VAPIX® WEB SERVICES

Event and Action Services

COPYRIGHT NOTICE

This document is copyright protected and is the property of Axis Communications AB and may not be copied, reproduced or distributed in any way without the prior written consent of Axis Communications AB.

VAPIX® LICENSE AGREEMENT

This VAPIX® License Agreement ("License") is a legal agreement between you (either individual or an entity) and Axis Communications AB ("Axis"). By using the INTERFACE and INTERFACE DESCRIPTION (each defined below), whether in whole or in part, you agree to be bound by the terms of this License.

1. GRANT OF LICENSE

Axis hereby grants to you the right to use the AXIS VAPIX application programming interface ("INTERFACE") and the written specification of the INTERFACE (the "INTERFACE DESCRIPTION") for the sole and limited purpose of creating, manufacturing and developing a solution that integrates any unit or portion included in the product range of Axis network products, as defined by Axis at its discretion (an "Axis Product") and to market, sell and distribute any such solution.

2. COPYRIGHT

The INTERFACE and the INTERFACE DESCRIPTION are owned by Axis and are protected by copyright laws and international treaty provisions. Any use of the INTERFACE and/or the INTERFACE DESCRIPTION outside the limited purpose set forth in Section 1 above is strictly prohibited.

3. RESTRICTIONS ON USE

You have no rights with respect to the INTERFACE, INTERFACE DESCRIPTION or any portions thereof and shall not use the INTERFACE, INTERFACE DESCRIPTION or any portion thereof except as expressly set forth herein. You may not reverse engineer, decompile, or disassemble the INTERFACE except to the extent required to obtain interoperability with other independently created computer programs as permitted by mandatory law.

4. THIRD PARTY RIGHTS

You agree that you are fully responsible for your own conduct while using the INTERFACE and integrating any Axis Products into your solution and the consequences thereof. Axis Products may be combined with a virtually infinite number of potential solutions. Consequently, you recognize that (i) other third parties may claim to own patents or copyrights that could cover certain solutions which integrate Axis products, or which result from the combination of Axis products and additional technology or solutions and (ii) you are responsible for ensuring that any solution which integrates with an Axis Product, or a combination of a solution and an Axis product, does not infringe upon or misappropriate any intellectual property or personal right of any third party.

5. TERMINATION

This License is effective until terminated. Your rights under this License will terminate automatically without notice from Axis if you fail to comply with any term(s) of this License. Upon the termination of this License, you shall cease all use and disposition of the INTERFACE and/or THE INTERFACE DESCRIPTION whether for the purpose set forth in Section 1 above or not.

6. REPRESENTATIONS AND WARRANTIES; DISCLAIMER

6.1. You represent and warrant that (i) any solution created, manufactured and/or developed by you which integrates an Axis Product shall not infringe or otherwise violate any third party rights, including but not limited to third party intellectual property rights; and (ii) your use of the INTERFACE and INTERFACE DESCRIPTION will comply with all applicable foreign and domestic laws, rules and regulations.

6.2. YOUR USE OF THE INTERFACE IS AT YOUR SOLE RISK. THE INTERFACE AND THE INTERFACE DESCRIPTION ARE DELIVERED FREE OF CHARGE AND "AS IS" WITHOUT WARRANTY OF ANY KIND. THE ENTIRE RISK AS TO THE USE, RESULTS AND PERFORMANCE OF THE INTERFACE AND THE INTERFACE DESCRIPTION IS ASSUMED BY THE USER/YOU. AXIS DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, NON-INFRINGEMENT AND PRODUCT LIABILITY, OR ANY WARRANTY ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE WITH RESPECT TO THE INTERFACE AND THE INTERFACE DESCRIPTION. Without limiting the generality of the foregoing, you acknowledge and agree that Axis does not make any representation or warranty that the integration of Axis Products into your solution does not infringe any third party rights. You are solely responsible for any intellectual property infringement claims that are based on or relate to solutions created, manufactured and distributed by you which integrate Axis Products. Axis is unaware of the details regarding your particular solution, has not conducted any investigation relating to potential third party rights issues relating to your solution and does not accept any responsibility or liability with respect thereto.

6.3. THIS LICENSE DOES NOT CONVEY ANY LICENSE TO THIRD PARTY INTELLECTUAL PROPERTY. YOU ARE SOLELY RESPONSIBLE FOR (I) EXAMINING WHETHER THE INTERFACE AND THE INTERFACE DESCRIPTION ARE ENCUMBERED BY OR INFRINGES UPON A RIGHT HELD BY A THIRD PARTY AND (II) ANY INTELLECTUAL PROPERTY INFRINGEMENT CLAIMS THAT ARISE OUT OF OR RELATE TO SOLUTIONS CREATED, MANUFACTURED AND DISTRIBUTED BY YOU WHICH INTEGRATE AXIS PRODUCTS.

7. LIMITATION OF LIABILITY

7.1. AXIS SHALL NOT BE LIABLE FOR LOSS OF DATA, LOSS OF PRODUCTION, LOSS OF PROFIT, LOSS OF USE, LOSS OF CONTRACTS OR FOR ANY OTHER CONSEQUENTIAL, ECONOMIC OR INDIRECT LOSS WHATSOEVER IN RESPECT OF USE OR DISPOSITION OF THE INTERFACE AND THE INTERFACE DESCRIPTION.

7.2. AXIS TOTAL LIABILITY FOR ALL CLAIMS IN ACCORDANCE WITH THE USE OF THE INTERFACE AND THE INTERFACE DESCRIPTION SHALL NOT EXCEED THE PRICE PAID FOR THE INTERFACE AND THE INTERFACE DESCRIPTION.

7.3. YOU UNDERTAKE NOT TO PURSUE ANY CLAIMS WHATSOEVER AGAINST AXIS OR ITS AFFILIATES RELATING TO OR EMANATING FROM THE INTERFACE AND THE INTERFACE DESCRIPTION OR YOUR INTEGRATION OF AN AXIS PRODUCT INTO YOUR SOLUTION.

8. INDEMNIFICATION

You will indemnify and hold Axis, its subsidiaries, affiliates, officers, employees, and agents harmless from any and all claims, damages, losses, liabilities, actions, judgments, costs, and expenses brought by a third party, including claims for infringement of intellectual property rights, arising out of or in connection with (i) your use of the INTERFACE or INTERFACE DESCRIPTION other than in accordance with the terms of this agreement, and/or (ii) any solution created, manufactured and/or developed by you which integrates an Axis Product.

9. GOVERNING LAW

This agreement shall be deemed performed in and shall be construed by the laws of Sweden. All disputes in connection with this agreement shall be finally settled by arbitration in accordance with the Rules of the Arbitration Institute of the Stockholm Chamber of Commerce. The place of arbitration shall be Malmö, Sweden. The language of the proceedings, documentation and the award shall be English.

Event and Action Services

Table of Contents

Table of Contents

Event and Action Services

1 Introduction

This document describes how to use the VAPIX® Event Service and Action Service APIs available in Axis network video products with firmware 5.50 and later. The APIs are based on web services.

The Event and Action Services are used to configure the Axis product to perform actions when the product detects an occurrence of some kind, for example to start a recording if motion is detected outside office hours or to run a guard tour once every hour.

The **Event Service** is used to:

- List events
- Configure recurring events (schedules and recurrences)
- Emit virtual input events

Events can be used in actions rules and in notification subscriptions.

For information about available events, see *Event Declarations, on page 18*.

The **Action Service** is used to:

- List action and recipient templates
- Create action configurations
- Create action rules

For information about available action templates, see *Action Templates, on page 36*.

For information about available recipient templates, see *Recipient Templates, on page 45*.

See also the Event and Action Services sample code and the VAPIX® Entry Service sample code.

1.1 Prerequisites

1.1.1 Identification

Use VAPIX® Entry Service API to check if the API is supported.

Namespace: `http://www.axis.com/vapix/ws/event1`
Namespace: `http://www.axis.com/vapix/ws/action1`
Firmware: 5.50
Supported events, action templates and recipient templates are product dependent.

1.1.2 API Specification

The API specifications are available as WSDL files at:

VAPIX® Event Service API	`http://www.axis.com/vapix/ws/event1/EventService.wsdl`
VAPIX® Action Service API	`http://www.axis.com/vapix/ws/action1/ActionService.wsdl`

1.1.3 Obsoletes

The parameter-based event handling system in products with firmware prior to 5.40 is deprecated but supported for backward compatibility.

Event and action configurations created using the parameter-based API and the web services-based API are stored in separate configuration domains.

1.2 Relationship to Parameter-Based API

VAPIX® Event and Action Services APIs replace the parameter-based event handling system in products with firmware prior to 5.40. Some changes include:

- Event Servers are replaced by Recipients.
- Event Types (parameter group `Event.E#`) are replaced by Action Rules. In contrast to Event Types, an Action Rule can have multiple conditions (triggers).
- Triggers are replaced by Start Event and Conditions.

See also *Terminology, on page 7*.

Existing Event Types can be converted to Action Rules from the product's webpages.

1.3 Terminology

Action – An action is a task that can be performed by the Axis product. The action is initiated by an action rule. Examples: record video, send e-mail, activate output port.

Action rule – An action rule specifies how and when the Axis product should perform an action. Example: record video when motion is detected outside office hours.

Condition – An additional condition that must be fulfilled to trigger the action rule.

Event – Event is used as a collective name for stateful and stateless events. Events are emitted when the Axis product detects an occurrence of some kind, for example motion in the camera's field of view or a signal from an I/O port. Events can be used in action rules (as start event or condition) and in notification subscriptions.

Fixed action – A fixed action runs during a fixed, predefined time. Depending on the action type, the length of time is defined by one or more action parameters, by the length of the audio clip to be played or similar.

Recipient – A recipient is a network resource that can receive data, for example video clips or notification messages. Examples: network share, FTP server, email address.

Stateful event – A stateful event is a property (a state variable) with a number of states. The event is always in one of its states. Example: The Motion Detection event is in state `true` when motion is detected and in state `false` when motion is not detected.

Stateless event – A stateless event is a momentary occurrence (a pulse). Example: Storage device removed.

Unlimited action – An unlimited action runs as long as all conditions are fulfilled.

Start event – A condition that must be fulfilled to trigger the action rule. Called **Trigger** in the product's webpages.

1.4 References

All VAPIX® references are available at:

http://www.axis.com/vapix

2 Events, Actions and Action Rules

This section contains examples showing how to use the Event and Action services. All examples are written in C#; modify as required for applications in other languages.

2.1 Identification Using Entry Service

To check if the Axis product supports the Event and Action Services APIs, use `GetServices` from VAPIX® Entry Service API as outlined below. The Entry Service API is available for products with firmware 5.60 and later. For products with firmware 5.50, use the helper functions defined in appendix *Helper Functions*.

We start by defining the IP address, user name and password for the Axis product and the namespaces of the event and action services. Then we use `CreateEntryServiceClient` to create an entry service client.

