ONVIF™
Action Engine Service Specification

Version 1.0
May, 2012

CONTENTS

1 Scope

This document defines the web service interface for configuration of the Actions and Action Trigger conditions based on events.

2 Normative references

ONVIF™ Core Specification, Version 2.2, May, 2012
<http://www.onvif.org/specs/core/ONVIF-Core-Specification-v220.pdf>

ONVIF™ Recording Control Service Specification, Version 2.2, May, 2012.
<http://www.onvif.org/specs/srv/rec/ONVIF-RecordingControl-Service-Spec-v220.pdf>

3 Terms and Definitions

3.1 Definitions

Action Trigger	Condition. Topic and message content filter based condition
Action	Executed when one of its conditions (Action Trigger) is satisfied

3.2 Abbreviations

FTP	File Transfer Protocol
E-Mail	Electronic Mail
SMTP	Simple Mail Transfer Protocol
POP	Post Office Protocol
SMS	Short Message Service

3.3 Namespaces

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

Table 1: Namespaces used in this specification

Prefix	Namespace URI	Description
tt	http://www.onvif.org/ver10/schema	XML schema descriptions in this specification.
tae	http://www.onvif.org/ver10/actionengine/wsdl	The namespace for the WSDL action engine service.
ter	http://www.onvif.org/ver10/error	The namespace for ONVIF defined faults.
tns1	http://www.onvif.org/ver10/topics	The namespace for the ONVIF topic namespace

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

Table 2: Referenced namespaces

Prefix	Namespace URI	Description
soapenv	http://www.w3.org/2003/05/soap-envelope	Envelope namespace as defined by SOAP 1.2 [SOAP 1.2, Part 1]
xs	http://www.w3.org/2001/XMLSchema	Instance namespace as defined by XS [XML-Schema, Part1] and [XML-Schema, Part 2]

4 Overview

This document describes Action Engine web service depicted also in video analytics architecture (Figure 6) in [Video Analytics Service Specification]. The Action Engine service enables new extensible device capabilities by providing further integration opportunities with other networked devices through actions that are triggered by events.

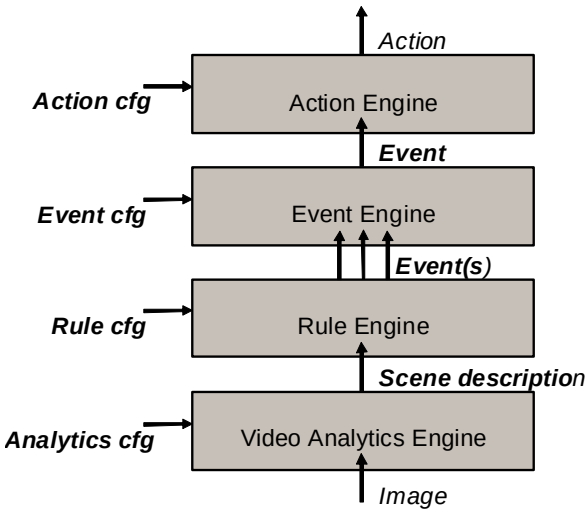

Figure 1 Video Analytics Architecture

Action Engine service interface allows service requester to list available action types (defined in action description language), creates new actions, modify the action configurations, and delete actions.

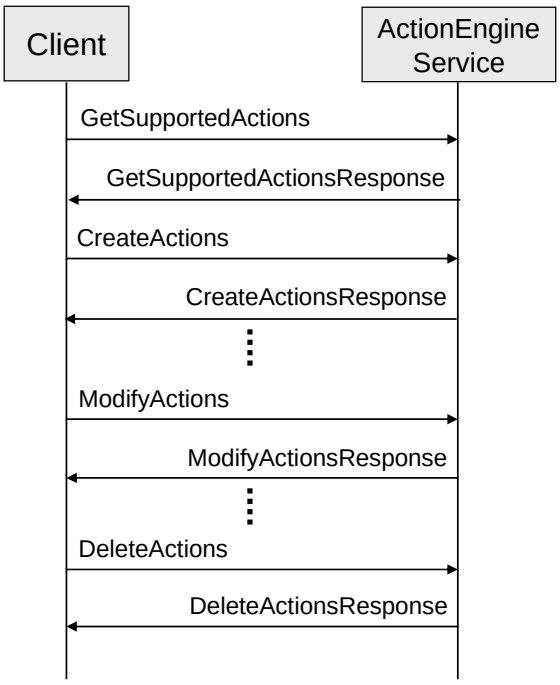

Figure 2 Action creation and configuration sequence

Action Engine interface also allows service requester to define the action triggers that are defined based on Topic expression (Section Topic Filter in [Core Specification]) and Message Content expression (Section Message Content Filter in [Core Specification]). Service requester can list, create, modify, and delete the action triggers. Action Engine service provider passes the triggering event information to triggered action instances.

