A MIP driver framework plugin for Ultinous Video Analytics Platform.

<https://github.com/Ultinous/uvap>

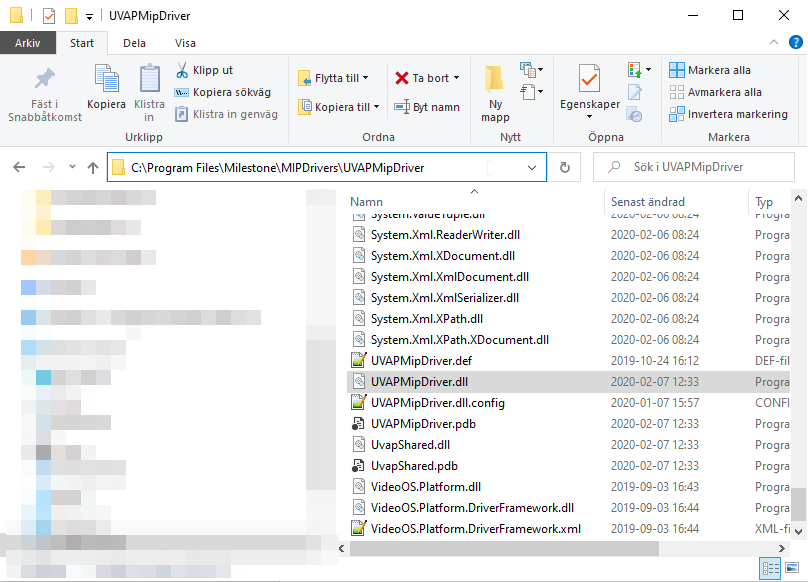
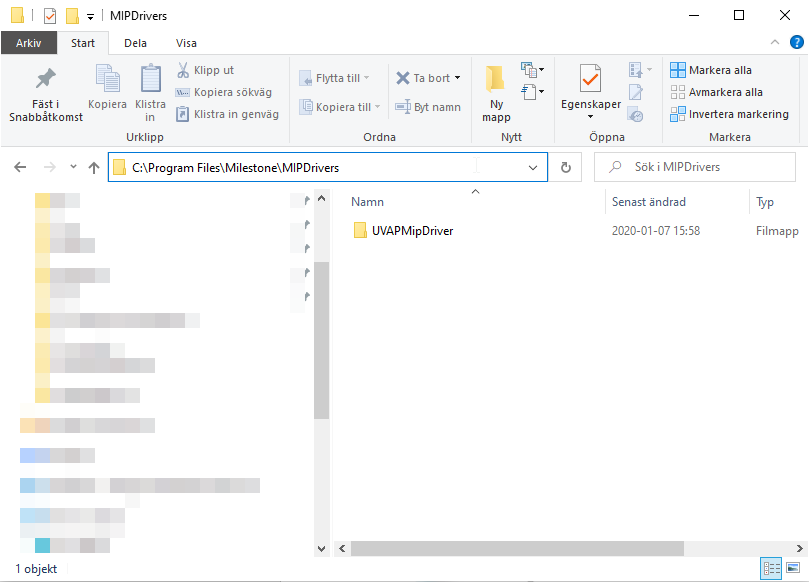
The purpose of the driver is to connect to a Kafka server and fetch data from up to 16 topics and store them in the system.

The driver currently supports 2 formats: head detection and skeleton.

Requirements: Milestone XProtect 2019 R3+RSHotfix or later.

## Installation

Create a new folder in C:\Program Files\Milestone\MIPDrivers on the chosen recording server. Put the compiled files into that folder. Restart the Recording Server service.



## Known issues

The Smart Client UI freeze often. The issue is with Milestone and the Smart Client. We’re currently discussing this issue with Milestone.

Contact: [dev@insupport.se](mailto:dev@insupport.se)

# Workflow

1. Install the UVAP without Kafka in demo BASE mode on an Ubuntu machine

<https://github.com/Ultinous/uvap/blob/master/docs/install/README.md>

Configure the MGR

<https://github.com/Ultinous/uvap/blob/master/docs/demo/demo_overview.md#configuring-uvap>

1. Install Confluent community (confluent-community)

<https://docs.confluent.io/current/installation/installing_cp/deb-ubuntu.html#systemd-ubuntu-debian-install>

Edit /etc/kafka/server.properties and set ‘listeners’ to the IP of the ubuntu machine

listeners=PLAINTEXT://{ip}:9092

Enable autostart for Kafka and Zookeeper

* sudo systemctl enable confluent-zookeeper
* sudo systemctl enable confluent-kafka

Start Zookeeper and Kafka

* sudo systemctl start confluent\*

1. Edit the file ~/uvap/config/uvap\_mgr/uvap\_mgr.prototxt

Set the kafka\_broker\_list to the IP of the Kafka server

environment:

{

debug\_level: 2

profile: false

analysis\_thread\_count: 2

gui: NO

drop\_on: {}

kafka\_broker\_list: "{ip}"

kafka\_topic\_prefix: "base."

