ONVIF recommends to use this 'Type' element instead of
'ClassCandidate' and 'Extension' above for new design. Acceptable values are defined in
tt:ObjectType.</xs:documentation>
      </xs:annotation>
```

2611 Add StringList to types.xsd and make accesscontrol.wsdl use it

Add a *simpleType* “StringList” definition in types.xsd

```
<xs:simpleType name="StringList">
  <xs:list itemType="xs:string"/>
</xs:simpleType>
```

And replace an *attribute* “SupportedRecognitionTypes” type under “AccessPointCapabilities” *complexType* in accesscontrol.wsdl.

```
<xs:complexType name="AccessPointCapabilities">
```

```
<xs:annotation>
  …
<xs:sequence>
  …
<xs:attribute name="DisableAccessPoint" type="xs:boolean" use="required">
  …
<xs:attribute name="Duress" type="xs:boolean">
  …
<xs:attribute name="AnonymousAccess" type="xs:boolean">
  …
<xs:attribute name="AccessTaken" type="xs:boolean">
  …
<xs:attribute name="ExternalAuthorization" type="xs:boolean">
  …
<xs:attribute name="SupportedRecognitionTypes" type="tt:StringAttrList">
```

With

```
<xs:complexType name="AccessPointCapabilities">
<xs:annotation>
  …
<xs:sequence>
  …
<xs:attribute name="DisableAccessPoint" type="xs:boolean" use="required">
  …
<xs:attribute name="Duress" type="xs:boolean">

  …
<xs:attribute name="AnonymousAccess" type="xs:boolean">
  …
<xs:attribute name="AccessTaken" type="xs:boolean">
  …
<xs:attribute name="ExternalAuthorization" type="xs:boolean">
  …
<xs:attribute name="SupportedRecognitionTypes" type="pt:StringList">
```

2613 ParentTopics in get supported rules and analytics modules

Replace annotation of “ParentTopic” under in onvif.xsd.

```
<xs:complexType name="ConfigDescription">
  <xs:sequence>
    <xs:element name="Parameters" type="tt:ItemListDescription">
```

```
        …
      </xs:element>
      <xs:element name="Messages" minOccurs="0" maxOccurs="unbounded">
       …
       <xs:complexType>
        <xs:complexContent>
         <xs:extension base="tt:MessageDescription">
          <xs:sequence>
           <xs:element name="ParentTopic" type="xs:string">
            <xs:annotation>
             <xs:documentation>
              The ParentTopic labels the message (e.g. "nn:RuleEngine/LineCrossing"). The real message can extend the ParentTopic
              by for example the name of the instaniated rule (e.g. "nn:RuleEngine/LineCrossing/corssMyFirstLine").
              Even without knowing the complete topic name, the subscriber will be able to distiguish the
              messages produced by different rule instances of the same type via the Source fields of the message.
              There the name of the rule instance, which produced the message, must be listed.
             </xs:documentation>
```

With

```
   <xs:complexType name="ConfigDescription">
    <xs:sequence>
     <xs:element name="Parameters" type="tt:ItemListDescription">
      …
     </xs:element>
     <xs:element name="Messages" minOccurs="0" maxOccurs="unbounded">
      …
      <xs:complexType>
       <xs:complexContent>
        <xs:extension base="tt:MessageDescription">
         <xs:sequence>
          <xs:element name="ParentTopic" type="xs:string">
           <xs:annotation>
            <xs:documentation>
             The topic of the message. For historical reason the element is named ParentTopic, but the full topic is expected.
            </xs:documentation>
```

2614 simpleType "StringList"

Add a *simpleType* “StringList” definition in onvif.xsd

```
<xs:simpleType name="StringList">
  <xs:list itemType="xs:string"/>
</xs:simpleType>
```

And replace an *element* “StringList” type