5 Service

5.1 Action representation

The configuration of an action contains the action name and the other Action type specific parameter configurations. The Parameters element of the Action element contains the configurations of action parameters. Each Parameter is defined by either a SimpleItem or an ElementItem data type. The Name attribute of each parameter shall be unique within the parameter list. The Value attribute of SimpleItem contains the parameter configuration. The child element of an ElementItem contains the configured value of an ElementItem. It is RECOMMENDED to represent as many configuration parameters as possible by SimpleItems.

5.2 Action description language

The description of an Action contains the action type name (Name) and the type information of all parameters belonging to the defined Action type. The parameters of a certain Action Type are listed below the ParameterDescription element. All parameters are either Simple or ElementItems and can be described by either a SimpleItemDescription or an ElementItemDescription. Both ItemDescriptions contain a Name attribute to identify the parameter and a Type attribute to reference a specific XML schema type. The Type attribute of the SimpleItemDescription shall reference to XML schema simple type definition. The Type attribute of the ElementItemDescription shall reference a global element declaration of an XML schema.

Section 5.4 demonstrates the usage of the Action Description Language. The following definitions are included for convenience:

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

<xs:complexType name="ActionConfigDescription">
  <xs:sequence>
    <xs:element name="ParameterDescription"
                type="tt:ItemListDescription"/>
  </xs:sequence>
  <xs:attribute name="Name" type="xs:QName" use="required"/>
  <xs:anyAttribute processContents="lax"/>
</xs:complexType>
```

5.3 Operations on actions

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

5.3.1 GetSupportedActions

The service provider returns the supported action types. The response returns a list of Action Descriptions according to the Action Description Language described in Section 5.2. The response also contains a list of URLs that provide the location of the schema files. These schema files describe the types and elements used in the Action Descriptions. If action descriptions reference types or elements of the ONVIF schema file, the ONVIF schema file shall be explicitly listed.

Table 3: GetSupportedActions command

GetSupportedActions	Access Class:READ_SYSTEM
Message name	**Description**
GetSupportedActionsRequest	*This is an empty message.*
GetSupportedActionsResponse	*The response contains the supported actions.* tt:SupportedActions **SupportedActions** [1][1]
Fault codes	**Description**
	There are no operation specific faults.

5.3.2 GetActions

The get actions operation retrieves currently installed Actions.

Table 4: GetActions command

GetActions	Access Class:READ_SYSTEM
Message name	**Description**
GetActionsRequest	*This is an empty message.*
GetActionsResponse	*The response is a list of installed actions for the specified configuration.* tt:Action **Action** [0][unbounded]
Fault codes	**Description**
	There are no operation specific faults.

5.3.3 CreateActions

The create action operation adds actions to configuration. The create action operation is atomic. If a service provider can not create all of requested actions, then, the service provider responds with a fault message.

Table 5: CreateActions command

CreateActions	Access Class:WRITE_SYSTEM
Message name	**Description**
CreateActionsRequest	*The request message specifies list of Actions to be added*. tt:ActionConfiguration **Action** [1][unbounded]
CreateActionsResponse	*The response is a list of created actions.* tt:Action **Action** [0][unbounded].
Fault codes	**Description**
env:Sender ter:InvalidArgVal	*The requested actions configuration is not valid.*

ter:InvalidAction	
env:Sender ter:InvalidArgVal ter:UknownAction	*The requested action type is not known.*
enc:Receiver ter:Action ter:TooManyActions	*There is not enough space to add actions.*

5.3.4 ModifyActions

The modify action operation modifies action configurations. The modify action operation is atomic. If a service provider can not modify all of requested action configurations, then, the service provider responds with a fault message.

Table 6: ModifyActions command

ModifyActions	Access Class:WRITE_SYSTEM
Message name	**Description**
ModifyActionsRequest	*The request message specifies list of Actions to be modified.* tt:Action **Action**[1][unbounded]
ModifyActionsResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidAction	*The requested action configuration is not valid.*
env:Sender ter:InvalidArgs ter:ActionNotExist	*The actions do not exist.*

All action parameters, except the action type, can be modified. The service provider shall return InvalidAction error if the request attempts to change the action type with modify action request.

5.3.5 DeleteActions

The delete operation deletes actions. The delete action operation is atomic. If a service provider can not delete all of requested actions, then, the service provider responds with a fault message.

Table 7: DeleteActions command

DeleteActions	Access Class:WRITE_SYSTEM
Message name	**Description**
DeleteActionsRequest	*The request message specifies list of Actions to be removed.* xs:ReferenceToken **Token**[1][unbounded]
DeleteActionsResponse	*The response is an empty message.*
Fault codes	**Description**

env:Sender ter:InvalidArgs ter:ActionNotExist	*The actions do not exist.*
env:Receiver ter:Action ter:ConfigurationConflict	*The service cannot delete the actions without creating a conflicting configuration.*

5.3.6 GetServiceCapabilities

The get capabilities operation returns the Action Engine capabilities.

Table 8: Get Service Capabilities command

GetServiceCapabilities	Access Class:PRE_AUTH
Message name	**Description**
GetServiceCapabilitiesRequest	*The request is an empty message.*
GetServiceCapabilitiesResponse	*The capability response message contains Action Engine capabilities information.* tt:ActionEngineCapabilities **Capabilities** [1][1]
Fault codes	**Description**
	There are no operation specific faults.

The capabilities data structure indicates the maximum number of action and action trigger instances (See Section 5.7.5).

5.3.7 GetActionTriggers

Returns configured action triggers.

Table 9: GetActionTriggers command

GetActionTriggers	Access Class:READ_SYSTEM
Message name	**Description**
GetActionTriggersRequest	*This is an empty message.*
GetActionTriggersResponse	*The request message specifies list of action triggers.* tt:ActionTrigger **ActionTrigger**[0][unbounded]
Fault codes	**Description**
	There are no operation specific faults.

5.3.8 CreateActionTriggers

Creates action triggers. The create action triggers operation is atomic. If a service provider can not create all of requested action triggers, then, the service provider responds with a fault message.

Table 10: CreateActionTriggers command

CreateActionTriggers	Access Class:WRITE_SYSTEM
Message name	**Description**
CreateActionTriggersRequest	*The request message specifies list of action triggers to be created.* tt:ActionTriggerConfiguration **ActionTrigger** [1][unbounded]
CreateActionTriggersResponse	*The request message specifies list of action triggers.* tt:ActionTrigger **ActionTrigger**[0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidActionTrigger	*The requested action trigger configuration is not valid.*
env:Receiver ter:Action ter:TooManyActionTriggers	*There is not enough space to add action triggers.*

5.3.9 ModifyActionTriggers

Modifies existing action triggers. The modify action triggers operation is atomic. If a service provider can not modify all of requested action trigger configurations, then, the service provider responds with a fault message.

Table 11: ModifyActionTriggers command

ModifyActionTriggers	Access Class:WRITE_SYSTEM
Message name	**Description**
ModifyActionTriggersRequest	*The request message specifies list of action triggers to be modified.* tt:ActionTrigger **ActionTrigger**[1][unbounded]
ModifyActionTriggersResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidActionTrigger	*The requested action trigger configuration is not valid.*
env:Sender ter:InvalidArgs ter:ActionTriggerNotExist	*The action triggers do not exist.*

5.3.10 DeleteActionTriggers

Deletes action triggers. The delete action triggers operation is atomic. If a service provider can not delete all of requested action triggers, then, the service provider responds with a fault message.

Table 12: DeleteActionTriggers command

DeleteActionTriggers	Access Class:WRITE_SYSTEM
Message name	**Description**
DeleteActionTriggersRequest	*The request message specifies list of action triggers to be removed.* tt:ReferenceToken **Token**[1][unbounded]
DeleteActionTriggersResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgs ter:ActionTriggerNotExist	*The action triggers do not exist.*

5.4 Standard actions

5.4.1 CommandAction

The command action executes one or more ONVIF commands on the device itself or on a remote device. This mechanism is applicable to a wide range of ONVIF commands.