}

Don’t close the file.

1. Add the skeleton process node

Under the data\_flow: line you will see a bunch of data\_node’s. Add the highlighted line

data\_node: {type: FRAME name: "input"}

data\_node: {type: DETECTIONS name: "dets"}

data\_node: {type: DETECTIONS name: "dets\_anonymization"}

data\_node: {type: DETECTIONS name: "dets\_filtered"}

data\_node: {type: HEAD\_POSE\_3DS name: "poses"}

data\_node: {type: FEATURE\_VECTORS name: "face\_feats"}

data\_node: {type: SKELETONS name: "skeletons"}

1. Add the skeleton process node like the picture shows

process\_node:

{

type: SKELETON\_ESTIMATOR

name: "skeleton\_estimator"

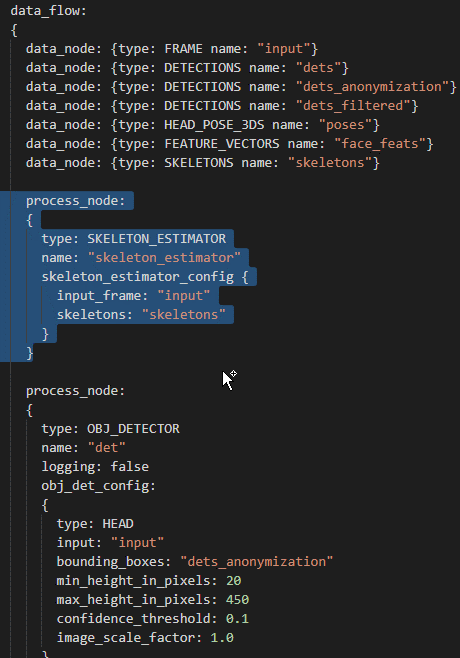
skeleton\_estimator\_config {

input\_frame: "input"

skeletons: "skeletons"

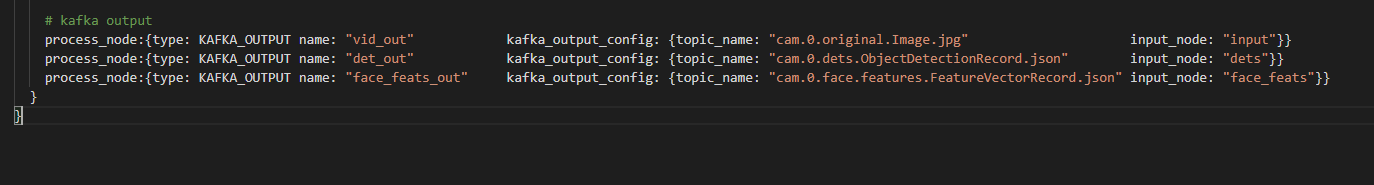
}

}



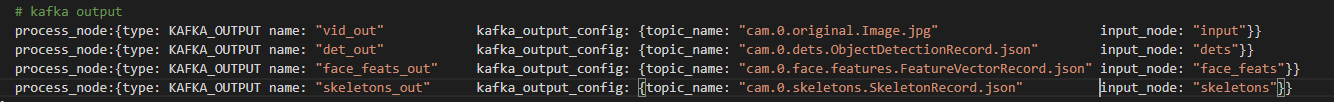
1. Add the skeleton Kafka output

At the bottom of the document you see a bunch of process nodes that look something like this:



Add the line just before the last bracket (})

process\_node:{type: KAFKA\_OUTPUT name: "skeletons\_out" kafka\_output\_config: {topic\_name: "cam.0.skeletons.SkeletonRecord.json" input\_node: "skeletons"}}



Save the document and restart the MGR:

* Docker container restart uvap\_mgr

If everything was performed correctly the UVAP should be configured with both the head detection and skeleton models. Check the Kafka topics with