```
<xs:element name="StringList" type="tt:StringAttrList"/>
```

With

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

2628 [Analytics Rules] ConfigurationToken description for requests clarification

Replace descriptions of “ConfigurationToken” under *REQUEST* parameter at section *5.3.3.1 GetSupportedRules* in Analytics Service specification.

REQUEST:

• ConfigurationToken [tt:ReferenceToken]

Token of the analytics configuration for which the supported rules should be listed.

With

REQUEST:

• ConfigurationToken [tt:ReferenceToken]

Token of an existing analytics configuration.

At section *5.3.3.2 Get Rules*,

REQUEST:

• ConfigurationToken [tt:ReferenceToken]

Token of the analytics configuration for which the rules should be listed.

With

REQUEST:

• ConfigurationToken [tt:ReferenceToken]

Token of an existing analytics configuration.

At section *5.3.3.3 CreateRules*,

REQUEST:

• ConfigurationToken [tt:ReferenceToken]

Token of the analytics configuration for which the rules should be installed.

With

REQUEST:

• ConfigurationToken [tt:ReferenceToken]

Token of an existing analytics configuration.

At section *5.3.3.4 ModifyRules*,

REQUEST:

• ConfigurationToken [tt:ReferenceToken]

Token of the analytics configuration for which the rules should be modified.

With

REQUEST:

• ConfigurationToken [tt:ReferenceToken]

Token of an existing analytics configuration.

At section *5.3.3.5 DeleteRules*,

REQUEST:

• ConfigurationToken [tt:ReferenceToken]

Token of the analytics configuration for which the rules should be removed.

With

REQUEST:

• ConfigurationToken [tt:ReferenceToken]

Token of an existing analytics configuration.

At section *5.3.3.6 GetRuleOptions*,

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 the analytics configuration for which the supported rules should be listed.

With

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.

2629 Typo in §9.9 of the ONVIF Core specs

A minor editorial change.

2632 Editorial change in Access Control spec

Replace a description at section *6.3.3 Credential* in Access Control Service specification.

Whenever a valid credential has passed all the necessary checks and a user or card holder is granted access to the access point, but not yet accessed it (entered or exited), the device shall provide the following event data:

With

Whenever a valid credential has passed all the necessary checks and a credential holder is granted access to the access point, but not yet accessed it (entered or exited), the device shall provide the following event data:

2633 Make sure that new Identifier events is not breaking backwards compatibility

Add a following “IdentifierAccess” *attribute* under “AccessPointCapabilities” *complexType* in accesscontrol.wsdl.

```
<xs:attribute name="IdentiferAccess" type="xs:boolean">
  <xs:annotation>
    <xs:documentation>
      Indicates whether or not this access point supports the AccessControl/Request/Identifier
      event to request external authorization.
      Identfier access requires that ExternalAuthorization is set to true.
      The IdentifierAccess capability is typically enabled for devices that do not have any
      knowledge of credential tokens. When IdentifierAccess is set to true then the device
      shall support the identifier events.
    </xs:documentation>
  </xs:annotation>
</xs:attribute>
```

And add the description at the end of section *5.2.1.3 AccessPointCapabilities* in Access Control Service specification.

• IdentifierAccess

Indicates whether or not this access point supports the AccessControl/Request/Identifier event to request external authorization.

Identfier access requires that ExternalAuthorization is set to true.

The IdentifierAccess capability is typically enabled for devices that do not have any knowledge of credential tokens. When IdentifierAccess is set to true then the device shall support the identifier events.

Replace a description at section *6.4.3 Credential*.

When the device detects that access is taken and the credential can be identified, it shall provide the following event:

With

When the device detects that access is taken and the credential can be identified (credential token is known), it shall provide the following event:

Replace a description at section *6.4.4 Identifier*.

When the device detects that access is taken, but the credential token is not known, it shall provide the following event:

With

When the device detects that access is taken, and the credential identifier is known but the credential token is not known, it shall provide the following event:

Replace a description at section *6.5.3 Credential*.

When the device detects that access is not taken and the credential can be identified, it shall provide the following event:

With

When the device detects that access is not taken and the credential can be identified (credential token is known), it shall provide the following event:

Replace a description at section *6.5.4 Identifier*.

When the device detects that access is not taken, but the credential token is not known, it shall provide the following event:

With

When the device detects that access is not taken, and the credential identifier is known but the credential token is not known, it shall provide the following event:

Replace a description at section *6.6.3 Credential*.

When the device denies access and the credential can be identified, it shall provide the following event:

With

When the device denies access and the credential can be identified (credential token is known), it shall provide the following event:

Replace first sentence of a description at section *6.8.1 General*.

A device that signals support for ExternalAuthorization capability for a particular access point instance shall provide events defined in this section whenever it requests for external authorization.

With

A device that signals support for ExternalAuthorization capability for a particular access point instance shall provide applicable events defined in this section whenever it requests for external authorization.

Replace a description at section *6.8.3 Credential*.

Whenever the device requests an external authorization, and the credential token is specified, the device shall send the following event:

With

Whenever the device requests an external authorization, and the credential token is known, the device shall send the following event:

Replace a description at section *6.8.4 Identifier*.

Whenever the device requests an external authorization, but the credential token is not known, the device shall send the following event:

With

If the IdentifierAccess capability is set to true for the access point, then whenever the device requests an external authorization, and the credential identifier is known but the credential token is not known, the device shall send the following event:

2634 Change xs:hexbinary to xs:hexBinary in Access Control spec

Correct data type of “IdentifierValue” *SimpleItemDescription* in Identifier event message description at section *6.3.4 Identifier, 6.4.4 Identifier, 6.5.4 Identifier, 6.6.4 Identifier*, and *6.8.4 Identifier* in Access Control Service specification, as replace

Topic: tns1:AccessControl/***(difference by section)***/Identifier

