Status
Technical details

Can you teach how to implement ONVIF protocol

Teaching or walking-through the detailed ONVIF implementation is really **outside the scope** of our free ONVIF verification service.

However, if you want, you can **hire us** to be your consultant to do this, but hiring us is **not free**. For more information, please refer to our official Consultancy page.

In other parts of this document, we have already provided many implementation hints. You should be able to extrapolate the information you need from these hints, but please be aware these hints are not officially supported.

Can you provide detailed XML dialog as an implementation example

See also

- Can you teach how to implement ONVIF protocol

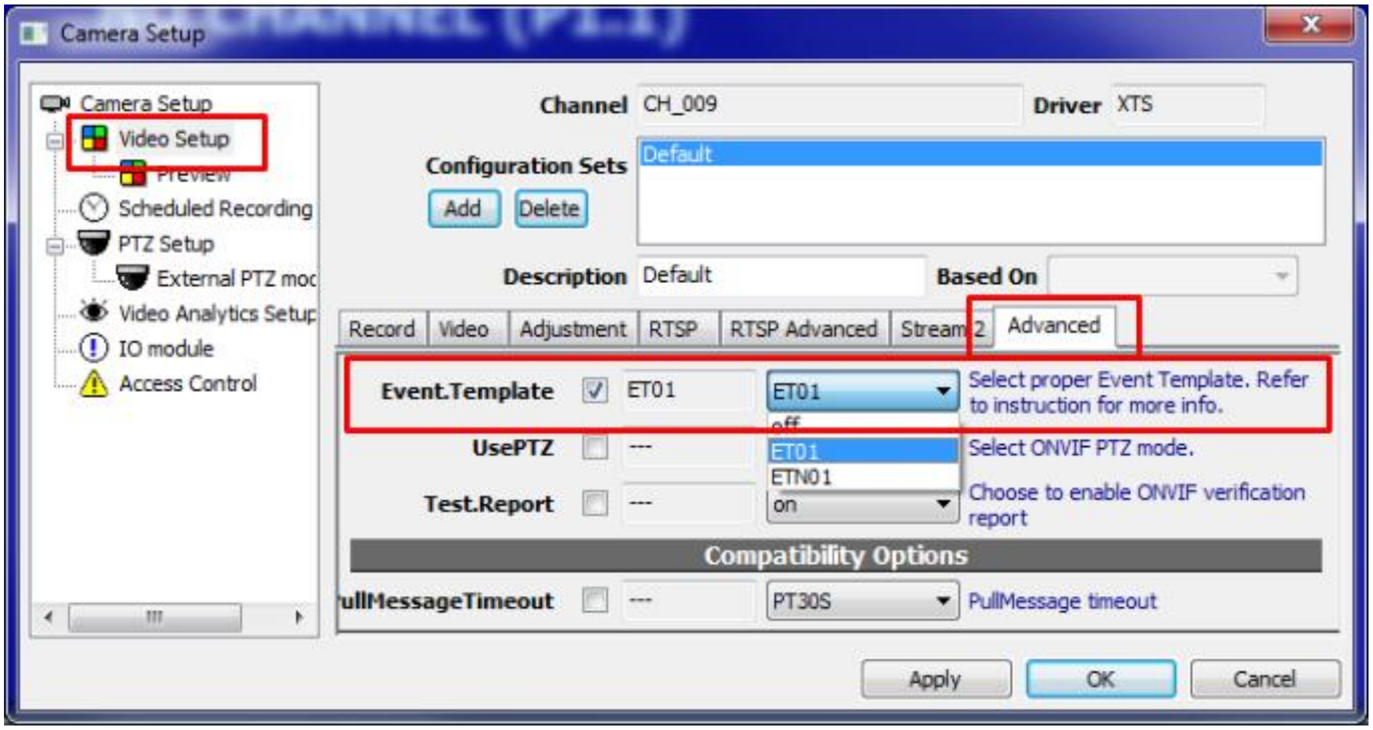
What is Event Template and why is it needed

Event Template is a parameter defined by Genius Vision to remedy the fact that some parts of ONVIF are vague or not clear enough about actual implementation of event notification mechanism. This, in reality, causes the developers to imagine and implement ONVIF in several slightly different ways and thus won't interoperate altogether.

As one of our principals stated, and because no specs are perfect that are without flaws or contraditions, we created this additional parameter in order to workaround such spec issues and provide several alternative implementation to accomodate these compatibility issues.

The summarized definitions of each **Event Template** can be looked-up here.

Following is an example of how to configure **Event Template** for a particular NVR channel.



Channel CH_009 Driver XTS

Configuration Sets Default

Description Default Based On

Event.Template	✓	ET01	ET01 (off / ET01 / ETN01)	Select proper Event Template. Refer to instruction for more info.
UsePTZ		---		Select ONVIF PTZ mode.
Test.Report		---	on	Choose to enable ONVIF verification report
Compatibility Options				
ullMessageTimeout		---	PT30S	PullMessage timeout

What ONVIF specs are referenced?

- ONVIF-Imaging-Service-Spec-v221.pdf
- ONVIF-Core-Specification-v230.pdf
- ONVIF-DeviceIo-Service-Spec-v221.pdf

How should I treat the implementation hints

Terms & conditions

To spare camera developers the efforts from having to look around & search in the dozens of ONVIF specs using our professional knowledge, we have put some implementation note in this document. However, you need to understand the following rules before reading these notes:

1. The implementation hints are **not supported**. The support effort is subject to consultant service which is available only through our distributor and require paying fees.
2. There is no guarantee of *any accuracy* indicated in the implemenation hints. *Use on your own risk*.
3. ONVIF verification should **not** be done by interpreting XML or "test report". Rather, it should be done **manually & visually** by operating the software and the camera and see if they interoperate as expected. Many camera ONVIF deficiencies are known to be caused by programmer's reliance on "test tools". One should not let machine to do what was meant for people's responsibility, especially when a programmer knows how to "cheat" the test-tool. Please read this article for more info.
4. Implementation hints does not substitute ONVIF spec in anyway. We still believe the best way to understand ONVIF is simply to read the specs and *get very familarized* with them.
5. If you need to test a working software, you're welcomed to download our software from our Free ONVIF Verification portal, for *evaluation purpose*.
6. For the actual spec documents referenced, please refer to What ONVIF specs are referenced?

XML namespaces

- Except for additionally described, all XML namespace prefix definitions used in the XML examples are consistent with ONVIF-Core-Specification-v230.pdf, **5.3 Namespaces**.

Legends

- In the XML example, if you see this **green-highlighted text**, it means you must substitute the highlighted text into programmatic or proper value.
- The "<ver20/imaging>**GetOptions**" is a short-hand notation, meaning it refers to the **GetOptions** command of Imaging service 2.0, and it's actually *different* from Imaging service 1.0. This notation is required to clearly express the XML namespace changes among spec versions, especially from 1.0 to 2.0. For more details please refer to this technical article: What does ONVIF version 1.0, 1.1, 1.2, 2.0 generally means? Why version 1.0, 1.1, 1.2, 2.0 can be incompatible?
- The bold black text, depending on context, such as "**GetProfiles**" usually means a command name defined ONVIF spec, where bold green text, such as "**Channel**" usually means an internal NVR variable or configurable parameter used to determine the actual software behavior.