Note

The function `CreateEntryServiceClient` is defined in the VAPIX® Entry Service API sample code. For more information about the entry service, see the VAPIX® Entry Service API documentation.

```
// Define the address, user name and password for the Axis product.  <ip-address>
is an IP address or host name.
string address="<ip-address>";
string username="<user name>";
string password="<password>";

// Define the namespaces of the event and action services.
string eventTargetNamespace = "http://www.axis.com/vapix/ws/event1";
string actionTargetNamespace = "http://www.axis.com/vapix/ws/action1";

// Create an Entry Service client.
EntryClient myEntryService = CreateEntryServiceClient(address, username, password);
```

Next, we use the entry service client and `GetServices` to get a list of all services in the Axis product. We then search the list to check if the event and action services are included.

If a service is found, use `CreateEventServiceClient` and `CreateActionServiceClient` to create service clients. See *Helper Functions*. These functions are not part of the API and are created in the same way as `CreateEntryServiceClient`.

```
// Get a list of all services.
Service[] serviceList = myEntryService.GetServices(false);

// Check if event service is supported.  If supported, create a service client.
for (i = 0; i < serviceList.count; i++)
{
 if (serviceList[i].Namespace == eventTargetNamespace)
 {
  // Get the service address.
  string eventXaddr = serviceList[i].Xaddr;

  // Create an event client.
   EventClient myEventService = CreateEventServiceClient(eventXaddr, username,
password);
 }

// Check if action service is supported.  If supported, create a service client.
 if (serviceList[i].Namespace == actionTargetNamespace)
 {
  // Get the service address.
  string actionXaddr = serviceList[i].Xaddr;
```

```
   // Create an action client.
    ActionClient myActionService = CreateActionServiceClient(actionXaddr, username,
password);
  }
}
```

2.2 Events

The term **event** is used as a collective name for stateful and stateless events.

Stateful event – A stateful event is a property (a state variable) with a number of states. The event is always in one of its states. Example: The Motion Detection event is in state `true` when motion is detected and in state `false` when motion is not detected.

Stateless event – A stateless event is a momentary occurrence (a pulse). Example: Storage device removed

Events are emitted when the Axis product detects an occurrence of some kind, for example motion in the camera's field of view or a signal from an I/O port. Events can be used in action rules (as start event or as a condition) and in notification subscriptions.

Events are added and removed dynamically depending on device configuration. For example, when a motion detection window is configured, a motion detection event is added.

To list the currently available events, use `GetEventInstances` as described in section *List Events*. The response is the event declaration topic tree described in section *Event Topic Tree Syntax*. The list of events can be used to construct event filter expressions for notification subscriptions and for action rule configuration.

It is also possible to list events using HTTP as described in section *List Events Using HTTP, on page 11*.

2.2.1 List Events

To list events, use `GetEventInstances` as in the example outlined below. The example uses `myEventService` created in section *Identification Using Entry Service, on page 8* .

```
try
{
// List event instances.
  TopicSetType response = myEventService.GetEventInstances();
  XmlElement[] eventInstances = response.Any;
}
catch (Exception e)
{
  HandleException(e);
}
```

2.2.2 Event Topic Tree Syntax

The events are listed as a `wstop:TopicSet` tree containing `aev:MessageInstance` elements in each leaf topic. The `aev:MessageInstance` element describes the content of the event.

The topic tree has the following syntax:

```
<aev:GetEventInstancesResponse>
<wstop:TopicSet>
<TOPIC1 aev:NiceName="topic1_nicename" wstop:topic="true">
 <TOPIC2 aev:NiceName="topic2_nicename" wstop:topic="true">
  <aev:MessageInstance aev:isProperty="true">
   <aev:SourceInstance>
    <aev:SimpleItemInstance aev:NiceName="key_nicename" Type="VALUETYPE"
     Name="KEYNAME">
     <aev:Value aev:NiceName="value1_nicename">value1</aev:Value>
     <aev:Value aev:NiceName="value2_nicename">value2</aev:Value>
     ...
    </aev:SimpleItemInstance>
    ...
   </aev:SourceInstance>
   <aev:DataInstance>
    <aev:SimpleItemInstance aev:NiceName="NICENAME" Type="VALUETYPE" Name="KEYNAME"
    isPropertyState="true">
     <aev:Value>VALUE1</aev:Value>
     <aev:Value>VALUE2</aev:Value>
     ...
    </aev:SimpleItemInstance>
   </aev:DataInstance>
  </aev:MessageInstance>
 </TOPIC2>
</TOPIC1>
</wstop:TopicSet>
</aev:GetEventInstancesResponse>
```

A real case would look like this:

```
<aev:GetEventInstancesResponse>
<wstop:TopicSet>
<tns1:AudioSource">
 <tnsaxis:TriggerLevel aev:NiceName="Audio detection" wstop:topic="true">
  <aev:MessageInstance aev:isProperty="true">
   <aev:SourceInstance>
    <aev:SimpleItemInstance aev:NiceName="Channel" Type="xsd:int"
     Name="channel">
     <aev:Value>1</aev:Value>
    </aev:SimpleItemInstance>
   </aev:SourceInstance>
   <aev:DataInstance>
    <aev:SimpleItemInstance aev:NiceName="Above alarm level" Type="xsd:boolean"
Name="triggered"
    isPropertyState="true">
    </aev:SimpleItemInstance>
   </aev:DataInstance>
  </aev:MessageInstance>
 </TriggerLevel>
</AudioSource>
</wstop:TopicSet>
</aev:GetEventInstancesResponse>
```

The prefix `aev` is a placeholder for the namespace `http://www.axis.com/vapix/ws/event1`.

Each `MessageInstance` element describes one type of event. If the event is stateful, the `isProperty` attribute is `true`. If the event is stateless, the `isProperty` attribute is omitted.

The `MessageInstance` element lists `SourceInstance` and `DataInstance` elements. Depending on the type of event, `SourceInstance` or `DataInstance` can be omitted. An event may have several `DataInstance` elements.

`SourceInstance` contains a `SimpleItem` describing the source of the event, for example an I/O port. `DataInstance` contains a `SimpleItem` describing the data that generates the event.

SimpleItems are described by `SimpleItemInstance` elements. Each `SimpleItemInstance` contains attributes and a list of `Value` child elements. The `Name` attribute is the name of the `SimpleItem`. The `Type` attribute describes the datatype for the `SimpleItem` values. The `NiceName` attribute is optional and contains the Key's nice name.

If the event is stateful, one of the `DataInstance` elements contains a `SimpleItem` of type boolean with the `isPropertyState` attribute set to `true`. This `SimpleItem` describes the current state of the event (for example the state of the I/O port).

For stateless event, the `isPropertyState` attribute is omitted.

The `Value` elements contains the possible values for the `SimpleItem`. The `NiceName` attribute is optional.

2.2.3 List Events Using HTTP

Listing events using HTTP is an alternative to using `GetEventInstances`. The HTTP method gives the same topic tree as `GetEventInstances`.

To request a list of events (the topic tree) using HTTP, send the following SOAP message to the Axis product using the URL `http://<ip>/vapix/services`. The HTTP header must contain the `Content-Type` below.

Example:

Content-Type: `application/soap+xml; action=http://www.axis.com/vapix/ws/event1/GetEventInstances; Charset=UTF-8`

Body:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:SOAP-ENC="http://www.w3.org/2003/05/soap-encoding"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <SOAP-ENV:Body>
  <m:GetEventInstances xmlns:m="http://www.axis.com/vapix/ws/event1"/>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

2.3 Actions

An **action** is a task that can be performed by the Axis product. The action is initiated by an action rule (see *page 14*). Examples: record video, send e-mail, activate output port.

A **fixed action** runs during a fixed, predefined time. Depending on the action type, the length of time is defined by one or more action parameters, by the length of the audio clip to be played or similar. An **unlimited action** runs as long as all conditions are fulfilled.

2.3.1 Action Templates

An **action template** defines an action type. The template lists the parameters needed to configure the action. If the action uses a recipient, the supported recipient type is also listed.

Supported operations:

- `GetActionTemplates` – list the action templates supported by the Axis product

An example of how an action implementation could look like:

```
NewActionConfiguration newAction = new NewActionConfiguration
{
  TemplateToken = "com.axis.action.fixed.play.audioclip",
  Parameters = new ActionParameters
  {
    Parameter = new[]
    {
      new ActionParameter { Name = "location", Value = "etc/audioclips/alarm.au" }
    }
  }
};
```

For available action templates, see *Action Templates, on page 36*.

2.3.2 Action Configurations

An **action configuration** specifies an action to be performed. The action configuration is created by specifying an action template and a list of parameters. If the action template refers to a recipient template, the parameters in the recipient template must also be added to the action configuration.

The Action Configuration ID identifies the action configuration and is used as input when creating an action rule.

Supported operations:

- `AddActionConfiguration` — create a new action configuration
- `GetActionConfigurations` — list all stored action configurations
- `RemoveActionConfiguration` — remove an action configuration

2.3.3 Create an Action Configuration

The example in this section shows how to create an action configuration that sets output port 1 to state `high`.

The example uses `myActionService` created in section *Identification Using Entry Service, on page 8* .

The action template `com.axis.action.unlimited.io.toggle` defines the unlimited output port action which has two parameters: `port` is I/O port number and `state` is the state to set the port to. For more information about the output port action template, see *page 38*.

Note

All parameters must be specified; if a parameter should not be used, its value should be set to an empty string.

```
try
{
  // Verify that the camera supports the action.
  ActionTemplate[] actiontemplates = myActionService.GetActionTemplates();
  if (actiontemplates.Any(
    template => template.TemplateToken == "com.axis.action.unlimited.io.toggle") ==
false)
  {
    // Camera does not support the output port action.
    return;
  }

  // Create output port action
  NewActionConfiguration newAction = new NewActionConfiguration
  {
    TemplateToken = "com.axis.action.unlimited.io.toggle",
    Parameters = new ActionParameters
    {
```

```
    Parameter = new[]
    {
      new ActionParameter { Name = "port", Value = "1" }
      new ActionParameter { Name = "state", Value = "high" }
    }
  }
 };

 // Add the action to the camera.
 string actionId = myActionService.AddActionConfiguration(newAction);
}
catch (Exception e)
{
 HandleException(e);
}
```

2.4 Recipients

A **recipient** is a network resource that can receive data, for example video clips or notification messages. Examples: network share, FTP server, email address.

2.4.1 Recipient Templates

A **recipient template** defines a recipient type. The template lists the parameters used to configure a recipient of the given type.

Supported operations:

- `GetRecipientTemplates` – list the recipient templates supported by the Axis product

For available recipient templates, see *Recipient Templates, on page 45*.

2.4.2 Recipient Configurations

A **recipient configuration** specifies the parameters for a recipient. A recipient configuration is created by specifying a recipient template and a list of parameters. Recipient configurations are optional and are only used to store the parameters on the Axis product. The recipient configurations are not used directly in action configurations.

Supported operations:

- `AddRecipientConfiguration` – create a new recipient configuration
- `GetRecipientConfigurations` – list all stored recipient configurations
- `RemoveRecipientConfiguration` – remove a recipient configuration

2.4.3 Using Recipients in an Action Configuration

The example in this section shows how to use a recipient in an action configuration. The action and the recipient are both defined by the action template. For example, the template `com.axis.action.fixed.notification.http` defines the action Send Notification and the recipient HTTP.