```
<tt:ActionDescription Name="tt:CommandAction">
  <tt:ParameterDescription>
    <tt:SimpleItemDescription Name="XAddr" Type="xs:anyURI"/>
    <tt:SimpleItemDescription Name="Operation" Type="xs:QName"/>
    <tt:ElementItemDescription Name="Parameters" Type="tt:AnyHolder"/>
  </tt:ParameterDescription>
</tt:ActionDescription>
```

The Action has the following configuration parameters;

Parameters	**Description**
Xaddr	Endpoint where the command shall be executed. Use localhost for local commands instead to avoid that device IP address changes break the commands.
Operation	Operation name including namespace
Parameters	Input parameter of the operation as defined in the corresponding ONVIF WSDL file.

5.4.2 E-Mail Action

E-mail action definition allows application to send an e-mail to the configured addresses. The e-mail content is provided during the execution.

```
<tt:ActionDescription Name="tt:EMailAction">
  <tt:ParameterDescription>
    <tt:ElementItemDescription Name="Destinations"
                               Type="tae:EMailServerConfiguration"/>
    <tt:ElementItemDescription Name="Receivers" Type="tae:EMailReceiverConfiguration"/>
    <tt:SimpleItemDescription  Name="Sender"    Type="xs:string"/>
    <tt:SimpleItemDescription  Name="Subject"   Type="xs:string"/>
    <tt:ElementItemDescription Name="Body"      Type="tae:EMailBodyTextConfiguration"/>
    <tt:ElementItemDescription Name="Attachment"
Type="tae:EMailAttachmentConfiguration" />
    <tt:ElementItemDescription Name="MediaReference" Type="tt:MediaSource"/>
  </tt:ParameterDescription>
</tt:ActionDescription>
```

The Action has the following configuration parameters;

Parameters		**Description**
EMailServerConfiguration		Contains Email Server configuration
	SMTP Configuration	SMTP Server IP address and port number
	POP Configuration	POP Server IP address and port number
	Authentication Configuration	Configure username and password
Receivers		Contains list of receivers (TO and CC fields)
Sender (From)		Sender information
Subject		E-Mail subject Line configuration
Body		E-Mail body text configuration
Attachment		E-Mail Attachment file name configuration
	File Name	Attachment file name
	File Name Suffix	Attachment file name suffix configuration
Media Reference		Media Profile Token

5.4.3 HTTP POST Action

HTTP POST action definition allows application to send data to the configured address via HTTP POST mechanism. The content is provided during the execution.

```
<tt:ActionDescription Name="tt:HttpPOSTAction">
  <tt:ParameterDescription>
    <tt:ElementItemDescription Name="Destinations" Type="tae:HttpHostConfigurations"/>
    <tt:ElementItemDescription Name="POSTContent" Type="tae:PostContentConfiguration"/>
  </tt:ParameterDescription>
</tt:ActionDescription>
```

The Action has the following configuration parameters;

Parameters		**Description**
Destinations		Contains HTTP server configurations
	Server Address	Server IP address and port number
	Authentication	Server Authentication configuration (username, password, authentication mechanism)
	URI	Request-Uri
	Protocol	Select HTTP or HTTPS
POST Content		POST content configuration for form data, triggering event data, and media
	Media Reference	Media Profile Token

5.4.4 FTP Action

FTP action definition allows application to send data to the configured address via FTP mechanism. The content is provided during the execution.

```
<tt:ActionDescription Name="tt:FtpAction">
  <tt:ParameterDescription>
    <tt:ElementItemDescription Name="Destinations" Type="tae:FtpHostConfigurations"/>
    <tt:ElementItemDescription Name="FtpContent" Type="tae:FtpContentConfiguration”/>
    <tt:ElementItemDescription Name="MediaReference" Type="tt:MediaSource”/>
  </tt:ParameterDescription>
</tt:ActionDescription>
```

The Action has the following configuration parameters;

Parameters		**Description**
Destinations		
	Server Address	Server IP and Port number configuration
	Upload Path	Upload path on FTP server
	Authentication	Authentication configuration
FTP Content		FTP Content Configuration Includes configurations for the upload of sequence of images and upload of a file from local storage
Media Reference		Media Profile Token

5.4.5 SMS Action

SMS action definition allows application to send data to the configured address via SMS Text Messaging mechanism. The content is provided during the execution.