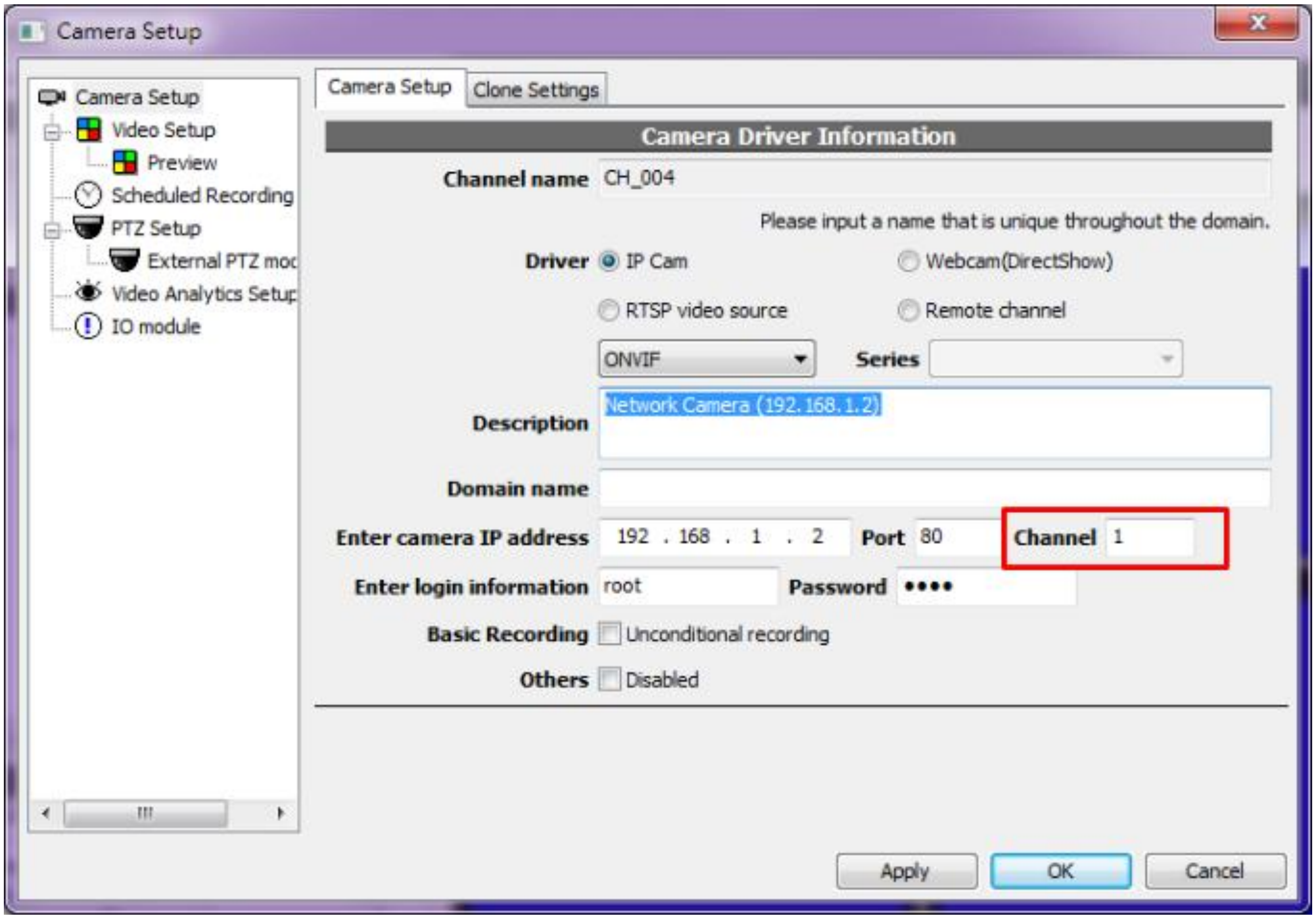
If I'm going to use Genius Vision NVR to verify my camera, what basic knowledge should I have for this software?

Please refer to following link for a short version of Genius Vision NVR manual for ONVIF self-verification.

- Setup Genius Vision NVR for ONVIF Testing

What is the "Channel" parameter of the NVR software

Channel is a NVR channel configuration parameter (as illustrated by the image below), which identifies the video source of a potentially multi-channel device. Normally the value is "**1**" and it means **the first** video source. If you change it to "**2**", it will select **the second** video source, and etc.



Camera Driver Information

Channel name: CH_004

Please input a name that is unique throughout the domain.

Driver: [✓ IP Cam] Webcam(DirectShow) RTSP video source Remote channel

ONVIF Series

Description: Network Camera (192.168.1.2)

Domain name

Enter camera IP address: 192 . 168 . 1 . 2 Port 80 Channel 1

Enter login information: root Password ••••

Basic Recording: Unconditional recording

Others: Disabled

In the terminology of ONVIF, this mostly means that among the video sources returned by **GetVideSources**, only that matches what is specified by **Channel** will be used, while among profiles returned by **GetProfiles,** only the profiles matching the aforementioned video source will be used.

What is the initial camera attaching sequence

Warning! This is an implementation hint article. Read this first.

Calling sequence

1. Calls <device>**GetCapabilities** to obtain device capabilities and various endpoints
2. Calls **GetVideoSources** to obtain a list of video sources. For one channel in Genius Vision NVR, only one video source will be used and it's determined by the **Channel** parameter (explained here) in NVR. **Channel** of value "**1**" designates the first video source, etc.
3. Calls **GetProfiles** to obtain a list of media profiles. Each profile token will be listed in the available stream selection of Genius Vision NVR. Please note only those profiles matching the designated video source (as specified by **Channel**) will be used.
4. Calls <ver20/imaging>**GetOptions** to obtain supported imaging options, if this call fails, the NVR will try to call <ver10/imaging>**GetOptions** instead. At this moment, an automatic protocol version detection of imaging service is performed and result is stored to an internal variable called **ImagingVersion**. The result of available imaging option will be listed on NVR.
5. Compare NVR imaging settings against the camera settings (using the result of **GetProfiles**), the NVR will determine a list of imaging parameters that needs to be changed, and then calls

   <ver20/imaging>**SetImagingSettings**, or <ver10/imaging>**SetImagingSettings**, depend on **ImagingVersion**.
6. Calls **GetVideoEncoderConfigurationOptions** for each stream configured. Available options will be listed on the NVR.
7. Compare NVR video encoder settings against the camera settings (using the result of **GetProfiles**). NVR will then determine a list of video encoder settings that needs to be changed, and then calls **SetVideoEncoderConfiguration**.
8. Calls **GetStreamUri** to obtain media URI, in order to get audio/video streaming, then use the designated parameter to invoke RTSP.

Mandatory commands summary

From the above we can understand that in order to properly support Genius Vision NVR, following ONVIF commands need to be implemented, but not all of them are mandatory. The mandatory commands are:

- <device>**GetCapabilities** - mandatory
- **GetVideoSources** - mandatory
- **GetProfiles** - mandatory
- **GetStreamUri** - mandatory

Optional commands summary

Lacking support of some commands will causes disabling of some NVR functions, but it can still work as a ordinary video camera:

- <ver20/imaging>**GetOptions** or <ver10/imaging>**GetOptions** - optional. Lacking support for this command causes NVR to disable all ImagingOptions functions.
- <ver20/imaging>**SetImagingSettings** or <ver10/imaging>**SetImagingSettings** - optional. Lacking support for this command causes NVR to disable all ImagingOptions functions.
- **GetVideoEncoderConfigurationOptions** - optional. Lacking support for this command causes NVR to disable all VideoEncoderOptions functions.
- **SetVideoEncoderConfiguration** - optional. Lacking support for this command causes NVR to disable all VideoEncoderOptions functions.