The action parameters and the recipient parameters are provided as `ActionParameters` in the `NewActionConfiguration` operation. In this example, parameters `message` and `parameters` belong to the Send Notification action while the remaining parameters belong to the HTTP recipient.

For more information about the action and the recipient templates, see *Send Notification Action, on page 43* and *HTTP Recipient, on page 45*.

Note

All parameters must be specified; if a parameter should not be used, its value should be set to an empty string.

The example uses `myActionService` created in section *Identification Using Entry Service, on page 8*.

```
try
{
  // Create send notification action
  NewActionConfiguration newAction = new NewActionConfiguration
  {
    TemplateToken = "com.axis.action.fixed.notification.http",
    Parameters = new ActionParameters
    {
      Parameter = new[]
      {
        new ActionParameter{ Name = "message", Value = "The message to send" },
        new ActionParameter{ Name = "parameters", Value = "" },
        new ActionParameter{ Name = "upload_url", Value = "http://ip_address/cgi-
bin/notify.cgi" },
        new ActionParameter{ Name = "login", Value = "myusername" },
        new ActionParameter{ Name = "password", Value = "mypassword" },
        new ActionParameter{ Name = "proxy_host", Value = "" },
        new ActionParameter{ Name = "proxy_port", Value = "" },
        new ActionParameter{ Name = "proxy_login", Value = "" },
        new ActionParameter{ Name = "proxy_password", Value = "" },
        new ActionParameter{ Name = "qos", Value = "0" }
      }
    }
  };
  // Add action to the camera.
  string actionId = myActionService.AddActionConfiguration(newAction);
}
catch (Exception e)
{
  HandleException(e);
}
```

2.5 Action Rules

An **action rule** specifies how and when the Axis product performs an action. Example: record video when motion is detected outside office hours.

An action rule consists of:

- a start event
- one or more conditions
- a primary action

Note

Fallback actions are not supported.

The primary action will be executed when the start event occurs and all specified conditions are fulfilled. The start event can be omitted. Conditions can also be omitted, but either a start event or at least one condition must be specified. The action will be stopped when any of the conditions is no longer fulfilled.

Note

Fixed actions can be used with any combination of start events and conditions. Unlimited actions require at least one condition to prevent the action from running indefinitely; the start event may however be omitted.

The start event and the conditions are specified using `TopicExpression` and `MessageContent` filters. See *Event Topic Tree Syntax, on page 9*. All mandated dialects specified in ONVIF™ Core Specification 1.02 are supported.

The start event can be any valid filter expression, as defined by ONVIF™ Core Specification 1.02.

Each condition should be specified by a separate filter expression which must describe one stateful event. Stateless events cannot be used as conditions. If the provided filter does not match any stateful events or matches multiple stateful events, the `AddActionRule` request will fail with `InvalidconditionFilterFault` reply.

The primary action is specified using an action configuration as described in *Setting Up an Action Rule*. Fallback actions are not supported.

Supported operations:

- `AddActionRule` — create a new action rule
- `GetActionRules` — list all stored action rules
- `RemoveActionRule` — remove an action rule

2.5.1 Setting Up an Action Rule

This example shows how to set up an action rule that plays an audio clip (alarm.au) if tampering is detected while Input 1 is active.

The example uses `myActionService` created in section *Identification Using Entry Service, on page 8*.

Start event: Tampering	Event: `tns1:VideoSource/tnsaxis:Tampering`	See *Camera Tampering Event, on page 30*.
Condition: Input 1 active	Event: `tns1:Device/tnsaxis:IO/tnsaxis:Port`	See *Digital Input Event, on page 24*.
Primary action: Play audio clip	Action template: `com.axis.action.fixed.play.audioclip`	See *Play Audio Clip Action, on page 40*.

The action rule is created in two steps:

1. The action configuration is created using `AddActionConfiguration`.
2. The action rule is created using `AddActionRule`. The Action Configuration ID `actionId` returned by `AddActionConfiguration` specifies the primary action.

```
try
{
  // Verify that the camera supports the action.
  ActionTemplate[] actiontemplates = myActionService.GetActionTemplates();
  if (actiontemplates.Any(
    template => template.TemplateToken == "com.axis.action.fixed.play.audioclip")
== false)
  {
    // Camera does not support the play audio clip action.
    return;
  }

  // Create play audio clip action
  NewActionConfiguration newAction = new NewActionConfiguration
  {
    TemplateToken = "com.axis.action.fixed.play.audioclip",
    Parameters = new ActionParameters
    {
      Parameter = new[]
      {
        new ActionParameter { Name = "location", Value = "etc/audioclips/alarm.au" }
      }
    }
  };
```

```
  // Add the action to the camera.
  string actionId = myActionService.AddActionConfiguration(newAction);

  // Create action rule
  NewActionRule newActionRule = new NewActionRule
  {
    Name = "Play audio clip on tampering",
    StartEvent = FormatTopicExpression("tns1:VideoSource/tnsaxis:Tampering"),
    Conditions = new[]
    {
      FormatTopicExpression(
        "tns1:Device/tnsaxis:IO/tnsaxis:Port",
        "boolean(//SimpleItem[@Name=\"port\" and @Value=\"1\"]) and "
        + "boolean(//SimpleItem[@Name=\"state\" and @Value=\"1\"])")
      },
    PrimaryAction = actionId,
    Enabled = true
  };

  // Add the action rule to the camera.
  string actionRuleId = myActionService.AddActionRule(newActionRule);
}
catch (Exception e)
{
  HandleException(e);
}
```

2.6 Scheduled Events

A **scheduled event** is active during specific time periods defined by an iCalendar schedule. Scheduled events can be used in action rules and in notification subscriptions.

There are two types of schedules:

- An **interval schedule** emits a stateful event at the scheduled start time. The event will remain active until the scheduled end time. The schedule can be repeated according to a recurring rule. Example: A schedule named "Office Hours" which is active from 9 a.m. to 5 p.m. Monday through Friday.
- A **pulse schedule** emits a stateless event at the scheduled start time and can be repeated according to a recurring rule. Pulse schedules are typically used for recurrences, for example to run an action once every hour.

Supported operations:

- `AddScheduledEvent` — create a scheduled event
- `GetScheduledEvents` — list scheduled events
- `RemoveScheduledEvent` — remove a scheduled event

2.6.1 Schedules

A schedule is specified by providing a start time, an end time and a recurring rule. The schedule format is designated by a wrapper element whose name defines the syntax and semantics of the schedule description format. The supported schedule description format is a subset of the iCalendar format specified in RFC 5545.

```
<aev:ICalendar>
 DTSTART:<start date>
 DTEND:<end date>
 RRULE:<recurring rule>
</aev:ICalendar>
```

<start date> is the start date and time of the first occurrence of the event. Both date and time are required and should be specified according to ISO 8601. The time is the product's local time; time zone information is not supported.

For interval schedules, <end date> is the end date and time of the first occurrence of the event. Use the same format as for <start date>. For pulse schedules, <end date> should be omitted.

<recurring rule> specifies the repetitions of the event. The following subset of the iCalendar specification is supported:

```
RRULE:FREQ=YEARLY[;BYMONTH=<1-12>,...][;INTERVAL=<1-*>]
RRULE:FREQ=MONTHLY[;BYDAY=[[-]<0-5>]<MO|TU|WE|TH|FR|SA|SU>,...][;INTERVAL=<1-*>]
RRULE:FREQ=WEEKLY[;BYDAY=<MO|TU|WE|TH|FR|SA|SU>,...][;INTERVAL=<1-*>]
RRULE:FREQ=DAILY[;INTERVAL=<1-*>]
RRULE:FREQ=HOURLY[;INTERVAL=<1-*>]
RRULE:FREQ=MINUTELY[;INTERVAL=<1-*>]
RRULE:FREQ=SECONDLY[;INTERVAL=<1-*>]
```

2.6.2 Create a Scheduled Event

This examples shows how to create a scheduled event with an interval schedule that runs from 6 am to 9 am, Monday through Friday.

The example uses myEventService created in section *Identification Using Entry Service, on page 8*.

```
try
{
  // Create the scheduled event.
  NewScheduledEvent scheduledEvent = new NewScheduledEvent
  {
    Name = "My schedule",
    Schedule = new Schedule
    {
      ICalendar = new ICalendar
      {
        Value =
          "DTSTART:20111212T06:00 " +
          "DTEND:20111212T09:00 " +
          "RRULE:FREQ=WEEKLY;BYDAY=MO,TU,WE,TH,FR"
      }
    }
  };

  // Add the scheduled event.
  string scheduledEventId = myEventService.AddScheduledEvent(scheduledEvent);
}
catch (Exception e)
{
  HandleException(e);
}
```

2.7 Virtual Input Events

A **virtual input event** is emitted when the state of a virtual input port is changed. See the Virtual input API. Virtual input ports can be used in the same way as (physical) input ports. A virtual input port is in one of the states: true and false.

Use ChangeVirtualInputState to emit a virtual input event. An emitted virtual input event is stateful.

3 Event Declarations

3.1 UserAlarm/Recurring Events

3.1.1 Scheduled Event

A Scheduled Event triggers actions during specific time periods.

For information about supported schedule formats, see *Schedules, on page 16*.

Topic

Name: `tns1:UserAlarm/tnsaxis:Recurring/tnsaxis:Interval`
Type: Stateful
Nice name: `Scheduled event`

Source instance

Nice name: `Schedule`
Type: string
Name: `id`

Value	Nice name
`com.axis.schedules.weekdays`	`Weekdays`
`com.axis.schedules.office_hours`	`Office Hours`
`com.axis.schedules.after_hours`	`After Hours`
`com.axis.schedules.weekends`	`Weekends`
`com.axis.schedules.genid.id-0`	`[User-defined name]`

Data instance

Nice name: Active
Type: boolean
Name: `active`
isPropertyState: `true`

3.1.2 Recurring Event

A Recurring Event triggers actions repeatedly, for example every 5 minutes.

For information about supported schedule formats, see *Schedules, on page 16*.

Topic

Name: `tns1:UserAlarm/tnsaxis:Recurring/tnsaxis:Pulse`
Type: Stateless
Nice name: `Recurring pulse`

Source instance

Nice name: `Schedule`
Type: string
Name: `id`

Value	Nice name
com.axis.schedules.genid.id-0	[User-defined name]

Data instance

Not applicable

3.2 Device Status Events

3.2.1 System Ready Event

The System Ready event is `true` when the product has been started and all services are running. The event can for example be used to detect that a product has been restarted.

Topic

Name: `tns1:Device/tnsaxis:Status/tnsaxis:SystemReady`
Type: Stateful
Nice name: `System ready`

Source instance

Not applicable

Data instance

Nice name: `Ready`
Type: boolean
Name: `ready`
isPropertyState: `true`

3.2.2 PTZ Ready Event

The PTZ Ready event is `true` when the PTZ functionality is ready to be used. The event can for example be used to detect that PTZ is ready to use after restart or after a PTZ driver has been installed.