```
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="AccessPointToken"
```

```
                                    Type="pt:ReferenceToken"/>
    </tt:Source>
    <tt:Data>
      <tt:SimpleItemDescription Name="IdentifierType"
                                    Type="xs:string"/>
      <tt:SimpleItemDescription Name="FormatType"
                                    Type="xs:string"/>
      <tt:SimpleItemDescription Name="IdentifierValue"
                                    Type="xs:hexbinary"/>
    </tt:Data>
  </tt:MessageDescription>
```

With

Topic: tns1:AccessControl/***(difference by section)***/Identifier

```
  <tt:MessageDescription IsProperty="false">
    <tt:Source>
      <tt:SimpleItemDescription Name="AccessPointToken"
                                    Type="pt:ReferenceToken"/>
    </tt:Source>
    <tt:Data>
      <tt:SimpleItemDescription Name="IdentifierType"
                                    Type="xs:string"/>
      <tt:SimpleItemDescription Name="FormatType"
                                    Type="xs:string"/>
      <tt:SimpleItemDescription Name="IdentifierValue"
                                    Type="xs:hexBinary"/>
    </tt:Data>
  </tt:MessageDescription>
```

2635 Editorial clarification in Door Control Spec

Replace the sixth paragraph of section *5.5.2 AccessDoor command* in Door Control Service specification.

A device that signals support for Access capability for a particular Door instance shall support this method. A device that signals support for AccessTimingOverride capability for a particular Door instance shall also provide the following optional timing parameters when performing AccessDoor command:

With

A device that signals support for Access capability for a particular Door instance shall support this method. A device that signals support for AccessTimingOverride capability

for a particular Door instance shall also support the following optional timing parameters when performing AccessDoor command:

2636 Request to extend the error code for Get<entity>ConfigurationOptions

Add following fault code under *FAULT* at section *5.3.4 Get<entity>ConfigurationOptions* in Media2 Service specification.

- env:Sender - ter:InvalidArgVal - ter:IncompatibleConfiguration
  The requested configuration is not compatible.

2637 Make MaxCredentials xs:unsignedInt instead of pt:PositiveInteger

Add a sentence into “MaxCredentials” description at section *5.2.2.1 ServiceCapabilities* in Credential Service specification.

- MaxCredentials
  The maximum number of credential supported by the device.

With

- MaxCredentials
  The maximum number of credential supported by the device. If set to 0, then operations using credential token are not supported.

And add annotation sentence into “MaxCredentials” under “ServiceCapabilities” *complexType* in credential.wsdl.