How to implement dual-streaming

Warning! This is an implementation hint article. Read this first.

To understand this part, please read the following hint first:

- What is the initial camera attaching sequence

For ONVIF dual-streaming, Genius Vision NVR will call **GetProfiles** to obtain a list of supported media profiles. Each profile token will be listed in the stream selection of Genius Vision NVR. To support dual-streaming, at least two profiles must be supported. Please note only those profiles matching the designated video source (as specified by **Channel,** explained here) will be used.

How to implement configuring of VideoEncoderOptions (Resolution, Codec, FrameRate etc)

Warning! This is an implementation hint article. Read this first.

To understand this part, please read the following hint first:

- What is the initial camera attaching sequence

To support configuring of **VideoEncoderOptions**, following commands must be supported by the camera:

- **GetProfiles**
- **GetVideoEncoderConfigurationOptions**
- **SetVideoEncoderConfiguration**

How to implement configuring of ImagingOptions (Brightness, Saturation, etc)

Warning! This is an implementation hint article. Read this first.

To understand this part, please read the following hint first:

- What is the initial camera attaching sequence

To support configuring of **ImagingOptions**, following commands must be supported by the camera:

- **GetProfiles**. The existence of each value under the returned by the matching **Profile**/**VideoSource**/**Imaging** (including **BacklightCompensation**, **Brightness**, **ColorSaturation**, **Contrast**, **Exposure**, **Focus**, **IrCutFilter**, **Sharpness**, **WideDynamicRange**, **WhiteBalance**, etc) determines if that value is can be used in NVR client.
- <ver20/imaging>**GetOptions** or <ver10/imaging>**GetOptions**
- <ver20/imaging>**SetImagingSettings**, or <ver10/imaging>**SetImagingSettings**

How to implement PTZ

Warning! This is an implementation hint article. Read this first.

Calling sequence

1. Check the NVR configuration parameter **UsePTZ** to see if PTZ is enabled. If not enabled, entire PTZ function initialization is skipped. The setting of **UsePTZ** will also be used to determine if version 1.0 or verision 2.0 of ONVIF PTZ will be used to communicate with the camera.
2. Calls <ver20/ptz>**GetPresets** or <ver10/ptz>**GetPresets** (depend on the value of **UsePTZ**) to get a list of PTZ presets.
3. If mouse PTZ operation is commenced, calls <ver20/ptz>**ContinuousMove** or <ver10/ptz>**ContinuousMove** (depend on the value of **UsePTZ**) to engage PTZ moving operation.
4. If preset-goto operation is commenced, calls <ver20/ptz>**GotoPreset** or <ver10/ptz>**GotoPreset** (depend on the value of **UsePTZ**) to engage PTZ preset-goto operation.

How to support event notification under ONVIF spec

There are generally two ways to support event notification, as defined in ONVIF spec

1. Basic notification
2. Pull-point style notification

Due to the reasons stated here, Genius Vision NVR does not support basic notification. So you should use pull-point style notification only, in order to pass our verification.

P.S. According to the spec, there is actually a third way to support event notification using RTP, but since we haven't seen any vendor has supported it, we won't be able to support it.

See also

- How to implement ONVIF Pull-point style event notification (ET01)

Does Genius Vision NVR supports basic notification

No.

Basic notification, as defined in ONVIF spec, only applies IP camera and NVR in the same network segment. The notification will not be able to be transmitted if the camera is on the Internet and NVR is inside an NAT. This call-back style requirement also potentially requires the user to turn off any firewall mechanism that might exist, even in production stage.

The limitation & requirements introduced by basic notification will likely to cause extreme confusion when problems arise in the field, and it's very difficult to sort out the cause of problem in short time or in an obvious way. In practical sense, using basic notification will *significantly increase the support cost* to all roles of IP video deployment, including system integrators, VMS software developers, and IP camera manufacturers. In contrast, pull-point style notification does not have these problems.

Therefore, we believe that using real-time pull-point style notification is the most appropriate & easy way for deployment of ONVIF event notification, which spares the users the effort from (1) having more IT/network knowledge (2) knowing the exact IP locations of camera and the NVR, in order to use event notification.

Furthermore, requiring users to turning firewall off even in production environment is *controversy*, we will not recommend users to do that.

For all the reasons stated above, we don't plan to add basic notification to Genius Vision NVR.

How to implement ONVIF Pull-point style event notification (ET01)

Warning! This is an implementation hint article. Read this first.

Calling sequence

1. This sequence only applies when **Event.Template** (explained here) is set to **ET01**.
2. Calls **CreatePullPointSubscription**
3. Repeatedly calls to **PullMessage** to get notification message. Notification schema is consolidated at here. (Please pay attention to **ET01**, which is gathered from the spec)

How to implement MotionAlarm (motion detection) event (ET01)

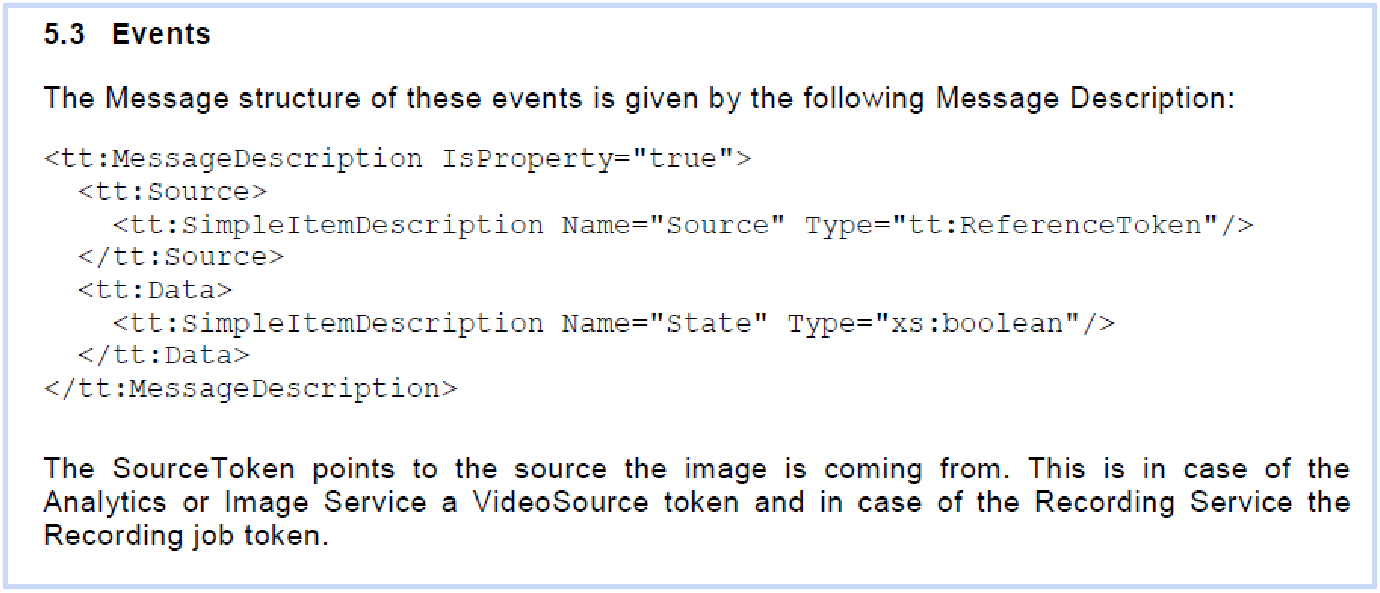
Warning! This is an implementation hint article. Read this first.