For products with multiple view areas or multiple video channels, there is one event for each view area or video channel. The `channel` element in SourceInstance specifies the view area or video channel.

Topic

Name: `tns1:PTZController/tnsaxis:PTZReady`
Type: Stateful
Nice name: `PTZ ready`

Source instance

Nice name: `Channel`
Type: integer
Name: `channel`

Data instance

Nice name: Ready
Type: boolean
Name: `ready`
isPropertyState: `true`

3.2.3 Temperature Above Event

The Temperature Above event is `true` when the temperature is above the operating range of the product.

Topic

Name: `tns1:Device/tnsaxis:Status/tnsaxis:Temperature/tnsaxis:Above`
Type: Stateful
Nice name: `Above operating temperature`

Source instance

Not applicable

Data instance

Nice name: Above range
Type: boolean
Name: `sensor_level`
isPropertyState: `true`

3.2.4 Temperature Below Event

The Temperature Below event is `true` when the temperature is below the operating range of the product.

Topic

Name: `tns1:Device/tnsaxis:Status/tnsaxis:Temperature/tnsaxis:Below`
Type: Stateful
Nice name: `Below operating temperature`

Source instance

Not applicable

Data instance

Nice name: Below range
Type: boolean
Name: `sensor_level`
isPropertyState: `true`

3.2.5 Temperature Inside Event

The Temperature Inside event is `true` when the temperature is inside the operating range of the product.

Topic

Name: `tns1:Device/tnsaxis:Status/tnsaxis:Temperature/tnsaxis:Inside`
Type: Stateful
Nice name: `Within operating temperature`

Source instance

Not applicable

Data instance

Nice name: Within range
Type: boolean
Name: `sensor_level`
isPropertyState: `true`

3.2.6 Temperature Outside Event

The Temperature Outside event is `true` when the temperature is outside the operating range of the product.

Topic

Name: `tns1:Device/tnsaxis:Status/tnsaxis:Temperature/tnsaxis:Above_or_below`
Type: Stateful
Nice name: `Above or below operating temperature`

Source instance

Not applicable

Data instance

Nice name: Above or below range
Type: boolean
Name: `sensor_level`
isPropertyState: `true`

3.2.7 Live Stream Accessed Event

The Live Stream Accessed event is `true` when the Axis product sends a live stream (video, audio or metadata) to a client .

Topic

Name: `tns1:Video/tnsaxis:Source/tnsaxis:LiveStreamAccessed`
Type: Stateful
Nice name: `Live stream accessed`

Source instance

Not applicable

Data instance

Nice name: Accessed
Type: boolean
Name: `accessed`
isPropertyState: `true`

3.3 Hardware Failure Events

3.3.1 Fan Failure Event

The Fan Failure event is `true` if the product's fan does not work. The fan can be built into the product or be a rack fan connected to a video encoder blade. For products with multiple fans, there is one event for each fan.

Topic

Name: `tns1:Device/tnsaxis:HardwareFailure/tnsaxis:FanFailure`
Type: Stateful
Nice name: `Fan failure`

Source instance

Nice name: `Fan`
Type: integer
Name: `fan`

Value	Nice name
0	–
1	–
...	–
[number of fans]	–

Data instance

Nice name: Fan failure
Type: boolean
Name: fan_failure
isPropertyState: true

3.3.2 Power Supply Failure Event

The Power Supply Failure event is true if the power supply does not work.

Topic

Name: tns1:Device/tnsaxis:HardwareFailure/tnsaxis:PowerSupplyFailure
Type: Stateful
Nice name: Power supply failure

Source instance

Nice name: Power
Type: integer
Name: power

Value	Nice name
0	–
1	–
...	–
[number of power supplies]	–

Data instance

Nice name: Power critical
Type: boolean
Name: power_critical
isPropertyState: true

3.3.3 Temperature Critical Event

The Temperature Critical event is true if the temperature measured by the temperature sensors exceeds the operating range of the product.

Topic

Name: tns1:Device/tnsaxis:HardwareFailure/tnsaxis:TemperatureCritical
Type: Stateful
Nice name: Temperature critical

Source instance

Nice name: –
Type: integer
Name: `temperature`

Value	Nice name
`0`	–
`1`	–
`...`	–
`[number of temperature sensors]`	–

Data instance

Nice name: Temperature is critical
Type: boolean
Name: `temperature_critical`
isPropertyState: `true`

3.4 Network Events

3.4.1 Network Lost Event

The Network Lost event is `true` if network connection is lost. For products with multiple network interfaces, for example wired and wireless, there is one event for each interface and one `Any` event which is `true` if any of the interfaces looses network connection.

The event can for example be used to start recording to the SD card if network is lost.

Topic

Name: `tns1:Device/tnsaxis:Network/tnsaxis:Lost`
Type: Stateful
Nice name: `Network lost`

Source instance

Nice name: `Interface`
Type: string
Name: `interface`

Value	Nice name
`Any`	–
`eth0`	–
`eth1`	–
`...`	–
`ethn [n = number of network interfaces + 1]`	–

Data instance

Nice name: Lost
Type: boolean
Name: `lost`
isPropertyState: `true`

3.5 Device I/O Events

3.5.1 Digital Input Event

The Digital Input event is `true` if the digital I/O port configured as input is active. There is one event for each input port. The `Any` event is `true` if any of the input ports is active.

Topic

Name: `tns1:Device/tnsaxis:IO/tnsaxis:Port`
Type: Stateful
Nice name: `Digital input port`

Source instance

Nice name: `Port`
Type: integer
Name: `port`

Value	Nice name
`-1`	`Any`
`0`	User-defined name (default `Input 1`)
`1`	User-defined name (default `Input 2`)
`...`	`...`
`n`	User-defined name (default `Input n+1`)

Data instance

Nice name: Active
Type: boolean
Name: `state`
isPropertyState: `true`

3.5.2 Virtual Input Event

The Virtual Input event is `true` if the virtual input is active. There is one event for each virtual input.

Topic

Name: `tns1:Device/tnsaxis:IO/tnsaxis:VirtualInput`
Type: Stateful
Nice name: `Virtual input`

Source instance

Nice name: `Input`
Type: integer
Name: `port`

Value	Nice name
`1`	`Input 1`
`2`	`Input 2`
`...`	`...`
`32`	`Input 32`

Data instance

Nice name: Active
Type: boolean
Name: `active`
isPropertyState: `true`

3.5.3 Manual Trigger Event

The Manual Trigger event is `true` if the manual trigger is active. The manual trigger can be activated using `axis-cgi/io/virtualinput.cgi` or using the button in the product's Live View page. For products with multiple view areas or multiple video channels, there is one manual trigger for each view area or video channel. The `channel` element in SourceInstance specifies the view area or video channel.

Topic

Name: `tns1:Device/tnsaxis:IO/tnsaxis:VirtualPort`
Type: Stateful
Nice name: `Manual trigger`

Source instance

Nice name: `Channel`
Type: integer
Name: `port`

Value	Nice name
`1`	–
`2`	–
`...`	–
`number of channels/view areas`	–

Data instance

Nice name: Active
Type: boolean
Name: `state`
isPropertyState: `true`

3.6 Device Sensor Events

3.6.1 PIR Sensor Event

The PIR Sensor event is `true` when the PIR sensor detects motion.

Topic

Name: `tns1:Device/tnsaxis:Sensor/tnsaxis:PIR`
Type: Stateful
Nice name: `PIR sensor`

Source instance

Nice name: `Sensor`
Type: integer
Name: `sensor`

Value	Nice name
0	–
1	–
...	–
n = number of PIR sensors - 1	–

Data instance

Nice name: Active
Type: boolean
Name: state
isPropertyState: true

3.7 Device Casing Events

3.7.1 Open Casing Detection Event

The Open Casing Detection Event is true when the casing of a connected external device, such as a junction box or cabinet, is removed or opened. This can for example be used to send notifications of maintenance or unauthorized tampering.

Topic

Name: tns1:Device/tnsaxis:Casing/tnsaxis:Open
Type: Stateful
Nice name: Casing Open

Source instance

Not applicable

Value	Nice name
JunctionBox	Junction box

Data instance

Nice name: Open
Type: boolean
Name: Open
isPropertyState: true

3.8 Connector Events

3.8.1 Audio Connector Event

The Audio Connector event is true if equipment is connected to the audio connector.

Topic

Name: tns1:Connector/tnsaxis:Source
Type: Stateful
Nice name: Audio connector

Source instance

Nice name: Connector
Type: integer
Name: connector

Value	Nice name
0	–
1	–
...	–
[number of connectors]	–

Data instance

Nice name: Connected
Type: boolean
Name: connected
isPropertyState: true

3.9 Video Encoder Events

3.9.1 Video Connected Event

The Video Connected event is available in video encoders. The event is true when the video encoder receives a video signal from the analog camera. There is one event for each video channel.

Topic

Name: tns1:VideoEncoder/tnsaxis:Connections
Type: Stateful
Nice name: Video connected

Source instance

Nice name: Channel
Type: integer
Name: channel

Value	Nice name
1	–
2	–
...	–
[number of video channels]	–

Data instance

Nice name: Connected
Type: boolean
Name: connected
isPropertyState: true

3.10 PTZ Controller Events

3.10.1 PTZ Preset Reached Event

The PTZ Preset Reached event is `true` when the camera stops at a preset position. Preset positions are defined for each video channel or view area. There is one event for each preset position. The `Any` event is `true` if the camera stops at any of the preset positions defined on that channel or view area.

Topic

Name: `tns1:PTZController/tnsaxis:PTZPresets/tnsaxis:Channel_X`
Type: Stateful
Nice name: `PTZ preset reached on channel X`

Source instance

Nice name: `Preset token`
Type: integer
Name: `PresetToken`

Value	Nice name
`-1`	`Any`
`1`	`Home`
`2`	User-defined name
`...`	`...`
`number of channels/view areas`	User-defined name

Data instance

Nice name: Preset reached
Type: boolean
Name: `on_preset`
isPropertyState: `true`

3.10.2 PTZ Moving Event

The PTZ Movement event is `true` when the camera is moving due to a PTZ operation.

The event can for example be used to prevent motion detection from being triggered while the camera moves due to a PTZ operation.

Topic

Name: `tns1:PTZController/tnsaxis:Move/tnsaxis:Channel_X`
Type: Stateful
Nice name: `PTZ movement on channel X`

Source instance

Nice name: `PTZ channel`
Type: integer
Name: `PTZConfigurationToken`

Value	Nice name
`X`	`Channel X`

Data instance

Nice name: Moving
Type: boolean
Name: `is_moving`
isPropertyState: `true`

3.10.3 PTZ Control Queue Event

The PTZ Control Queue event shows the user who is controlling the PTZ functionality.