```
<tt:ActionDescription Name="tt:SMSNotificationAction">
  <tt:ParameterDescription>
    <tt:ElementItemDescription Name="SMSProvider"
                               Type="tae:SMSProviderConfiguration"/>
    <tt:ElementItemDescription Name="SMSSenderInfo"
                               Type="tae:SMSSenderConfiguration"/>
    <tt:SimpleItemDescription  Name="Destination" Type="xs:string"/>
    <tt:SimpleItemDescription  Name="Message"  Type="tae:SMSMessage"/>
  </tt:ParameterDescription>
</tt:ActionDescription>
```

The Action configuration contains the following configuration parameters;

Parameters	**Description**
SMSProvider	SMS Provider Information
SMSSenderInfo	SMS Sender Information
Destination	Destinations that will receive the message
Message	Text Message

5.4.6 Camera Local Recording Action

Camera Local Recording Action definition allows application to initiate recording of data to the camera local storage. The content is provided during the execution..

```
<tt:ActionDescription Name="tt:RecordingAction">
  <tt:ParameterDescription>
    <tt:ElementItemDescription Name="RecordingConfiguration"
                               Type="tae:RecordingActionConfiguration"/>
  </tt:ParameterDescription>
</tt:ActionDescription>
```

The Action configuration contains the following configuration parameters;

Parameters		**Description**
Recording configuration		Local recording configuration
	Pre recording duration	Recording before the triggering event
	Post recording duration	Recording after alarm recording duration
	Record duration	Record duration
	Recording frame rate	Recording frame rate
	Audio Recording on/off	Whether Audio recording on/off

5.5 Capabilities

The capabilities reflect optional functions and functionality of a service. The following capabilities are available:

MaximumActions: The maximum number of actions that the service provider can concurrently support.

MaximumTriggers: The maximum number of trigger configurations that the service provider can concurrently support.

5.6 Events

The action engine events allow controlling of action execution as well as building of action chains.

The action service shall dispatch events through the event service.

5.6.1 Action completed

The device shall be capable of generating the following event whenever an action has been completed. This event is triggered whenever an action completes without a fault message. When an event is generated due to execution of CommandAction, the Data element of event message can contain the typed information of the request and response message payloads of corresponding ONVIF message exchange.

```
Topic: tns1:ActionEngine/Completed

<tt:MessageDescription IsProperty="false">
   <tt:Source>
    <tt:SimpleItemDescription Name="Token" Type="tt:ReferenceToken"/>
    <tt:SimpleItemDescription Name="ActionName" Type="xs:string"/>
   </tt:Source>
   <tt:Data>
    <tt:ElementItemDescription Name="RequestResponse"
            Type="tt:ActionEngineEventPayload"/>
   </tt:Data>
</tt:MessageDescription>
```

5.6.2 Action failed

The device shall be capable of generating the following event whenever an action has been completed with an error.

This event is triggered whenever an action completes with a fault message or the action could not be executed because of the other error. The response contains the fault codes including any sub codes.

```
Topic: tns1:ActionEngine/Failed

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

```
    <tt:SimpleItemDescription Name="ActionName" Type="xs:string"/>
   </tt:Source>
   <tt:Data>
    <tt:ElementItemDescription Name="RequestResponse"
            Type="tt:ActionEngineEventPayload"/>
   </tt:Data>
</tt:MessageDescription>
```

5.7 Service-specific data types

5.7.1 ActionConfiguration

Action Configuration data type contains the configuration settings of action configuration parameters, service requester given action Name, and service provider supported action type value.

```
<xs:complexType name="ActionConfiguration"/>
     <xs:attribute name="Name" type="xs:string" use="required"/>
     <xs:attribute name="Type" type="xs:QName" use="required"/>
     <xs:element name="Parameters" type="tt:ItemList"/>
</xs:complexType>
```

- **Name**
  User given name.

- **Type**
  Denotes the action type.

- **Parameters**
  Action configuration parameter settings.

5.7.2 Action

Action data type contains the configuration settings of one action instance and service provider assigned unique identifier for this action configuration.

```
<xs:complexType name="Action"/>
 <xs:attribute name="Token" type="tt:ReferenceToken" use="required"/>
 <xs:element name="Configuration" type="tae:ActionConfiguration"/>
</xs:complexType>
```

- **Token**

  Unique Action identifier that service implementation assigned to the action configuration.

- **Configuration**

  Action configuration contains action type, user given action name, and configuration parameter settings.