Calling sequence

1. This sequence only applies when **Event.Template** (explained here) is set to **ET01**.
2. Engage event notification sequence.

Event definition

ONVIF-Imaging-Service-Spec-v221.pdf

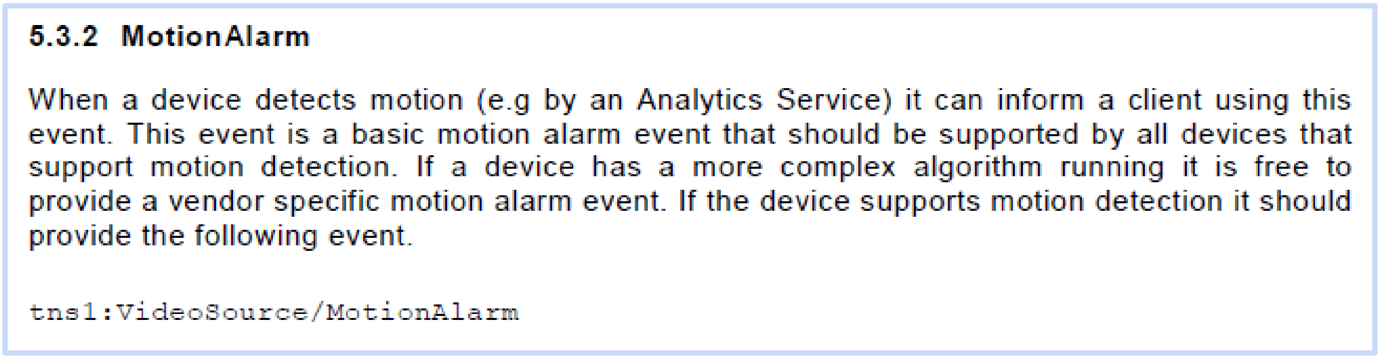


5.3 Events

The Message structure of these events is given by the following Message Description:

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="Source" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

The SourceToken points to the source the image is coming from. This is in case of the Analytics or Image Service a VideoSource token and in case of the Recording Service the Recording job token.



5.3.2 MotionAlarm

When a device detects motion (e.g by an Analytics Service) it can inform a client using this event. This event is a basic motion alarm event that should be supported by all devices that support motion detection. If a device has a more complex algorithm running it is free to provide a vendor specific motion alarm event. If the device supports motion detection it should provide the following event.

```
tns1:VideoSource/MotionAlarm
```

Message XML example

```
      <wsnt:NotificationMessage>
        <wsnt:Topic Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet">
tns1:VideoSource/MotionAlarm
</wsnt:Topic>
        <wsnt:Message>
          <tt:Message UtcTime="2008-10-10T12:24:57.321Z">
            <tt:Source>
              <tt:SimpleItem Name="Source" Value="[videoSourceToken]" />
           </tt:Source>
            <tt:Data>
              <tt:SimpleItem Name="State" Value="[motionState]" />
            </tt:Data>
          </tt:Message>
        </wsnt:Message>
      </wsnt:NotificationMessage>
```

- **[videoSourceToken]**: Must match one of the tokens returned by **GetVideoSources**
- **[motionState]**: **true** if motion is in progress, **false** if not.

See also

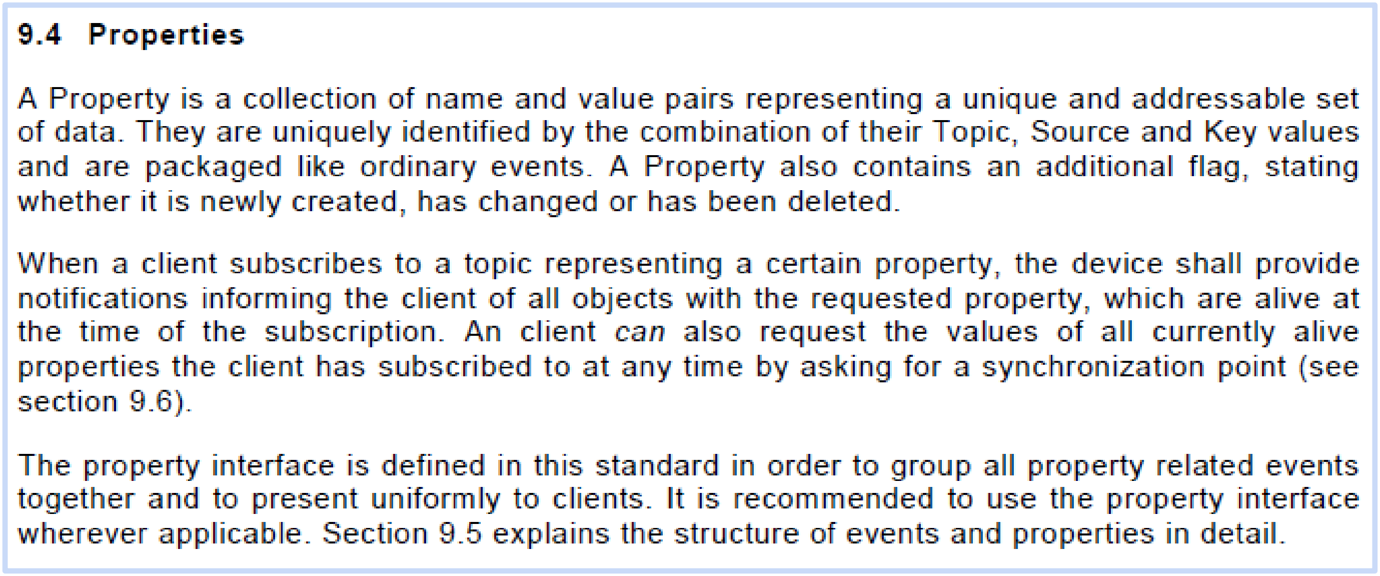
- How to verify ONVIF motion detection

What is Property in the context of ONVIF event? What is SetSynchronizationPoint? How should this be implemented?

Warning! This is an implementation hint article. Read this first.

Introduction of Property in spec

Property, as described in ONVIF-Core-Specification-v230.pdf, is a simplified way to represent *status change* by ONVIF event notification mechanism:



9.4 Properties

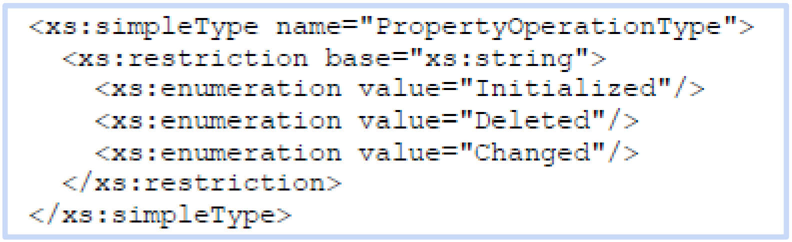
A Property is a collection of name and value pairs representing a unique and addressable set of data. They are uniquely identified by the combination of their Topic, Source and Key values and are packaged like ordinary events. A Property also contains an additional flag, stating whether it is newly created, has changed or has been deleted.