Topic

Name: `tns1:PTZController/tnsaxis:ControlQueue`
Type: Stateful
Nice name: `PTZ control queue`

Source instance

Nice name: `PTZ channel`
Type: integer
Name: `PTZConfigurationToken`

Value	Nice name
1	Channel 1
2	Channel 2
...	...
number of channels	Channel n

Data instance

Nice name: Queue owner
Type: string
Name: `queue_owner`
isPropertyState: `true`

3.10.4 Autotracking Event

The Autotracking event is `true` when autotracking is active.

Topic

Name: `tns1:PTZController/tnsaxis:AutoTracking`
Type: Stateful
Nice name: `Autotracking`

Source instance

Nice name: `Channel`
Type: integer
Name: `channel`
Table valid for multichannel products.

Value	Nice name
1	–
2	–
...	–
n = number of video channels	–

For single channel products, the channel is always 0.

Data instance

Nice name: Tracking motion
Type: boolean
Name: `tracking`
isPropertyState: `true`

3.11 Video Analytics Events

3.11.1 Motion Detection Event

The Motion Detection event is `true` when motion is detected in a motion detection window.

Topic

Name: `tns1:VideoAnalytics/tnsaxis:MotionDetection`
Type: Stateful
Nice name: `Motion detection`

Source instance

Nice name: `Window`
Type: integer
Name: `window`

Value	Nice name
0	User-defined name (default: `[0] DefaultWindow`)
1	User-defined name (default: `[1] DefaultWindow`)
...	...
n = maximum number of windows + 1	User-defined name (default: `[n] DefaultWindow`)

Data instance

Nice name: motion detected
Type: boolean
Name: `motion`
isPropertyState: `true`

3.12 Video Source Events

3.12.1 Camera Tampering Event

The Camera Tampering event is emitted when the camera is redirected, covered or de-focused.

Topic

Name: `tns1:VideoSource/tnsaxis:Tampering`
Type: Stateless
Nice name: `Camera tampering`

Source instance

Nice name: `Channel`
Type: integer
Name: `channel`

Data instance

Nice name: Tampering
Type: integer
Name: `tampering`

Value	Nice name
`1`	–

3.12.2 Day/Night Vision Mode Event

The Day/Night Vision Mode event is `true` when the product is in day mode (IR cut filter is on).

This event can for example be used to control an external IR light connected to the product's digital output port.

Topic

Name: `tns1:VideoSource/tnsaxis:DayNightVision`
Type: Stateful
Nice name: `Day night vision`

Source instance

Nice name: `Video source configuration token`
Type: integer
Name: `VideoSourceConfigurationToken`

Value	Nice name
`1`	–
`2`	–
`...`	–
`n = number of video channels`	–

Data instance

Nice name: day
Type: boolean
Name: `day`
isPropertyState: `true`

3.13 Audio Source Events

3.13.1 Audio Detection Event

The Audio Detection event is `true` when the sound level rises above, falls below or passes the audio alarm level.

The audio alarm level is defined by parameter `Audio.A#.AlarmLevel`.

Topic

Name: `tns1:AudioSource/tnsaxis:TriggerLevel`
Type: Stateful
Nice name: `Audio detection`

Source instance

Nice name: `Channel`
Type: integer
Name: `channel`

Value	Nice name
`1`	–
`2`	–
`...`	–
`n = number of audio channels`	–

Data instance

Nice name: `Above alarm level`
Type: boolean
Name: `triggered`
isPropertyState: `true`

3.14 Storage Events

3.14.1 Recording Ongoing Event

Recording Ongoing Event indicates if there are any ongoing edge storage recordings on an Axis device. The event is triggered when the device is recording to a storage media.

Topic

Name: `tnsaxis:Storage/tnsaxis:Recording`
Type: Stateful
Nice name: `Recording ongoing`

Source instance

none

Data instance

Nice name: Recording ongoing
Type: boolean
Name: recording
isPropertyState: true

3.14.2 Storage Disruption Detection

The Storage disruption event is true when there is/are problem(s) facing the storage device. The event comprises the following failure situations: storage becomes unavailable, storage becomes read only, storage is removed, storage becomes full, storage is locked, storage has a read/write error.

This state event is generated with the data instance Disruption set to TRUE if the storage device is unmounted, a read/write error occur, when the storage is mounted in "read only" mode, when a storage device is lost or removed or when the storage device is full or locked.

This state event is generated with the data instance Disruption set to FALSE if a device is mounted successfully with read/write permissions, a storage full state is followed by successful removal of surplus data to the level that the storage full state is cleared or a storage device that was locked has been unlocked using the web GUI or the VAPIX API.

For more information about what the problem is, use checkdisk or listdisk from Storage API document

Topic

Name: `tnsaxis:Storage/tnsaxis:Disruption`
Type: Stateful
Nice name: `Storage Disruption`

Source instance

Name:: disk_id
Type: string
Nice name: Disk

Data instance

Nice name: Storage Disruption
Type: boolean
Name: `disruption`
isPropertyState: `true`

3.15 System Message Events

3.15.1 PTZ Error Event

The PTZ Error event is emitted if the PTZ functionality is not working as expected.

Topic

Name: `tns1:Device/tnsaxis:SystemMessage/tnsaxis:PTZError`
Type: Stateless
Nice name: `PTZ error`

Source instance

Nice name: `Channel`
Type: integer
Name: `channel`

Data instance

Nice name: Description
Type: string
Name: `description`
Value: String that describes the error.

3.15.2 Action Failed Event

The Action Failed event is emitted if an action cannot be started. The event is emitted for both primary actions and fallback actions.

Note

Fallback actions are not supported.

Topic

Name: `tns1:Device/tnsaxis:SystemMessage/tnsaxis:ActionFailed`
Type: Stateless
Nice name: `Action failed`

Source instance

Not applicable

Data instance

Nice name: Description
Type: string
Name: `description`
Value: String that describes the error.

3.16 ACAP Events

3.16.1 Application Event - C Applications

Events from AXIS Camera Application Platform (ACAP) applications are defined by the application. For applications written in C, events are defined using the library `libs/acapevent`.

Topic

Name: `tns1:CameraApplicationPlatform/tnsaxis:`[ApplicationName]
Type: Application dependent
Nice name: Defined by the application

Source instance

Nice name: –
Type: string
Name: `app`

Value	Nice name
[ApplicationName]	

Data instance

Nice name: –
Type: string
Name: `event`
isPropertyState: `true`. Omitted if the event is stateless.

Value	Nice name
[String with the name of the event.]	

3.16.2 Application Event - Lua Applications

Events from AXIS Camera Application Platform (ACAP) applications are defined by the application. For applications written in Lua, events are defined from the application package's xml configuration file.

Topic

Name: `tns1:RuleEngine/tnsaxis:`[ApplicationName]`/tnsaxis:`[EventName]
Type: Application dependent
Nice name: Defined by the application

Source instance

Nice name: Defined from application configuration
Type: boolean or string (defined from application configuration)
Name: Defined from application configuration

Value	Nice name
Defined from application configuration	

Data instance

Nice name: Defined from application configuration
Type: boolean or string (defined from application configuration)
Name: Defined from application configuration
isPropertyState: `true`. Omitted if the event is stateless.

4 Action Templates

This section describes available action templates. Each template lists the parameters needed to configure the action. All parameters must be specified; if a parameter should not be used, its value should be set to an empty string.

Note

Supported templates are product and firmware dependent. Use `GetActionTemplates` to list the action templates supported by the Axis product.

4.1 Activate Light Action

The Activate Light action is used to turn on a light built into the products, for example white LEDs and infrared LEDs.

This action can be run as:

- **fixed action** – keep the light lit for a predefined time (defined by parameter `duration`).
- **unlimited action** – keep the light lit as long as all event conditions are fulfilled.

Action ID: `com.axis.action.fixed.light`
Action ID: `com.axis.action.unlimited.light`

Parameter	Valid values	Description
`source`	Unsigned integer	The Light ID.
`level`[1]	`0 ... 100`	The light intensity.
`fade_in`[1]	`0` `1`	`0` = Set light directly to the set `level` `1` = Turn on the light gradually until the set `level` is reached.
`fade_out`[1]	`0` `1`	`0` = Turn off the light abruptly `1` = Turn off the light gradually
`frequency`[1]	`0 ... 100`	The strobe frequency in Hz. Set to `0` to disable.
`duration`	Unsigned integer	Fixed actions: Number of seconds to keep the light lit.

1. Product-dependent

4.2 Autotracking Action

The autotracking action is used to start autotracking from a preset position, that is, to go to the preset position `preset_name` and from that position follow a moving object within the camera's field of view.

The action can be run as

- **fixed action** – continue autotracking until there are no moving objects
- **unlimited action** – continue autotracking as long as all conditions are fulfilled

Example 1:
The action can for example be used in an action rule triggered by a signal from an I/O port. Consider a PTZ camera monitoring a number of doors with door sensors connected to the camera's I/O ports. When one of the doors is opened, the action rule is triggered and the camera moves to a preset position facing that door and starts follow a person or vehicle coming out of the door.

Action ID: `com.axis.action.fixed.motiontracking`
Action ID: `com.axis.action.unlimited.motiontracking`

Parameter	Valid values	Description
channel	Unsigned integer	The video channel.
goto_home_time-out	Unsigned integer	Fixed action: Number of seconds to wait before returning to the home position when tracking stops.
preset_name	String	The PTZ preset position from which tracking should start.

4.3 Day/Night Mode Action

The Day/Night Mode action is used to enable and disable the IR cut filter. The action is unlimited, that is, the IR cut filter remains in set state as long as all conditions are fulfilled. When the conditions are no longer fulfilled, the IR cut filter returns to the opposite state.

Action ID: com.axis.action.unlimited.ircutfilter

Parameter	Valid values	Description
active_state	yes no auto[1]	Active state. yes = The IR cut filter is enabled. no = The IR cut filter is disabled. auto = The IR cut filter is enabled and disabled automatically depending on the lighting conditions.
inactive_state	yes no auto[1]	Inactive state. yes = The IR cut filter is enabled. no = The IR cut filter is disabled. auto = The IR cut filter is enabled and disabled automatically depending on the lighting conditions.

1. Product dependent. Not supported by video encoders and mechanical PTZ cameras.

Note

For fixed cameras, the IR cut filter is controlled by the ImageSource.I0.DayNightIrCutFilter parameter. For mechanical PTZ cameras and for video encoders, the IR cut filter is controlled by the PTZ.Various.V1.IrCutFilter parameter.

4.4 Go to PTZ Preset Action

The Go to PTZ preset action is used to steer the camera view to a PTZ preset position. When the action is completed, the camera view will return to the home position.

This action can be run as:

- **fixed action** – stay at the preset position for a predefined time (defined by parameter home_timeout)
- **unlimited action** – stay at the preset position as long as all event conditions are fulfilled.

Action ID: com.axis.action.fixed.goto.preset
Action ID: com.axis.action.unlimited.goto.preset

Parameter	Valid values	Description
channel	Unsigned integer	Video channel
preset_name	String	Name of the PTZ preset to go to. Preset names can be retrieved from axis-cgi/com/ptz.cgi?query=presetposall
home_timeout	Unsigned integer or -1	Number of seconds to wait before returning to the home position. Use -1 if the camera view should not return to the home position.