5.7.3 ActionTriggerConfiguration

Action Trigger configuration data type contains mandatory Topic Expression (Section Topic Filter in [Core Specification]), optional Message content expression (Section Message Content Filter in [Core Specification]), and set of actions to be triggered.

```
<xs:complexType name="ActionTriggerConfiguration"/>
 <xs:element name="TopicExpression" type="wsnt:TopicExpressionType"/>
 <xs:element   name="ContentExpression"   type="wsnt:QueryExpressionType"
minOccurs="0"/>
 <xs:element  name="ActionToken"  type="tt:ReferenceToken"  minOccurs="0"
maxOccurs="unbounded"/>
</xs:complexType>
```

- **TopicExpresion**
  Topic expression, for example, to trigger only for relays. Trigger based on event topic.

- **ContentExpression**

  Content expression, for example, to trigger only when the relay value is “On”. Trigger based on content data in event.

- **ActionToken**

  Reference to actions to be triggered when the conditions are satisfied.

5.7.4 ActionTrigger

Action Trigger data type contains the service provider assigned unique identifier for the configuration and action trigger configuration data.

```
<xs:complexType name="ActionTrigger"/>
 <xs:attribute name="Token" type="tt:ReferenceToken" use="required"/>
 <xs:element name="Configuration" type="tae:ActionTriggerConfiguration"/>
</xs:complexType>
```

- **Token**

  Unique Action Trigger identifier that service provider assigned to the action trigger configuration.

- **Configuration**

  Action Trigger Configuration

5.7.5 ActionEngineCapabilities

Action Engine Capabilities data structure contains the maximum number of supported actions and number of actions in use for generic as well as specific action types.

```
<xs:complexType name="ActionEngineCapabilities"/>
 <xs:attribute name="MaximumTriggers" type="xs:positiveInteger"/>
 <xs:attribute name="MaximumActions" type="xs:positiveInteger"/>
 <xs:element name="ActionCapabilities" type="tae:ActionTypeLimits" minOccurs="0"
maxOccurs="unbounded"/>
</xs:complexType>
```

- **MaximumActions**
  The maximum number of actions that the service provider can concurrently support.

- **MaximumTriggers**

  The maximum number of trigger configurations that the service provider can concurrently support.

- **ActionCapabilities**

  Limits for each action type.

5.7.5.1 ActionTypeLimits

ActionTypeLimits data structure contains maximum and current usage information for a specific action type in the service provider.

```
<xs:complexType name="ActionTypeLimits"/>
 <xs:attribute name="Type" type="xs:QName" use="required"/>
 <xs:attribute name="Maximum" type="xs:positiveInteger" use="required"/>
 <xs:attribute name="InUse" type="xs:nonNegativeInteger" use="optional" default="0"/>
</xs:complexType>
```

- **Type**
  Action Type

- **Maximum**
  For the specific action type, the maximum number of actions that could be concurrently supported by the service provider

- **InUse**

  For the specific action type, the number of actions in use by the service provider

5.7.6 SupportedActions

SupportedActions data structure lists the available action types that service provider supports. For each action type, data structure contains the action configuration parameters.

```
<xs:complexType name="SupportedActions"/>
 <xs:element name="ActionContentSchemaLocation" type="xs:anyURI" minOccurs="0"
maxOccurs="unbounded"/>
 <xs:element name="ActionDescription" type="tae:ActionConfigDescription" minOccurs="0"
maxOccurs="unbounded"/>
</xs:complexType>
```

- **ActionContentSchemaLocation**

List of location of schemas, that are referenced in the supported action descriptions. If the action descriptions reference data types in the ONVIF schema file, then, the ONVIF schema file MUST be explicitly listed.

- **ActionDescription**
  List of actions supported by Action Engine Service provider

5.7.7 ActionEngineEventPayload

Action Engine Event Payload data structure contains the information about the ONVIF command invocations. Since this event could be generated by other or proprietary actions, the command invocation specific fields are defined as optional and additional extension mechanism is provided for future or additional action definitions.

```
<xs:complexType name="ActionEngineEventPayload">
 <xs:sequence>
  <xs:element name="RequestInfo" type="soapenv:Envelope"
              minOccurs="0" maxOccurs="1"/>
  <xs:element name="ResponseInfo" type="soapenv:Envelope"
              minOccurs="0" maxOccurs="1"/>
  <xs:element name="Fault" type="soapenv:Fault"
              minOccurs="0" maxOccurs="1"/>
  <xs:element name="Extension" type="tt:ActionEngineEventPayloadExtension"
              minOccurs="0"/>
 </xs:sequence>
 <xs:anyAttribute processContents="lax"/>
</xs:complexType>
```

5.8 Service-specific fault codes

Table 13 below lists the analytics service-specific fault codes. Each command can also generate a generic fault. Refer to 5.11.2.1 in [Core Specification].