When a client subscribes to a topic representing a certain property, the device shall provide notifications informing the client of all objects with the requested property, which are alive at the time of the subscription. An client *can* also request the values of all currently alive properties the client has subscribed to at any time by asking for a synchronization point (see section 9.6).

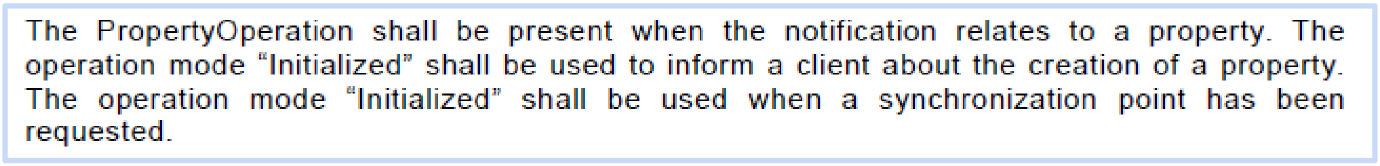
The property interface is defined in this standard in order to group all property related events together and to present uniformly to clients. It is recommended to use the property interface wherever applicable. Section 9.5 explains the structure of events and properties in detail.

Property operations

According to the spec, there are three kinds of property operation, **Initialized**, **Deleted**, and **Changed**:



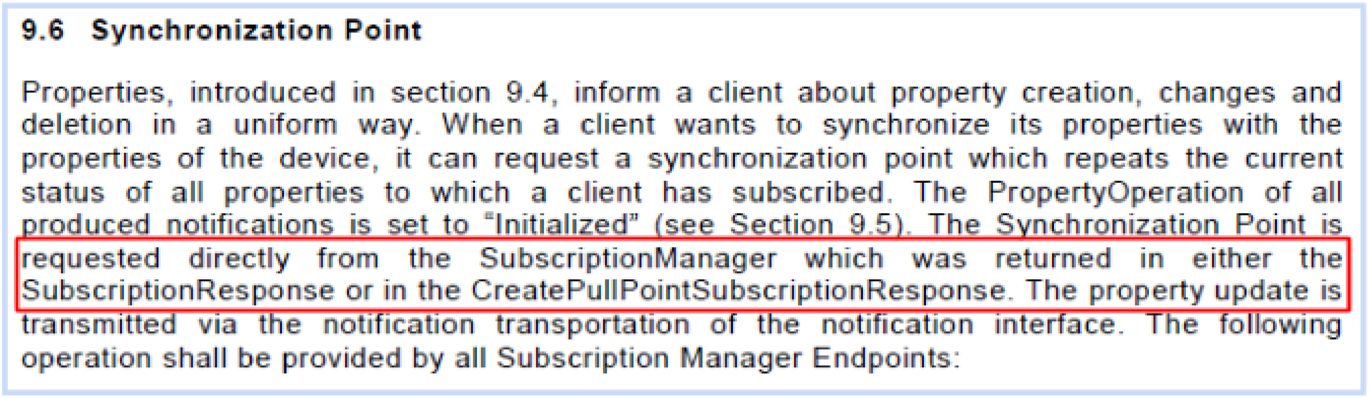
```
<xs:simpleType name="PropertyOperationType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Initialized"/>
    <xs:enumeration value="Deleted"/>
    <xs:enumeration value="Changed"/>
  </xs:restriction>
</xs:simpleType>
```



The PropertyOperation shall be present when the notification relates to a property. The operation mode "Initialized" shall be used to inform a client about the creation of a property. The operation mode "Initialized" shall be used when a synchronization point has been requested.

What is a Synchronization Point

According to spec,


9.6 Synchronization Point

Properties, introduced in section 9.4, inform a client about property creation, changes and deletion in a uniform way. When a client wants to synchronize its properties with the properties of the device, it can request a synchronization point which repeats the current status of all properties to which a client has subscribed. The PropertyOperation of all produced notifications is set to "Initialized" (see Section 9.5). The Synchronization Point is requested directly from the SubscriptionManager which was returned in either the SubscriptionResponse or in the CreatePullPointSubscriptionResponse. The property update is transmitted via the notification transportation of the notification interface. The following operation shall be provided by all Subscription Manager Endpoints:

Notice the red highlighted text, a Synchronzation Point is requested automatically when **CreatePullPointSubscription** is requested. What it really means is that the NVC (client) does not need to issue **SetSynchronizationPoint** explicitly.

Does the NVC (client) need to send SetSynchronizationPoint explicitly

No.

Please refer to previous section. This is clearly defined in spec *without ambiguity*.

What does it really mean by Synchronization Point, I still don't understand

The ONVIF specs tend to get very technical about some pretty simple concepts. What you really need to do is that after receiving **CreatePullPointSubscription** request, NVT should treat the client doesn't know anything about the current device status. Therefore NVT should send all (subscribed) properties with their states to the client, with **PropertyOperation="Initialized"** in **NotificationMessage**. This way the client would know the initial state of each subscribed property.

For example, say a digital input is at **ON** state. When an NVT received **CreatePullPointSubscription** request from client, it should immediately send the **ON** state event to client (with **PropertyOperation="Initialized"**), so the client wouldn't wrongfully assume the digital input state is at **OFF** state (because the client is not told the correct information).

If NVC doesn't need to send SetSynchronizationPoint, then what's the point of having such a command

According to spec:

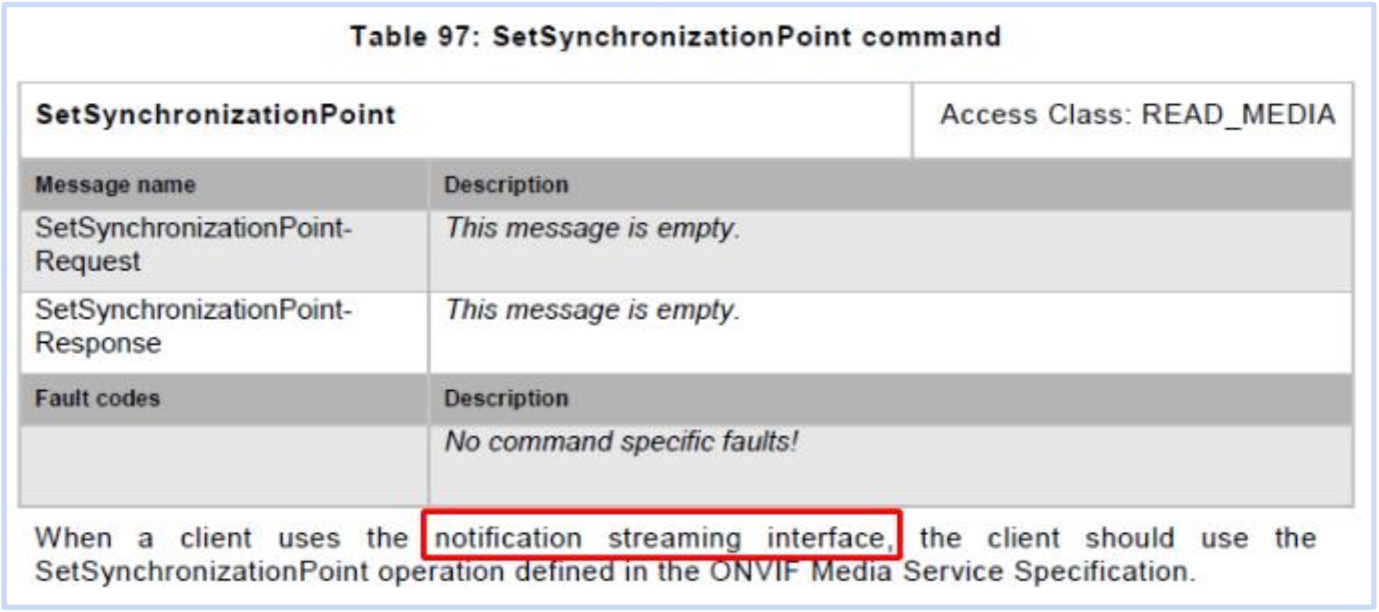


Table 97: SetSynchronizationPoint command

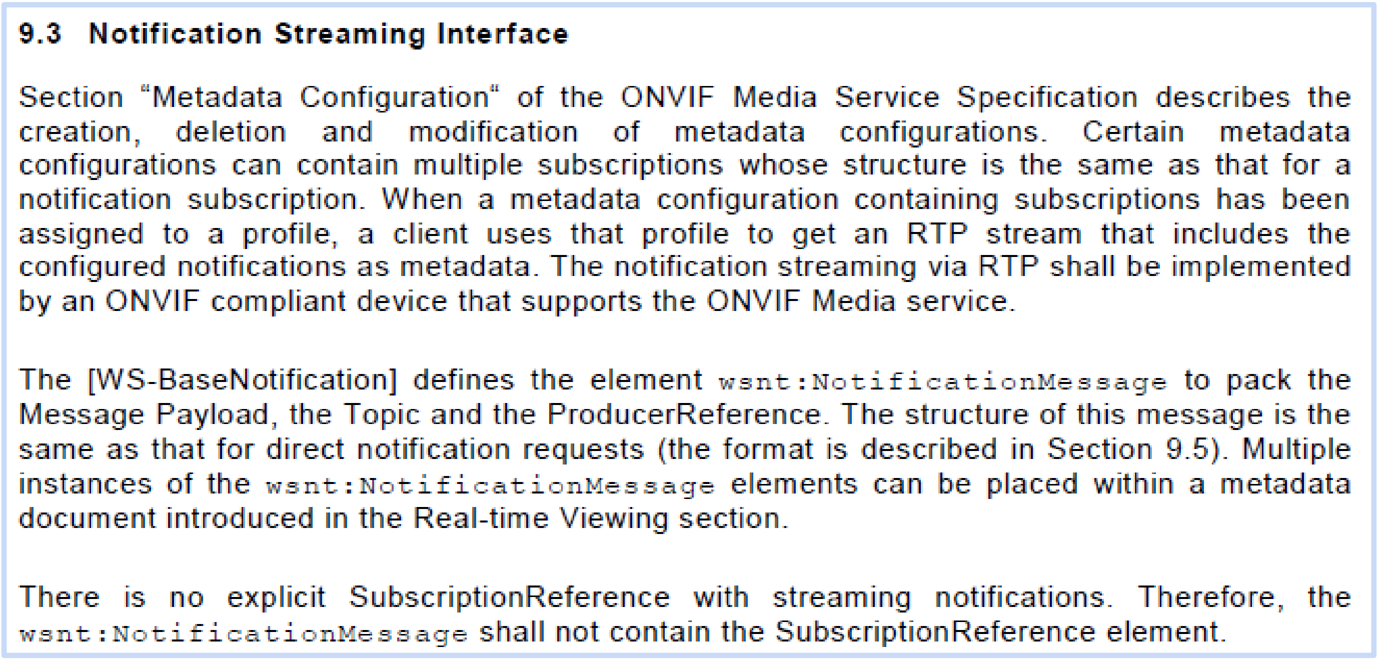
SetSynchronizationPoint	Access Class: READ_MEDIA
Message name	**Description**
SetSynchronizationPoint-Request	*This message is empty.*
SetSynchronizationPoint-Response	*This message is empty.*
Fault codes	**Description**
	No command specific faults!

When a client uses the notification streaming interface, the client should use the SetSynchronizationPoint operation defined in the ONVIF Media Service Specification.

The **SetSynchronizationPoint** command is reserved for **notification streaming interface** (what is this?). Genius Vision NVR uses exclusively Pull-Point style event notification, so this command is never called by Genius Vision NVR.

What is notification streaming interface?

According to spec (ONVIF-Core-Specification-v230.pdf),



9.3 Notification Streaming Interface

Section "Metadata Configuration" of the ONVIF Media Service Specification describes the creation, deletion and modification of metadata configurations. Certain metadata configurations can contain multiple subscriptions whose structure is the same as that for a notification subscription. When a metadata configuration containing subscriptions has been assigned to a profile, a client uses that profile to get an RTP stream that includes the configured notifications as metadata. The notification streaming via RTP shall be implemented by an ONVIF compliant device that supports the ONVIF Media service.

The [WS-BaseNotification] defines the element `wsnt:NotificationMessage` to pack the Message Payload, the Topic and the ProducerReference. The structure of this message is the same as that for direct notification requests (the format is described in Section 9.5). Multiple instances of the `wsnt:NotificationMessage` elements can be placed within a metadata document introduced in the Real-time Viewing section.

There is no explicit SubscriptionReference with streaming notifications. Therefore, the `wsnt:NotificationMessage` shall not contain the SubscriptionReference element.

This special type of notification interface works in conjunction with RTP/RTSP, in which the NVT embeds notification packet inside RTP packets. Genius Vision NVR does not currently support this type of notification interface.

How to implement Digital Input (ET01)

Warning! This is an implementation hint article. Read this first.

Calling sequence

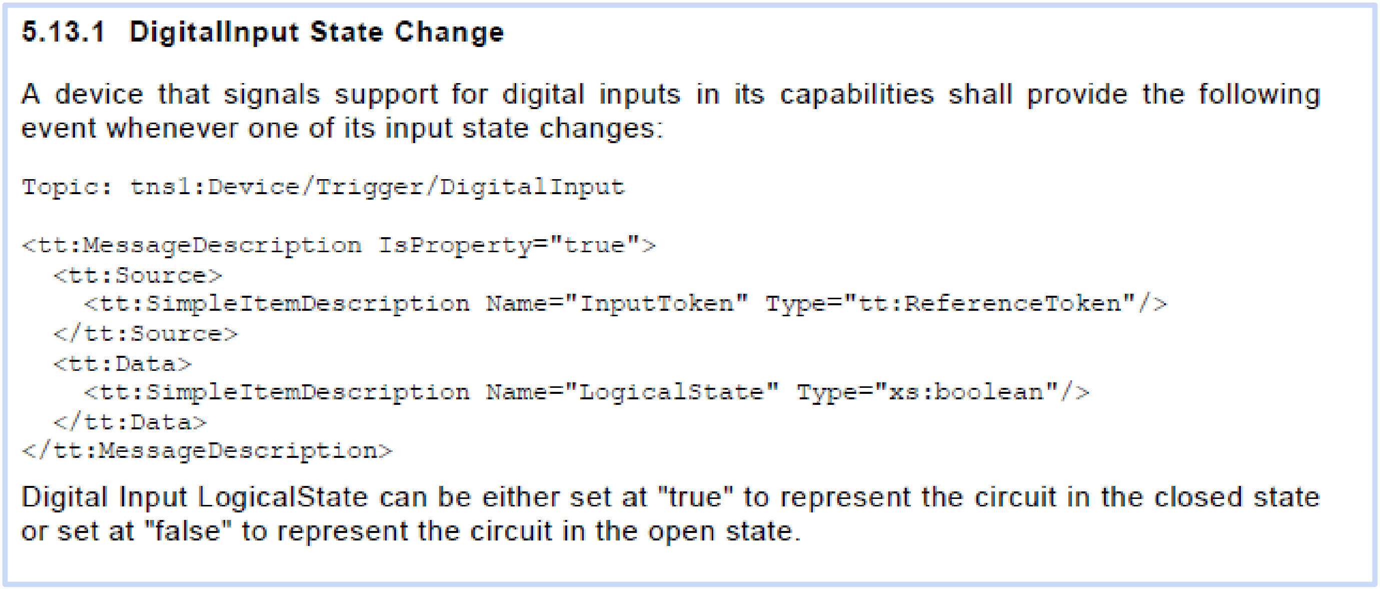
For ONVIF digital input function, the calling sequence of Genius Vision NVR can be summerized as follows:

1. This sequence only applies when **Event.Template** (explained here) is set to **ET01**.
2. Calls <device>**GetDigitalInputs** to obtain number of input and their tokens. If failed, NVR will try to call <ver10/deviceio>**GetDigitalInputs**.
3. Engage event notification sequence.

Reflecting digital input state change

Event definition in spec

ONVIF-DeviceIo-Service-Spec-v221.pdf



5.13.1 DigitalInput State Change

A device that signals support for digital inputs in its capabilities shall provide the following event whenever one of its input state changes:

```
Topic: tns1:Device/Trigger/DigitalInput

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="InputToken" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="LogicalState" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

Digital Input LogicalState can be either set at "true" to represent the circuit in the closed state or set at "false" to represent the circuit in the open state.

Message XML example

```
      <wsnt:NotificationMessage>
        <wsnt:Topic Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet">
tns1:Device/Trigger/DigitalInput
</wsnt:Topic>
        <wsnt:Message>
          <tt:Message UtcTime="2008-10-10T12:24:57.321Z">
            <tt:Source>
              <tt:SimpleItem Name="InputToken" Value="[digitalInputToken]" />
           </tt:Source>
            <tt:Data>
              <tt:SimpleItem Name="LogicalState" Value="[digitalInputState]" />
            </tt:Data>
          </tt:Message>
        </wsnt:Message>
      </wsnt:NotificationMessage>
```

- **[digitalInputToken]**: Must match one of the tokens returned by <device>**GetDigitalInputs** or <ver10/deviceio>**GetDigitalInputs**.
- **[digitalInputState]**: **true** if ON, **false** if OFF.

See also

- How to setup & verify ONVIF DIO (Digital Input/Relay Output)

How to implement Relay Output (ET01)

Warning! This is an implementation hint article. Read this first.

Calling sequence

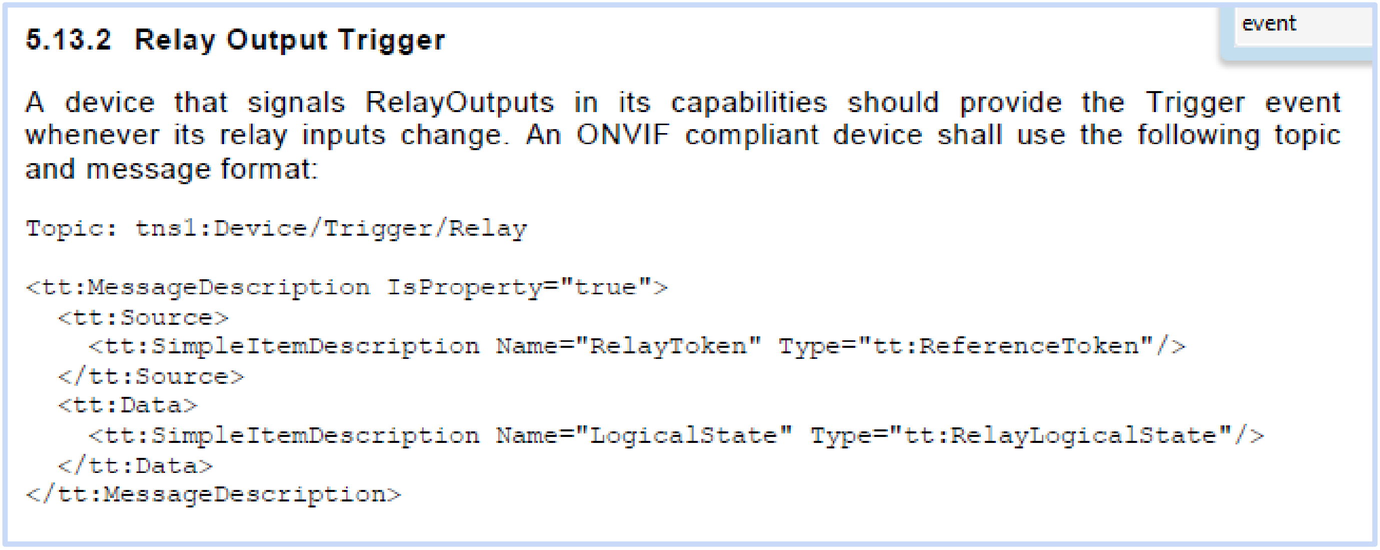
For ONVIF relay output function, the calling sequence of Genius Vision NVR can be summerized as follows:

1. This sequence only applies when **Event.Template** (explained here) is set to **ET01**.
2. Calls <device>**GetRelayOutputs** to obtain number of input and their tokens. If failed, NVR will try to call <ver10/deviceio>**GetRelayOutputs**. This automatic protocol version detection will be remember in an internal variable called **RelayUseDevIO**.
3. Engage event notification sequence.
4. If relay is operated by user to switch ON or OFF, NVR will call <device>**SetRelayOutputState** or <ver10/deviceio>**SetRelayOutputState**, depending on **RelayUseDevIO**.

Reflecting relay state change

Event definition in spec

ONVIF-DeviceIo-Service-Spec-v221.pdf



5.13.2 Relay Output Trigger

A device that signals RelayOutputs in its capabilities should provide the Trigger event whenever its relay inputs change. An ONVIF compliant device shall use the following topic and message format:

```
Topic: tns1:Device/Trigger/Relay

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="RelayToken" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="LogicalState" Type="tt:RelayLogicalState"/>
  </tt:Data>
</tt:MessageDescription>
```

Message XML example

Relay output state is reflected using notification events.