4.5 Guard Tour Action

The Guard Tour action is used to start a guard tour (preset tour).

The action can be run as:

- **fixed action** – run the guard tour once
- **unlimited action** – keep the guard tour running as long as all conditions are fulfilled

Action ID: `com.axis.action.fixed.guardtour`
Action ID: `com.axis.action.unlimited.guardtour`

Parameter	Valid values	Description
`tour_id`	Unsigned integer	The guard tour to start. If the guard tour is defined by the `GuardTour.G#` parameter group, the `tour_id` is the number `#`.
`goto_home`	`0` `1`	`1` = Return to the home position when the action is finished. `0` = Stay at the current position when the action is finished.
`channel`	Unsigned integer	The view area in which the guard tour is defined. The view area is defined by the `Image.I#` parameter group where `#` is the view area number.

4.6 LED Control

The LED control action is used to flash product's LED indicator.

This action can be run as:

- **fixed action** – flash the LED during the time defined by parameter `duration`
- **unlimited action** – flash the LED as long as all event conditions are fulfilled

Action ID: `com.axis.action.fixed.ledcontrol`
Action ID: `com.axis.action.unlimited.ledcontrol`

Parameter	Valid values	Description
`led`	String	The LED name. Valid names are listed as `LedName` in the response from `axis-cgi/ledcontrol/getleds.cgi?schemaversion=1`
`color`	String	The color. Valid color names are listed as `ColorName` in the response from `axis-cgi/ledcontrol/getleds.cgi?schemaversion=1`
`duration`	Unsigned integer	Fixed actions: Number of seconds to keep flashing the LED.
`interval`	Unsigned integer	Number of milliseconds between flashes.

4.7 Output Port Action

The Output Port action is used to activate and inactive the product's output ports.

This action can be run as:

- **fixed action** – set the port state when action is triggered and return to the opposite state after the time defined by parameter `duration`

  unlimited action – set the port state when the action is triggered and return to the opposite state when event conditions are no longer fulfilled

Action ID: `com.axis.action.fixed.io.toggle`
Action ID: `com.axis.action.unlimited.io.toggle`

Parameter	Valid values	Description
`port`	Unsigned integer	The I/O port number of the output port to activate. The I/O ports are numbered starting from zero. The I/O port number is the `#` of parameter `IOPort.I#`.
`state`	`high` `low`	State to set the port to.
`duration`	Unsigned integer	Fixed actions: Number of seconds before returning to the opposite state. Use `duration = 0` if the port should remain in the set state.

4.8 Overlay Text Action

The Overlay Text action is used to display a dynamic text overlay. The action makes use of the modifier `#D`. By default, `#D` is an empty string. When the overlay text action is activated, `#D` is replaced by the text stored in parameter `text` (see table below).

To use this action, overlay text must be enabled in the video channel and the text string must contain the modifier `#D`. Use the parameter group `Image.I#.Text` to enabled and configure overlay text. See example below.

This action can be run as:

- **fixed action** – keep the overlay text during a predefined time (defined by parameter `duration`)
- **unlimited action** – keep the overlay text as long as all event conditions are fulfilled.

Action ID: `com.axis.action.fixed.set_overlay`
Action ID: `com.axis.action.unlimited.set_overlay`

Parameter	Valid values	Description
`text`	String	The text to display. The modifier `#D` will be replaced by this text when the action is active.
`channels`	String	Comma-separated list of the video channels where the overlay text should be displayed.
`duration`	Unsigned integer	Unlimited actions: Number of seconds to display the text.

Example 1:
This example shows how to use to the overlay text action to display the text "Active input ports: input 1" for 10 seconds when input port 1 becomes active. When the input port is inactive, the text "Active input ports:" is displayed.

First, enable overlay text and set the text to "Active input ports: #D".

```
http://myserver/axis-cgi/param.cgi?action=update&Image.I0.Text.TextEnabled=yes
http://myserver/axis-cgi/param.cgi?action=update
&Image.I0.Text.String=Active%20input%20ports%3A%20%23D
```

Next, create an action configuration and an action rule. This example uses `myActionService` created in section *Identification Using Entry Service, on page 8*.

```
try
{
  // Verify that the camera supports the action.
  ActionTemplate[] actiontemplates = myActionService.GetActionTemplates();
  if (actiontemplates.Any(
    template => template.TemplateToken == "com.axis.action.fixed.set_overay") ==
false)
  {
    // Camera does not support the overlay text action.
    return;
  }

  // Create overlay text action
  NewActionConfiguration newAction = new NewActionConfiguration
  {
    TemplateToken = "com.axis.action.fixed.set_overlay",
    Parameters = new ActionParameters
    {
      Parameter = new[]
      {
        new ActionParameter { Name = "text", Value = "input 1" }
        new ActionParameter { Name = "channels", Value = "1" }
        new ActionParameter { Name = "duration", Value = "10" }
      }
    }
  };

  // Add the action to the camera.
  string actionId = myActionService.AddActionConfiguration(newAction);

  // Create action rule
  NewActionRule newActionRule = new NewActionRule
  {
    Name = "Display overlay text",
    StartEvent = FormatTopicExpression("tns1:Device/tnsaxis:IO/tnsaxis:Port",
       "boolean(//SimpleItem[@Name=\"port\" and @Value=\"1\"]) and "
       + "boolean(//SimpleItem[@Name=\"state\" and @Value=\"1\"])")
    PrimaryAction = actionId,
    Enabled = true
  };

  // Add the action rule to the camera.
  string actionRuleId = myActionService.AddActionRule(newActionRule);
}
catch (Exception e)
{
  HandleException(e);
}
```

4.9 Play Audio Clip Action

The Play Audio Clip action is used to play an audio clip. The audio clip is an uploaded or recorded audio file which can be played by speakers built into or connected to the Axis product.

Action ID: `com.axis.action.fixed.play.audioclip`

Parameter	Valid values	Description
location	Local file path	The audio clip location (path and file name) on the Axis product. The location is stored in parameter MediaClip.M#.Location. Default value: /etc/audioclips/MediaClip.M#.au

Note

The # in MediaClip.M# identifies the clip and is replaced by an integer starting from 0.

4.10 Recorded Tour Action

The Recorded Tour action is used to start a recorded tour. The action is unlimited; the recorded tour will keep running as a long as all conditions are fulfilled.

Action ID: com.axis.action.unlimited.recordedtour

Parameter	Valid values	Description
tour_id	Unsigned integer	The ID of the recorded tour to start. IDs of configured recorded tours can be retrieved using axis-cgi/recordedtour/list.cgi
goto_home	0 1	1 = Return to the home position when the action is finished. 0 = Stay at the current position when the action is finished.

4.11 Record Video Action

The Record Video action records video to a storage device, for example an SD card or a network share. Refer to the Edge Storage API for information about storage devices and how to retrieve recordings.

This action can be run as:

- **fixed action** – video is recorded during a pre-event and post-event time
- **unlimited action** – video is recorded during a pre-event time, while the event is running and during a post-event time

Action ID: com.axis.action.fixed.recording.storage
Action ID: com.axis.action.unlimited.recording.storage

Parameter	Valid values	Description
stream_options	URL-encoded string	List of stream parameters such as resolution, compression etc. All parameters supported by RTSP and HTTP stream requests can be used. These parameters are listed in the Video Streaming API. Example: stream_options=resolution%3D640x480%26compression%3D30%26rotation%3D180
pre_duration	Unsigned integer	Pre-trigger time in milliseconds. Specify the number of milliseconds to include from the time immediately before the event.
post_duration	Unsigned integer	Post-trigger time in milliseconds. Specify the number of milliseconds to include from the time immediately after the event.
storage_id	Disk ID	Disk ID of the storage device to record to. Use axis-cgi/disks/list.cgi?diskid=all to list the disks supported by the product. See Edge Storage API for information about storage disks.

4.12 Send Images Action

The Send Images action is used to send a images (JPEG) to a recipient.

This action can be run as:

- **fixed action** – images are sent during a pre-event and post-event time.
- **unlimited action** – images are sent during a pre-event time, while the event is running and during a post-event time.

Recipient	Recipient ID	Action ID
HTTP	`com.axis.recipient.http`	`com.axis.action.fixed.send_images.http` `com.axis.action.unlimited.send_images.http`
HTTPS	`com.axis.recipient.https`	`com.axis.action.fixed.send_images.https` `com.axis.action.unlimited.send_images.h-ttps`
SMTP	`com.axis.recipient.smtp`	`com.axis.action.fixed.send_images.smtp` `com.axis.action.unlimited.send_images.smtp`
FTP	`com.axis.recipient.ftp`	`com.axis.action.fixed.send_images.ftp` `com.axis.action.unlimited.send_images.ftp`
Network Share	`com.axis.recipient.networksh-are`	`com.axis.action.fixed.send_images.netwo-rkshare` `com.axis.action.unlimited.send_images.n-etworkshare`

The following parameters should be specified:

Parameter	Valid values	Description
`stream_options`	URL-encoded string	List of stream parameters such as resolution, compression etc. All parameters supported by RTSP and HTTP stream requests can be used. These parameters are listed in the Video Streaming API. Example: `stream_options=resolution%3D640x480%26compression%3D30%26rotation%3D180`
`pre_duration`	Unsigned integer	Pre-trigger time in milliseconds. Specify the number of milliseconds to include from the time immediately before the event.
`post_duration`	Unsigned integer	Post-trigger time in milliseconds. Specify the number of milliseconds to include from the time immediately after the event.
`create_folder`	String	Create a folder on the recipient to store uploaded files in. Modifiers can be used, see *Modifiers, on page 48*.
`filename`	String	File name for uploaded files. Modifiers can be used, see *Modifiers, on page 48*.
`max_sequence_number`	Unsigned integer	The maximum sequence number to use in file names if modifier `#s` is used. `0` = no maximum
`max_images`	Unsigned integer	Maximum number of images to upload.
`images_per_mail`	Unsigned integer	SMTP recipients: The maximum number of images per email.
`custom_params`	String	HTTP and HTTPS recipients: Additional CGI arguments.

subject	String	SMTP recipients: Email subject. Modifiers can be used, see *Modifiers, on page 48*.
message	String	SMTP recipients: Email message. Modifiers can be used, see *Modifiers, on page 48*.

4.13 Send Notification Action

The Send Notification action is used to send a notification message to a recipient. The notification message can be formatted using modifiers, see *Modifiers, on page 48*.

When using a TCP recipient the action can be run as:

- **fixed action** – messages are sent during a pre-event and post-event time
- **unlimited action** – messages are sent during a pre-event time, while the event is running and during a post-event time

When using HTTP and SMTP recipient, the action is always fixed.