```
<xs:complexType name="ServiceCapabilities">
  <xs:annotation>
    …
  <xs:sequence>
  …
  <xs:attribute name="MaxLimit" type="pt:PositiveInteger" use="required">
    …
  <xs:attribute name="CredentialValiditySupported" type="xs:boolean" use="required">
    …
  <xs:attribute name="CredentialAccessProfileValiditySupported" type="xs:boolean"
use="required">
    …
  <xs:attribute name="ValiditySupportsTimeValue" type="xs:boolean" use="required">
    …
```

```
<xs:attribute name="MaxCredentials" type="pt:PositiveInteger" use="required">
 <xs:annotation>
  <xs:documentation>
   The maximum number of credential supported by the device.
  </xs:documentation>
```

With

```
<xs:complexType name="ServiceCapabilities">
 <xs:annotation>
  …
 <xs:sequence>
 …
 <xs:attribute name="MaxLimit" type="pt:PositiveInteger" use="required">
  …
 <xs:attribute name="CredentialValiditySupported" type="xs:boolean" use="required">
  …
 <xs:attribute name="CredentialAccessProfileValiditySupported" type="xs:boolean"
use="required">
  …
 <xs:attribute name="ValiditySupportsTimeValue" type="xs:boolean" use="required">
  …
 <xs:attribute name="MaxCredentials" type="pt:PositiveInteger" use="required">
  <xs:annotation>
   <xs:documentation>
    The maximum number of credential supported by the device.
    If set to 0, then operations using credential token are not supported.
   </xs:documentation>
```

2638 Add documentation to enums in PACS types.xsd

Replace a *simpleType* “RecognitionType” definition in types.xsd

```
<xs:simpleType name="RecognitionType">
 <xs:annotation>
  <xs:documentation>Recognition/identification types supported by ONVIF.</xs:documentation>
 </xs:annotation>
 <xs:restriction base="xs:string">
  <xs:enumeration value="pt:Card">
   <xs:annotation>
    <xs:documentation>A credential number is used for
recognition/identification.</xs:documentation>
   </xs:annotation>
```

```
    </xs:enumeration>
    <xs:enumeration value="pt:PIN"/>
    <xs:enumeration value="pt:Fingerprint"/>
    <xs:enumeration value="pt:Face"/>
    <xs:enumeration value="pt:Iris"/>
    <xs:enumeration value="pt:Vein"/>
    <xs:enumeration value="pt:Palm"/>
    <xs:enumeration value="pt:Retina"/>
    <xs:enumeration value="pt:LicensePlate"/>
    <xs:enumeration value="pt:REX">
     <xs:annotation>
      <xs:documentation>A request-to-exit button is used for anonymous recognition (but not for identification).</xs:documentation>
     </xs:annotation>
    </xs:enumeration>
   </xs:restriction>
  </xs:simpleType>
```

With

```
  <xs:simpleType name="RecognitionType">
   <xs:annotation>
    <xs:documentation>Recognition/identification types supported by ONVIF.</xs:documentation>
   </xs:annotation>
   <xs:restriction base="xs:string">
    <xs:enumeration value="pt:Card">
     <xs:annotation>
      <xs:documentation>A credential number is used for recognition/identification.</xs:documentation>
     </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="pt:PIN">
     <xs:annotation>
      <xs:documentation>A PIN code is used for recognition/identification.</xs:documentation>
     </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="pt:Fingerprint">
     <xs:annotation>
      <xs:documentation>A fingerprint is used for recognition/identification.</xs:documentation>
     </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="pt:Face">
     <xs:annotation>
```