The specific faults are defined as subcode of a generic fault, see Section 5.11.2.1 in [Core Specification]. The parent generic subcode is the *subcode* at the top of each row below and the specific fault *subcode* is at the bottom of the cell.

Table 13: The Action Engine specific fault codes

Fault Code	Parent Subcode Subcode	Fault Reason	Description
env:Sender	ter:InvalidArgVal ter:ActionAlreadyExist	The action exists	The action identifier exists already in the configuration.
env:Sender	ter:InvalidArgVal ter:UnknownAction	Unknown action type	The provider does not have the requested action type.
env:Sender	ter:InvalidArgVal ter:InvalidAction	Action configuration problem	The suggested action configuration is invalid.
env:Sender	ter:InvalidArgs ter:ActionNotExist	The action does not exist	The suggested configurations do not exist.
env:Sender	ter:InvalidArgVal ter:InvalidActionTrigger	The action trigger is invalid	The suggested configuration is invalid.
env:Sender	ter:InvalidArgs ter:ActionTriggerNotExist	The action trigger does not exist	The action triggers do not exist
env:Receiver	ter:Action ter:TooManyActions	Too many action configuration	There is not sufficient resource to support suggested action configurations
env:Receiver	ter:Action ter:TooManyActionTriggers	Too many action trigger configurations	There is not sufficient resource to support suggested action trigger configurations
env:Receiver	ter:Action ter:ConfigurationConflict	Conflict when using new settings	The new settings result in an inconsistent configuration

Annex A

(informative)

A.1 Example: Command Action configuration

Starting and stopping of recording job on Recording Control Service by using CommandAction definition is illustrated in the following.

Start a recording on the same device when a relay is switched on.

```
<tae:Action Token="453638">
 <tae:Configuration Name="StartRecording" Type="tae:CommandAction">
   <tt:Parameters>
     <tt:SimpleItem Name="XAddr"
            Value="http://127.0.0.1/Onvif/device_service"/>
     <tt:SimpleItem Name="Operation" Value="tt:SetRecordingJobMode"/>
     <tt:ElementItem Name="Parameters">
       <tt:JobToken>123</tt:JobToken>
       <tt:Mode>Active</tt:Mode>
     </tt:ElementItem>
   </tt:Parameters>
 </tae:Configuration>
</tae:Action>
```

Stop the recording one minute after the relay is switched off.

```
<tae:Action Token="5674749">
 <tae:Configuration Name="StopRecording" Type="tae:CommandAction">
   <tt:Parameters>
     <tt:SimpleItem Name="XAddr"
            Value="http://127.0.0.1/Onvif/device_service"/>
     <tt:SimpleItem Name="Operation" Value="tt:SetRecordingJobMode"/>
     <tt:ElementItem Name="Parameters">
       <tt:JobToken>123</tt:JobToken>
       <tt:Mode>Idle</tt:Mode>
     </tt:ElementItem>
   </tt:Parameters>
 </tae:Configuration>
</tae:Action>
```

A.2 Example: E-Mail Action configuration

The following demonstrates the E-Mail action configuration information.

```
<tt:Action Token="576858">

 <tt:Configuration Name="Notify_SGrp_1" Type="tt:EMailAction">

  <tt:Parameters>

   <!-- Server configurations -->
   <tt:ElementItem Name="Destinations">
    <tt:SMTPConfig>
     <tt:HostAddress formatType="ipv4">172.34.123.65</tt:HostAddress>
    </tt:SMTPConfig>
    <tt:POPConfig>
     <tt:HostAddress formatType="ipv4">162.34.123.56</tt:HostAddress>
    </tt:POPConfig>
    <tt:AuthenticationConfig mode="none"/>
   </tt:ElementItem>

   <!-- Email receiver configurations -->
   <tt:ElementItem Name="Receivers">
    <tt:TO>sg1@hq.co</tt:TO>
    <tt:TO>sg2@hq.co</tt:TO>
    <tt:CC>sgm@hq.co</tt:CC>
   </tt:ElementItem>

   <tt:ElementItem Name="Sender">NVT@123.23.23.22</tt:ElementItem>

   <!-- Attached File Name configurations -->
   <tt:ElementItem Name="Attachment">
    <tt:FileName>EVT_</tt:FileName>
    <tt:doSuffix>dateTime</tt:doSuffix>
   </tt:ElementItem>

   <!-- Media Profile configuration -->
   <tt:ElementItem Name="MediaReference">
    <tt:ProfileToken>4757585</tt:ProfileToken>
   </tt:ElementItem>

  </tt:Parameters>

 </tt:Configuration>

</tt:Action>
```