```
      <wsnt:NotificationMessage>
        <wsnt:Topic Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet">
tns1:Device/Trigger/Relay
</wsnt:Topic>
        <wsnt:Message>
          <tt:Message UtcTime="2008-10-10T12:24:57.321Z">
            <tt:Source>
              <tt:SimpleItem Name="RelayToken" Value="[relayToken]" />
            </tt:Source>
            <tt:Data>
              <tt:SimpleItem Name="LogicalState" Value="[relayLogicalState]" />
            </tt:Data>
          </tt:Message>
        </wsnt:Message>
        </wsnt:NotificationMessage>
```

- **[relayToken]**: Must match one of the tokens returned by <device>**GetRelayOutputs** or <ver10/deviceio>**GetRelayOutputs**, depending on **RelayUseDevIO**.
- **[relayLogicalState]**: **active** or **inactive**, reflecting actual relay status.

Changing relay state

- Camera must implement <device>**GetRelayOutputs** or <ver10/deviceio>**GetRelayOutputs** in order to indicate its support to relay ouptuts.
- Camera must implement <device>**SetRelayOutputState** or <ver10/deviceio>**SetRelayOutputState** in order to be able to change relay state.
- The version of spec implemented by the relay output commands must be consistent.

See also

- How to setup & verify ONVIF DIO (Digital Input/Relay Output)

What is the InitialTerminationTime issue

Warning! This is an implementation hint article. Read this first.

Relevant spec hints

The meaning of **InitialTerminationTime** in **CreatePullPointSubscription** command is not very clearly defined in the spec. However it can be roughly interpreted as:

- If **InitialTerminationTime** defined in relative time, the camera should automatically extend the actual termination time by the value specified with **InitialTerminationTime** each time the camera respond to a **PullMessageRequest**.

Status

- Due to the principle stated here, in later versions of NVR software, an new template **ET02** is added to accomodate this particular blurred spec issue.

Technical details

This interpretation can be extrapolated by following spec reading (ONVIF-Core-Specification-v230.pdf, with annotation in colors):

9.12.3 CreatePullPointSubscription

A client can subscribe to specific notifications with the information from the TopicProperties. The following XML example shows the subscription for notifications produced by the Rule Engine of the device. The client reacts only to notifications that reference VideoAnalyticsConfiguration "2" and VideoSourceConfiguration "1". The Subscription has a timeout of one minute. If the subscription is not explicitly renewed or messages are not pulled regularly, it will be terminated automatically after this time.

Pay attention to ambiguous description

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
  xmlns:tet="http://www.onvif.org/ver10/events/wsdl"
  xmlns:tns1="http://www.onvif.org/ver10/topics">
  <SOAP-ENV:Header>
    <wsa:Action>
http://www.onvif.org/ver10/events/wsdl/EventPortType/CreatePullPointSubscriptionReq
uest
    </wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
    <tet:CreatePullPointSubscription>
      <tet:Filter>
        <wsnt:TopicExpression

         Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet">
          tns1:RuleEngine//.
        </wsnt:TopicExpression>
        <wsnt:MessageContent
Dialect="http://www.onvif.org/ver10/tev/messageContentFilter/ItemFilter">
          boolean(//tt:SimpleItem[@Name="VideoAnalyticsConfigurationToken"
                          and @Value="2"] ) and
          boolean(//tt:SimpleItem[@Name="VideoSourceConfigurationToken"
                          and @Value="1"] )
        </wsnt:MessageContent>
      </tet:Filter>
      <tet:InitialTerminationTime>
          PT1M
      </tet:InitialTerminationTime>
    </tet:CreatePullPointSubscription>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

9.12.4 CreatePullPointSubscriptionResponse

When the device accepts the Subscription, it returns the `http://160.10.64.10/Subscription?Idx=0` URI which represents the Endpoint of this Subscription. Additionally, the client is informed about the CurrentTime of the device and the TerminationTime of the created Subscription.

```
<?xml version="1.0" encoding="UTF-8"?>
  <SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
    xmlns:tet="http://www.onvif.org/ver10/events/wsdl">
  <SOAP-ENV:Header>
    <wsa:Action>
http://www.onvif.org/ver10/events/wsdl/EventPortType/CreatePullPointSubscription
Response
    </wsa:Action>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tet:CreatePullPointSubscriptionResponse>
      <tet:SubscriptionReference>
        <wsa:Address>
          http://160.10.64.10/Subscription?Idx=0
        </wsa:Address>
      </tet:SubscriptionReference>
      <wsnt:CurrentTime>
        2008-10-09T13:52:59
      </wsnt:CurrentTime>
      <wsnt:TerminationTime>
        2008-10-09T13:53:59
      </wsnt:TerminationTime>
    </tet:CreatePullPointSubscriptionResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Termination time is computed **initially**.

9.12.5 PullMessagesRequest

The client sends a PullMessagesRequest to the Endpoint given in the CreatePullPointSubscriptionResponse to get Notifications corresponding to a certain Subscription. The following sample request contains a Timeout of five (5) seconds and limits the total number of messages in the response to two (2).

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:tet="http://www.onvif.org/ver10/events/wsdl" >
  <SOAP-ENV:Header>
```

```
    <wsa:Action>
http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/PullMessagesRequest
</wsa:Action>
    <wsa:To>http://160.10.64.10/Subscription?Idx=0</wsa:To>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tet:PullMessages>
      <tet:Timeout>
      PT5S
      </tet:Timeout>
      <tet:MessageLimit>
        2
      </tet:MessageLimit>
    </tet:PullMessages>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

9.12.6 PullMessagesResponse

The following PullMessageResponse contains two notifications which match the subscription. The Response informs the client that two objects have crossed lines corresponding to rules “MyImportantFence1” and “MyImportantFence2”.

```
<?xml version="1.0" encoding="UTF-8"?>
  <SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:wstop="http://docs.oasis-open.org/wsn/t-1"
    xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
    xmlns:tet="http://www.onvif.org/ver10/events/wsdl"
    xmlns:tns1="http://www.onvif.org/ver10/topics"
    xmlns:tt="http://www.onvif.org/ver10/schema">
    <SOAP-ENV:Header>
      <wsa:Action>
http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/PullMessagesResponse
      </wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
      <tet:PullMessagesResponse>
        <tet:CurrentTime>
          2008-10-10T12:24:58
        </tet:CurrentTime>
        <tet:TerminationTime>
          2008-10-10T12:25:58
        </tet:TerminationTime>
        <wsnt:NotificationMessage>
          <wsnt:Topic
Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet">
            tns1:RuleEngine/LineDetector/Crossed
          </wsnt:Topic>
          <wsnt:Message>
            <tt:Message UtcTime="2008-10-10T12:24:57.321Z">
              <tt:Source>
                <tt:SimpleItem Name="VideoSourceConfigurationToken"
                               Value="1"/>
                <tt:SimpleItem Name="VideoAnalyticsConfigurationToken"
                               Value="2"/>
                <tt:SimpleItem Value="MyImportantFence1" Name="Rule"/>
              </tt:Source>
              <tt:Data>
                <tt:SimpleItem Name="ObjectId" Value="15" />
              </tt:Data>
            </tt:Message>
          </wsnt:Message>
        </wsnt:NotificationMessage>
        <wsnt:NotificationMessage>
          <wsnt:Topic
```

Termination time is updated, **not by client**

```
          tns1:RuleEngine/LineDetector/Crossed
        </wsnt:Topic>
        <wsnt:Message>
          <tt:Message UtcTime="2008-10-10T12:24:57.789Z">
            <tt:Source>
              <tt:SimpleItem Name="VideoSourceConfigurationToken"
                             Value="1"/>
              <tt:SimpleItem Name="VideoAnalyticsConfigurationToken"
                             Value="2"/>
              <tt:SimpleItem Value="MyImportantFence2" Name="Rule"/>
            </tt:Source>
            <tt:Data>
              <tt:SimpleItem Name="ObjectId" Value="19"/>
            </tt:Data>
          </tt:Message>
        </wsnt:Message>
      </wsnt:NotificationMessage>
    </tet:PullMessagesResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```