```
      <xs:documentation>A face is used for recognition/identification.</xs:documentation>
    </xs:annotation>
   </xs:enumeration>
   <xs:enumeration value="pt:Iris">
    <xs:annotation>
      <xs:documentation>An iris is used for recognition/identification.</xs:documentation>
    </xs:annotation>
   </xs:enumeration>
   <xs:enumeration value="pt:Vein">
    <xs:annotation>
      <xs:documentation>A vein is used for recognition/identification.</xs:documentation>
    </xs:annotation>
   </xs:enumeration>
   <xs:enumeration value="pt:Palm">
    <xs:annotation>
      <xs:documentation>A palm is used for recognition/identification.</xs:documentation>
    </xs:annotation>
   </xs:enumeration>
   <xs:enumeration value="pt:Retina">
    <xs:annotation>
      <xs:documentation>A retina is used for recognition/identification.</xs:documentation>
    </xs:annotation>
   </xs:enumeration>
   <xs:enumeration value="pt:LicensePlate">
    <xs:annotation>
      <xs:documentation>A license plate is used for recognition/identification.</xs:documentation>
    </xs:annotation>
   </xs:enumeration>    <xs:enumeration value="pt:REX">
    <xs:annotation>
      <xs:documentation>
        A request-to-exit button is used for anonymous recognition (but not for identification).
      </xs:documentation>
    </xs:annotation>
   </xs:enumeration>
  </xs:restriction>
</xs:simpleType>
```

2647 In PACS specs: Change tt:StringAttrList to pt:StringList

Remove a namespace declaration in authenticationbehavior.wsdl

```
<xs:schema targetNamespace="http://www.onvif.org/ver10/authenticationbehavior/wsdl"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:pt="http://www.onvif.org/ver10/pacs"
  xmlns:tt="http://www.onvif.org/ver10/schema"
  xmlns:tab="http://www.onvif.org/ver10/authenticationbehavior/wsdl"
  elementFormDefault="qualified"
  version="19.12">
```

with

```
<xs:schema targetNamespace="http://www.onvif.org/ver10/authenticationbehavior/wsdl"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:pt="http://www.onvif.org/ver10/pacs"
  xmlns:tab="http://www.onvif.org/ver10/authenticationbehavior/wsdl"
  elementFormDefault="qualified"
  version="19.12">
```

And remove following import declaration

```
<xs:import namespace="http://www.onvif.org/ver10/schema"
schemaLocation="../../schema/onvif.xsd"/>
```

Replace an *attribute* “SupportedAuthenticationModes” type under *complexType* “ServiceCapabilities”.

```
<xs:complexType name="ServiceCapabilities">
  <xs:annotation>
    …
  <xs:sequence>
    …
  <xs:attribute name="MaxLimit" type="pt:PositiveInteger" use="required">
    …
  <xs:attribute name="MaxAuthenticationProfiles" type="pt:PositiveInteger" use="required">
    …
  <xs:attribute name="MaxPoliciesPerAuthenticationProfile" type="pt:PositiveInteger"
use="required">
    …
  <xs:attribute name="MaxSecurityLevels" type="pt:PositiveInteger" use="required">
    …
  <xs:attribute name="MaxRecognitionGroupsPerSecurityLevel" type="pt:PositiveInteger"
use="required">
    …
  <xs:attribute name="MaxRecognitionMethodsPerRecognitionGroup" type="pt:PositiveInteger"
use="required">
```

…
<xs:attribute name="ClientSuppliedTokenSupported" type="xs:boolean" default="false">
…
<xs:attribute name="SupportedAuthenticationModes" type="tt:StringAttrList">

With

<xs:complexType name="ServiceCapabilities">
<xs:annotation>
…
<xs:sequence>
…
<xs:attribute name="MaxLimit" type="pt:PositiveInteger" use="required">
…
<xs:attribute name="MaxAuthenticationProfiles" type="pt:PositiveInteger" use="required">
…
<xs:attribute name="MaxPoliciesPerAuthenticationProfile" type="pt:PositiveInteger" use="required">
…
<xs:attribute name="MaxSecurityLevels" type="pt:PositiveInteger" use="required">
…
<xs:attribute name="MaxRecognitionGroupsPerSecurityLevel" type="pt:PositiveInteger" use="required">
…
<xs:attribute name="MaxRecognitionMethodsPerRecognitionGroup" type="pt:PositiveInteger" use="required">
…
<xs:attribute name="ClientSuppliedTokenSupported" type="xs:boolean" default="false">
…
<xs:attribute name="SupportedAuthenticationModes" type="pt:StringList">

3.1. Careful notice

Following change requests might cause influence upon consistency between previous implementations.

2582
2598
2637