* kafka-topics --list --zookeeper localhost:2181

*Should return:*

\_\_confluent.support.metrics

\_\_consumer\_offsets

base.cam.0.ages.AgeRecord.json

base.cam.0.anonymized\_original.Image.jpg

base.cam.0.dets.ObjectDetectionRecord.json = Topic for Headdetection model

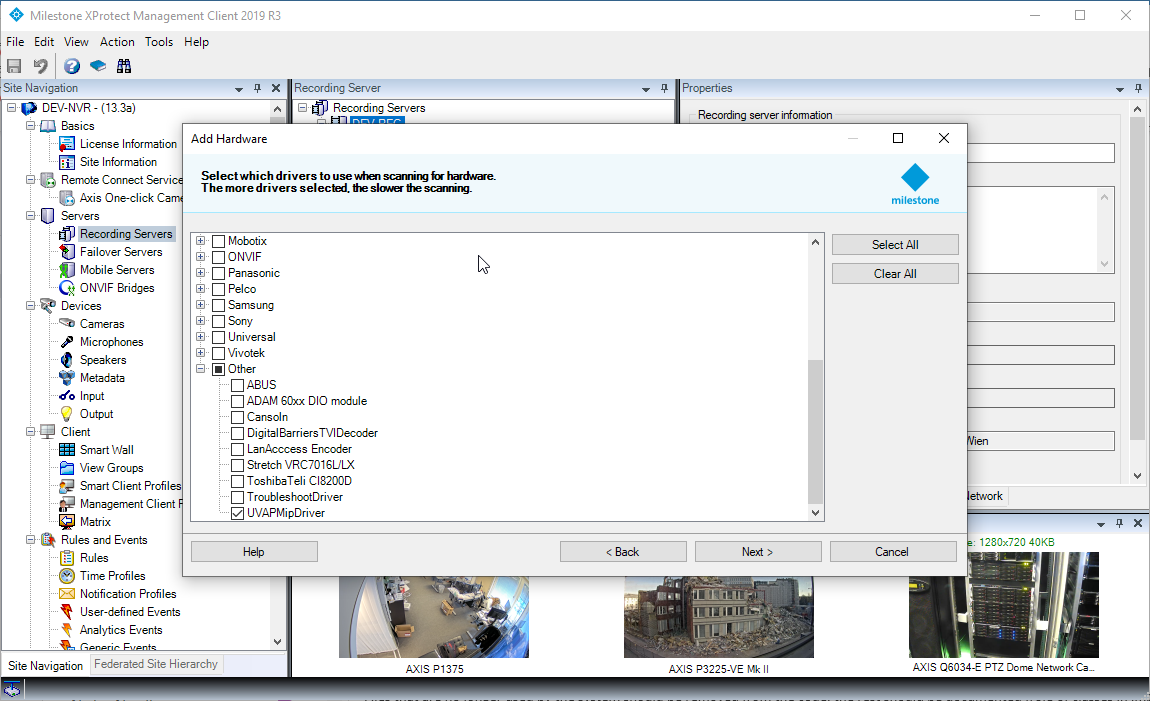
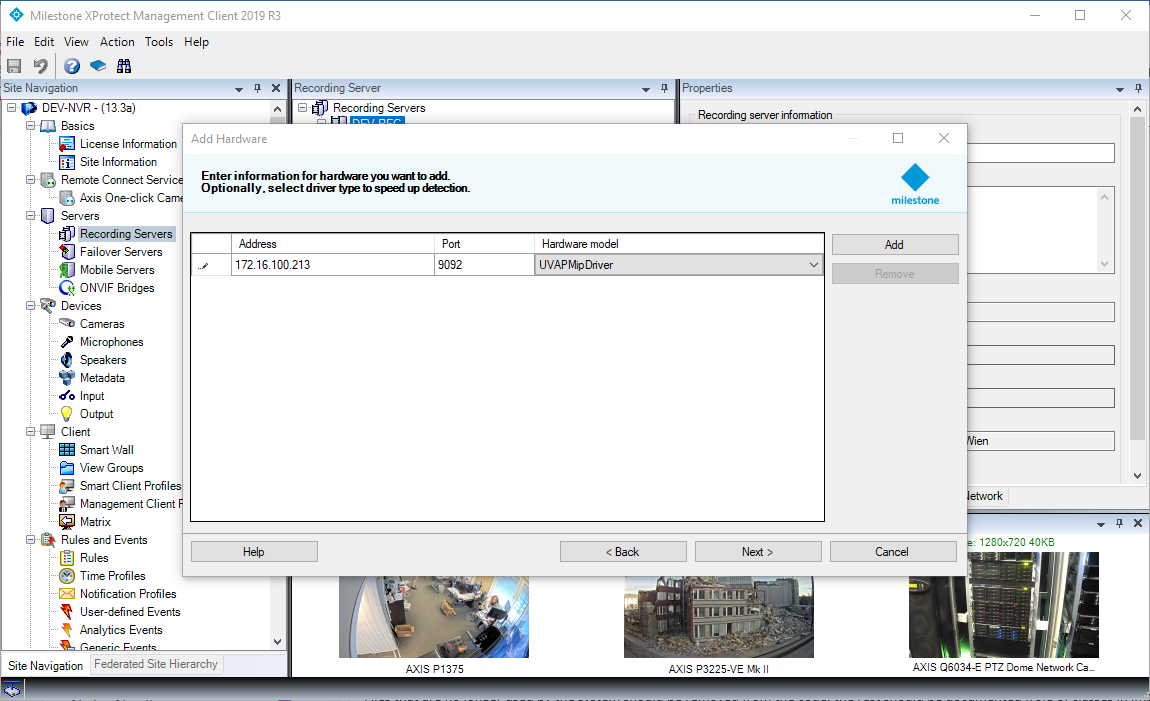
base.cam.0.frameinfo.FrameInfoRecord.json