Recipient	Recipient ID	Action ID
HTTP	com.axis.recipient.http	com.axis.action.fixed.notification.http
SMTP	com.axis.recipient.smtp	com.axis.action.fixed.notification.smtp
TCP	com.axis.recipient.tcp	com.axis.action.fixed.notification.tcp com.axis.action.unlimited.notification.tcp

The following parameters should be specified:

Parameter	Valid values	Description
message	String	Notification message. Spaces are allowed. Modifiers can be used, see *Modifiers, on page 48*. HTTP recipients: The message will be sent in a CGI argument called Message. Spaces are allowed. If using more than 255 characters, some or all of the information provided in parameters (see below) will be excluded.
parameters	String	HTTP recipients: Additional CGI arguments. Spaces are not allowed; all text must be URL-encoded.
period	Unsigned integer	TCP recipients, continuous upload: Number of seconds between successive notification messages.
subject	String	SMTP recipients: Email subject. Modifiers can be used, see *Modifiers, on page 48*.

4.14 Send Video Clip Action

The Send Video Clip action sends a video clip to a recipient.

This action can be run as:

- **fixed action** – video is recorded during a pre-event and post-event time
- **unlimited action** – video is recorded during a pre-event time, while the event is running and during a post-event time

Recipient	Recipient ID	Action ID
HTTP	com.axis.recipient.http	com.axis.action.fixed.send_videoclip.http com.axis.action.unlimited.send_videoclip.http
HTTPS	com.axis.recipient.https	com.axis.action.fixed.send_videoclip.https com.axis.action.unlimited.send_videoclip.https
SMTP	com.axis.recipient.smtp	com.axis.action.fixed.send_videoclip.smtp com.axis.action.unlimited.send_videoclip.smtp
FTP	com.axis.recipient.ftp	com.axis.action.fixed.send_videoclip.ftp com.axis.action.unlimited.send_videoclip.ftp
Network share	com.axis.recipient.networkshare	com.axis.action.fixed.send_videoclip.networkshare com.axis.action.unlimited.send_videoclip.networkshare

The following parameters should be specified:

Parameter	Valid values	Description
stream_options	URL-encoded string	List of stream parameters such as resolution, compression etc. All parameters supported by RTSP and HTTP stream requests can be used. These parameters are listed in the Video Streaming API. Example: stream_options=resolution%3D640x480%26compression%3D30%26rotation%3D180
pre_duration	Unsigned integer	Pre-trigger time in milliseconds. Specify the number of milliseconds to include from the time immediately before the event.
post_duration	Unsigned integer	Post-trigger time in milliseconds. Specify the number of milliseconds to include from the time immediately after the event.
create_folder	String	Create a folder on the recipient to store uploaded files in. Modifiers can be used, see *Modifiers, on page 48*.
filename	String	File name for uploaded files. Modifiers can be used, see *Modifiers, on page 48*.
max_file_size	Unsigned integer	Not supported. Use empty string.
max_duration	Unsigned integer	Not supported. Use empty string.
custom_params	String	HTTP and HTTPS recipients: Additional CGI arguments.
subject	String	SMTP recipients: Email subject. Modifiers can be used, see *Modifiers, on page 48*.
message	String	SMTP recipients: Email message. Modifiers can be used, see *Modifiers, on page 48*.

5 Recipient Templates

This section describes available recipient templates. Each template lists the parameters needed to configure the recipient. All parameters must be specified; if a parameter should not be used, its value should be set to an empty string.

Note

Supported templates are product and firmware dependent. Use `GetRecipientTemplates` to list the recipient templates supported by the Axis product.

5.1 FTP Recipient

An FTP recipient can receive video clips, uploaded images and notification messages.

Recipient ID: `com.axis.recipient.ftp`

Parameter	Valid values	Description
`host`	IP address or host name	Required. The IP address or host name to the FTP server.
`port`	Unsigned integer	FTP port. Default: `21`
`login`	String	User name for the FTP server
`password`	String	Password for the FTP server user
`passive`	`1` `0`	Use passive mode. If `passive=1`the Axis product establishes a passive FTP connection to the FTP server using the `PASV` command. A passive connection is often required if there is a firewall between the product and the FTP server.
`temporary`		Not supported.
`qos`	`0 ... 63`	Quality of Service. The DSCP (Differentiated Services Codepoint) value for network traffic from the FTP recipient. Default: `0`

5.2 HTTP Recipient

An HTTP recipient can receive video clips, uploaded images and notification messages.

Recipient ID: `com.axis.recipient.http`

Parameter	Valid values	Description
`upload_url`	String	Required. A URL containing the network address to the HTTP server and the script that will handle the request. For example: `http://ip_address/cgi-bin/notify.cgi`
`login`	String	User name for the HTTP server
`password`	String	Password for the HTTP server user
`proxy_host`	IP address or host name	Proxy address
`proxy_port`	Unsigned integer	Proxy port
`proxy_login`	String	User name for the proxy

proxy_password	String	Password for the proxy user
qos	0 ... 63	Quality of Service. The DSCP (Differentiated Services Codepoint) value for network traffic from the HTTP recipient. Default: 0

5.3 HTTPS Recipient

An HTTPS recipient can receive video clips, uploaded images and notification messages.

Recipient ID: com.axis.recipient.https

Parameter	Valid values	Description
upload_url	String	Required. A URL containing the network address to the HTTPS server and the script that will handle the request. For example: https://ip_address/cgi-bin/notify.cgi
login	String	User name for the HTTPS server
password	String	Password for the HTTPS server user
proxy_host	IP address or host name	Proxy address
proxy_port	Unsigned integer	Proxy port
proxy_login	String	User name for the proxy
proxy_password	String	Password for the proxy user
qos	0 ... 63	Quality of Service. The DSCP (Differentiated Services Codepoint) value for network traffic from the HTTPs recipient. Default: 0
validate_server_cert	0 1	1 = validate the HTTPS server certificate against certificates installed in the product. 0 = Do not validate the certificate.

5.4 Network Share Recipient

A Network Share recipient can receive video clips and uploaded images.

A network share can be a share on a NAS (Network Attached Storage) or any server that uses CIFS (Common Internet File System) also known as SMB (Server Message Block).

Using a network share recipient allows you to store recordings with user defined directory, structure and file names. The files are stored as mkv files.

Recipient ID: com.axis.recipient.https

Parameter	Valid values	Description
upload_path	String	The path to the directory on the network share where uploaded images or recordings should be stored.
share_id	Share ID	The Share ID. The Share ID can be retrieved from axis-cgi/disks/networkshare/list.cgi?shareid=all
qos	0 ... 63	Quality of Service. The DSCP (Differentiated Services Codepoint) value for network traffic from the Network Share recipient. Default: 0

5.5 SMTP Recipient

An SMTP recipient can receive notification messages. Snapshots can be attached as email attachments.

Recipient ID: `com.axis.recipient.smtp`

Parameter	Valid values	Description
`email_to`	Email address	The email address to send messages to
`email_from`	Email address	Email address of the sender
`host`	IP address or host name	IP address or host name for the SMTP server
`port`	Unsigned integer	SMTP server port
`login`	String	User name for the SMTP server
`password`	String	Password for the SMTP server user
`pop_host`	IP address or host name	IP address or host name for the POP3 server used for authentication
`pop_port`	Unsigned integer	POP3 server port
`pop_login`	String	User name for the POP server
`pop_password`	String	Password for the POP server user
`qos`	`0 ... 63`	Quality of Service. The DSCP (Differentiated Services Codepoint) value for network traffic from the SMTP recipient. Default: `0`

`encryption`	`none`
	`ssl`
	`tls`
		The encryption method to use.

`validate_server_cert`	`0` `1`	`1` = validate the SMTP server certificate against certificates installed in the product. `0` = Do not validate the certificate.

5.6 TCP Recipient

A TCP recipient can receive notification messages.

Recipient ID: `com.axis.recipient.tcp`

Parameter	Valid values	Description
`host`	IP address or host name	Required. The IP address or host name of the TCP server.
`port`	Unsigned integer	TCP port
`qos`	`0 ... 63`	Quality of Service. The DSCP (Differentiated Services Codepoint) value for network traffic from the TCP recipient. Default: `0`

A Modifiers

Modifiers can be used to format file names, folders for uploaded images, notification messages, text in image overlays and similar. A modifier always starts with a % or # character, followed by another character.

The percent ("%") and hashtag ("#") must be URL encoded. That means:

- % = `%25`
- # = `%23`

Example 1:
Use `image-%F-%H-%M-%S.jpg` so timestamp uploaded video snapshots with the date, hour, minute and second the snapshot was taken.

The following modifiers are available:

Date		
Modifier	**Description**	**Example**
`%c`	Date and time.	Sun Dec 25 10:25:01 2011
`%D`	Date in format MM/DD/YY.	25/12/2011
`%F`	Date in format YYYY-MM-DD.	2011–12–25
`%x`	Same as `%D`.	

Time		
Modifier	**Description**	**Example**
`%p`	AM or PM according to the given time or the corresponding strings for the current locale. Noon is treated as PM and midnight as AM.	
`%r`	Time in a.m or p.m. notation.	10:25:01 AM
`%R`	Time in 24-hour notation without seconds.	10:25
`%T`	Time in 24-hour notation with seconds.	10:25:01
`%X`	Same as `%T`.	
`%z`	Time zone as offset from UTC.	+0000
`%Z`	Time zone name or abbreviation.	GMT

Year		
Modifier	**Description**	**Example**
`%C`	The century as a 2-digit number (year/100).	20
`%G`	The ISO 8601 week-numbering year as a 4-digit number.	2011
`%g`	The ISO 8601 week-numbering year as a 2-digit number without the century, range 00 to 99.	11
`%Y`	The Gregorian calendar year as a 4-digit number.	2011
`%y`	The Gregorian calendar year as a 2-digit number without the century, range 00 to 99.	11

Month		
Modifier	**Description**	**Example**
%b	Abbreviated month name.	Dec
%B	Full month name.	December
%h	Same as %b.	
%m	Month as a 2-digit number, range 01 to 12.	12

Week		
Modifier	**Description**	**Example**
%U	The week number as a 2-digit number, range 00 to 53. Sunday is the first day of the week. Week 01 is the week starting with the first Sunday of the current year.	52
%V	The ISO 8601 week number as a 2-digit number, range 01 to 53. Monday is the first day of the week. Week 01 is the first week that has at least four days in the current year.	51
%W	The week number as a 2-digit number, range 00 to 53. Monday is the first day of the week. Week 01 is the week starting with the first Monday of the current year.	51

Day		
Modifier	**Description**	**Example**
%a	Abbreviated weekday name.	Sun
%A	Full weekday name.	Sunday
%d	Day of the month as a 2-digit number, range 01 to 31.	25
%e	Same as %d but a leading zero is replaced by a blank space.	25
%j	Day of the year as a 3-digit number, range 001 to 366.	359
%u	Day of the week as an 1-digit number, range 1 to 7. Monday is 1.	7
%w	Day of the week as an 1-digit number, range 0 to 6. Sunday is 0.	0

Hour		
Modifier	**Description**	**Example**
%H	Hour in 24-hour fomat, range 00 to 23.	10
%I	Hour in 12-hour format, range 01 to 12.	10
%k	Same as %H but a leading zero is replaced by a blank space.	
%l	Same as %I but a leading zero is replaced by a blank space.	