A.3 Example: HTTP POST Action configuration

The following demonstrates the configuration for HTTP POST action. The configuration utilizes the Media Profile Reference to indicate the media source.

```
<tt:Action Token="343234">

 <tt:Configuration Name="POST2_VMS1" Type="tt:HttpPOSTAction">

  <tt:Parameters>

   <!-- HTTP Server Address configuration -->
   <tt:ElementItem Name="Destinations">
    <tt:HttpDestination uri="/post_event">
     <tt:HostAddress formatType="ipv4">192.134.123.214</tt:HostAddress>
    </tt:HttpDestination>
   </tt:ElementItem>

   <!-- HTTP POST Request Body configuration and Media Profile Token -->
   <tt:ElementItem Name="POSTContent">
     <tt:MediaReference>
        <tt:ProfileToken>6565746</tt:ProfileToken>
     </tt:MediaReference>
     <tt:PostBody includeMedia="true" includeEvent="true"/>
   </tt:ElementItem>

  </tt:Parameters>

 </tt:Configuration>

</tt:Action>
```

A.4 Example: FTP Action configuration

An example configuration for sending images (from Media Profile) into a directory (Destinations/UploadPath) in FTP server (Destinations) for 15 minutes (FtpContent/HowLong) with 10 seconds snapshots (FtpContent/SampleInterval) with the configured file names (FtpContent/FileName).

```
<tt:Action Token="45345" >

 <tt:Configuration Name="FTP_STORE_BRNCH1" Type="tt:FtpAction">

  <tt:Parameters>

   <tt:ElementItem Name="Destinations">
     <tt:FtpDestination>
      <tt:HostAddress formatType="ipv4">132.34.13.64</tt:HostAddress>
      <tt:UploadPath>/home/cam2</tt:UploadPath>
      <tt:FtpAuthentication>
       <tt:username> </tt:username>
       <tt:password> </tt:password>
      </tt:FtpAuthentication>
     </tt:FtpDestination>
   </tt:ElementItem>

   <tt:ElementItem Name="FtpContent">
     <tt:FtpContentConfig Type="image_upload">
      <tt:UploadImages>
       <tt:HowLong>P0Y0M0DT0H15M</tt:HowLong>
       <tt:SampleInterval>P0Y0M0DT0H0M10S</tt:SampleInterval>
       <tt:FileName file_name="ALM_" suffix="sequence"/>
      </tt:UploadImages>
     </tt:FtpContentConfig>
   </tt:ElementItem>

   <tt:ElementItem Name="MediaReference">
     <tt:ProfileToken>5657959</tt:ProfileToken>
   </tt:ElementItem>

  </tt:Parameters>

 </tt:Configuration>

</tt:Action>
```

A.5 Example: GetActions Response

For example, GetActionsResponse message contains installed Action information as;

```
..

<tt:Action Token=”191918”>
 <tt:Configuration Name=”HQMail” Type=”tt:EMailAction”>
  <tt:Parameters>
   <tt:ElementItem Name=”Destinations”>
     ….
   </tt:ElementItem>
  </tt:Parameters>
 </tt:Configuration>
</tt:Action>

<tt:Action Token=”1913338”>
 <tt:Configuration Name=”HQMail_2” Type=”tt:EMailAction”>
  <tt:Parameters>
   <tt:ElementItem Name=”Destinations”>
     ….
   </tt:ElementItem>
  </tt:Parameters>
 </tt:Configuration>
</tt:Action>

..
```

Annex B. Revision History

Rev.	Date	Editor	Changes
1.0	Feb-2012	Hasan T. Ozdemir	First release.