Minute		
Modifier	**Description**	**Example**
%M	Minute as a 2-digit number, range 00 to 59.	25

Second		
Modifier	**Description**	**Example**
%f	1/100 seconds as a 2-digit number.	67
%s	The number of seconds since EPOCH, that is, since 1970–01–01 00:00:00 UTC.	1319178723
%S	The current second as a 2-digit number, range 00 to 59.	10

Product and system information		
Modifier	**Description**	**Example**
#I	The IP address.	10.13.24.88
#m	The short MAC address (last 6 characters).	77:F8:26
#M	The full MAC address (all characters).	00:40:8C:77:F8:26
#n	The host name.	axis-00408c77f826
#U<*index*>	The fan status. <*index*> = A specified fan. For example 1.	Stopped
#TC<*index*>	The temperature of the camera sensor in Celsius. <*index*> = A specified camera sensor. For example 1.	48,4
#TF<*index*>	The temperature of the camera sensor in Fahrenheit. *index*> = A specified camera sensor. For example 1.	119,1

Video information		
Modifier	**Description**	**Example**
#b	Bit rate in kbit/s (no decimals).	16333
#B	Bit rate in Mbit/s (two decimals).	16.33
#r	Frame rate with two decimals.	30.00
#R	Frame rate without decimals.	30
#v	Video source number.	1

Pan/Tilt/Zoom information		
Modifier	**Description**	**Example**
#x	Pan coordinate (signed, with two decimals).	–77.61
#y	Tilt coordinate (signed, with two decimals).	–7.61
#z	Zoom coordinate (range 1 to 19999).	256
#Z	Zoom magnification (with one decimal).	12.0
#p	Preset position number. If not at a preset position, blank space is used.	3

#P	Preset name. If not at a preset position, blank space is used.	Door
#L	OSDI (On Screen Directional Indicator) zone name. A zone within specified pan and tilt coordinates. If not within an OSDI zone, blank space is used.	Construction

Others		
Modifier	**Description**	**Example**
#s	Sequence number of the video image as a 5-digit number.	
#D	Dynamic text overlay.	
%%	The % character.	
##	The # character.	

B Helper Functions

This section defines the helper functions used in this document.

- `CreateActionServiceClient` and `CreateEventServiceClient`, see *Create Web Service Connections*.
- Functions for data formatting, see *Formatting Data*.
- Error handling functions, see *Error Handling*.

B.2 Create Web Service Connections

`CreateActionServiceClient` creates an action service client.

```
/// <summary>
/// Creates an action service client with the specified address and the provided
credentials.
/// </summary>
private static ActionClient CreateActionServiceClient(
 string address, string userName, string password)
{
 // Create SOAP 1.2 text encoding binding element.
 TextMessageEncodingBindingElement soap12Encoding =
  new TextMessageEncodingBindingElement(MessageVersion.Soap12, Encoding.UTF8);

 // Create HTTP transport binding element with basic authentication.
 HttpTransportBindingElement httpTransport = new HttpTransportBindingElement();
 httpTransport.AuthenticationScheme = AuthenticationSchemes.Basic;

 Binding binding = new CustomBinding(soap12Encoding, httpTransport);
 EndpointAddress endpoint = new EndpointAddress(
  string.Format("http://{0}/vapix/services", address));

 // Create the client and set credentials.
 ActionClient client = new ActionClient(binding, endpoint);
 client.ClientCredentials.UserName.UserName = userName;
 client.ClientCredentials.UserName.Password = password;

 return client;
}
```

`CreateEventServiceClient` creates an event service client.

```
/// <summary>
/// Creates an event service client with the specified address and the provided
credentials.
/// </summary>
private static EventClient CreateEventServiceClient(
 string address, string userName, string password)
{
 // Create SOAP 1.2 text encoding binding element.
 TextMessageEncodingBindingElement soap12Encoding =
  new TextMessageEncodingBindingElement(MessageVersion.Soap12, Encoding.UTF8);

 // Create HTTP transport binding element with basic authentication.
 HttpTransportBindingElement httpTransport = new HttpTransportBindingElement();
 httpTransport.AuthenticationScheme = AuthenticationSchemes.Basic;

 Binding binding = new CustomBinding(soap12Encoding, httpTransport);
```

```
 EndpointAddress endpoint = new EndpointAddress(
  string.Format("http://{0}/vapix/services", address));

 // Create the client and set credentials.
 EventClient client = new EventClient(binding, endpoint);
 client.ClientCredentials.UserName.UserName = userName;
 client.ClientCredentials.UserName.Password = password;

 return client;
}
```

B.3 Formatting Data

The following functions are used to format XML data.

```
/// <summary>
/// Document header for XML documents.
/// </summary>
private const string DocumentHeader = @"<?xml version=""1.0"" encoding=""UTF-8""?>";
```

```
/// <summary>
/// XML for a topic filter.
/// </summary>
private const string TopicXml =
 @"<wsnt:TopicExpression " +
  @"xmlns:tns1=""http://www.onvif.org/ver10/topics"" " +
  @"xmlns:tnsaxis=""http://www.axis.com/2009/event/topics"" " +
  @"xmlns:wsnt=""http://docs.oasis-open.org/wsn/b-2"" " +
  @"Dialect=""http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"">" +
  @"{0}" +
 @"</wsnt:TopicExpression>";
```

```
/// <summary>
/// XML for a message filter.
/// </summary>
private const string MessageXml =
 @"<wsnt:MessageContent " +
  @"xmlns:wsnt=""http://docs.oasis-open.org/wsn/b-2"" " +
  @"Dialect=""http://www.onvif.org/ver10/tev/messageContentFilter/ItemFilter"">" +
  @"{0}" +
 @"</wsnt:MessageContent>";
```

```
/// <summary>
/// Formats a topic expression from the provided topic and message filters.
/// </summary>
private static FilterType FormatTopicExpression(string topicFilter, string
messageFilter = null)
{
 // Create a list with the topic filter formatted as XML.
 List<XmlElement> xmlElements = new List<XmlElement>
 {
  ToXml(TopicXml, topicFilter)
 };

 // If there is a message filter add it to the list.
 if (!string.IsNullOrWhiteSpace(messageFilter))
 {
  xmlElements.Add(ToXml(MessageXml, messageFilter));
```

```
  }

  // Return the formatted topic expression.
  return new FilterType { Any = xmlElements.ToArray() };
}
```

```
/// <summary>
/// Converts a string to an <see cref="XmlElement"/>.
/// </summary>
private static XmlElement ToXml(string xml, string data)
{
  string document = DocumentHeader + string.Format(xml, data);

  XmlDocument doc = new XmlDocument();
  doc.LoadXml(document);

  return doc.DocumentElement;
}
```

B.4 Error Handling

The following functions are used for error handling.

```
/// <summary>
/// Method for handling exceptions that can occur during web service calls.
/// </summary>
/// <param name="exception">The exception.</param>
private void HandleException(Exception exception)
{
  if (exception is MessageSecurityException)
  {
    // Invalid credentials.
  }
  if (exception is FaultException)
  {
    // Check the documentation to see a list of possible faults.
  }
  if (exception is ProtocolException)
  {
    // .NET fails to catch some FaultExceptions so look at the ProtocolException to be
sure
    // that it isn't a hidded FaultException.
    Exception e = ToFaultException(exception as ProtocolException);

    if (e is FaultException)
    {
      // Check the documentation to see a list of possible faults.
    }
    else
    {
      // It is a real ProtocolException.
      // These can be thrown if the client and server uses different versions of the
WSDL.
    }
  }

  if (exception is CommunicationException ||
    exception is TimeoutException)
  {
    // Failed to contact camera.
```

```
  }

  // No recognized exception
  throw exception;
}
```

```
/// <summary>
/// Transforms the exception to a FaultException if possible.
/// </summary>
/// <param name="e">The exception to transform.</param>
/// <returns>A FaultException or e if transformation failed.</returns>
private static Exception ToFaultException(CommunicationException e)
{
 WebException webException = e.InnerException as WebException;
 if (webException != null && webException.Status == WebExceptionStatus.ProtocolError)
 {
  try
  {
   using (HttpWebResponse response = (HttpWebResponse)webException.Response)
   using (Stream responseStream = response.GetResponseStream())
   {
    if (responseStream != null && responseStream.CanRead)
    {
     using (StreamReader textStream = new StreamReader(responseStream,
Encoding.UTF8))
     {
      string soapErrorXml = textStream.ReadToEnd();
      return ParseFaultException(soapErrorXml) ??  e;
     }
    }
   }
  catch (IOException)
  {
   return e;
  }
 }

 return e;
}
```

```
// Strings used in the XML parsing.
private const string SoapNamespace = "{http://www.w3.org/2003/05/soap-envelope}";
private const string Fault = SoapNamespace + "Fault";
private const string Code = SoapNamespace + "Code";
private const string SubCode = SoapNamespace + "Subcode";
private const string Value = SoapNamespace + "Value";
private const string Reason = SoapNamespace + "Reason";
private const string Text = SoapNamespace + "Text";
```

```
/// <summary>
/// Parses the SOAP error.
/// </summary>
/// <param name="soapErrorXml">The SOAP error XML.</param>
/// <returns>The parsed error or null if parsing failed.</returns>
private static FaultException ParseFaultException(string soapErrorXml)
{
 try
 {
  XDocument xml = XDocument.Parse(soapErrorXml);
  var dec = xml.Descendants(Fault);
  var fault =
   (from faultItem in dec
    select new
    {
     Code = GetValueOrDefault(faultItem.Elements(Code).Elements(Value)),
     SubCode = GetValueOrDefault(
     faultItem.Elements(Code).Elements(SubCode).Elements(Value)),
     SubSubCode = GetValueOrDefault(
     faultItem.Elements(Code).Elements(SubCode).Elements(SubCode).Elements(Value)),
     Reason = GetValueOrDefault(faultItem.Elements(Reason).Elements(Text))
    }).First();

  FaultCode code;
  if (!string.IsNullOrWhiteSpace(fault.SubSubCode))
  {
   FaultCode subSubCode = new FaultCode(fault.SubSubCode, string.Empty);
   FaultCode subCode = new FaultCode(fault.SubCode, string.Empty, subSubCode);
   code = new FaultCode(fault.Code, string.Empty, subCode);
  }
  else if (!string.IsNullOrWhiteSpace(fault.SubCode))
  {
   FaultCode subCode = new FaultCode(fault.SubCode, string.Empty);
   code = new FaultCode(fault.Code, string.Empty, subCode);
  }
  else
  {
   code = new FaultCode(fault.Code, string.Empty);
  }

  FaultException error = new FaultException(fault.Reason, code);

  return error;
 }
 catch (XmlException)
 {
  // Parsing failed - return null;
  return null;
 }
}
```

```
/// <summary>
/// Gets the value of the first element in the list or an empty string if list is
empty.
/// </summary>
/// <returns>The value or an empty string if list is empty.</returns>
private static string GetValueOrDefault(IEnumerable<XElement> elements)
{
 var element = elements == null ?  null :  elements.FirstOrDefault();
 return element == null ?  string.Empty :  element.Value;
}
```

VAPIX® Web Services
Event and Action Services
Ver. M5.12
Date: April 2014
Part No. 56628