base.cam.0.genders.GenderRecord.json

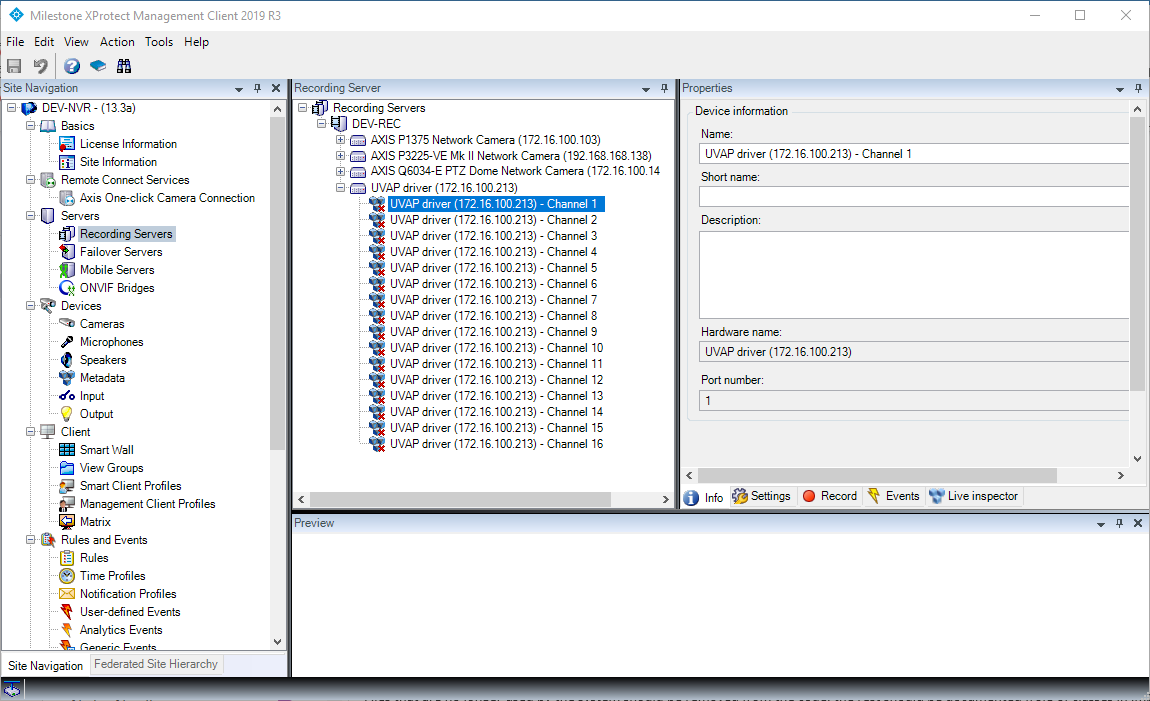
base.cam.0.original.Image.jpg

base.cam.0.poses.HeadPose3DRecord.json

base. cam.0.skeletons.SkeletonRecord.json = Topic for skeleton model

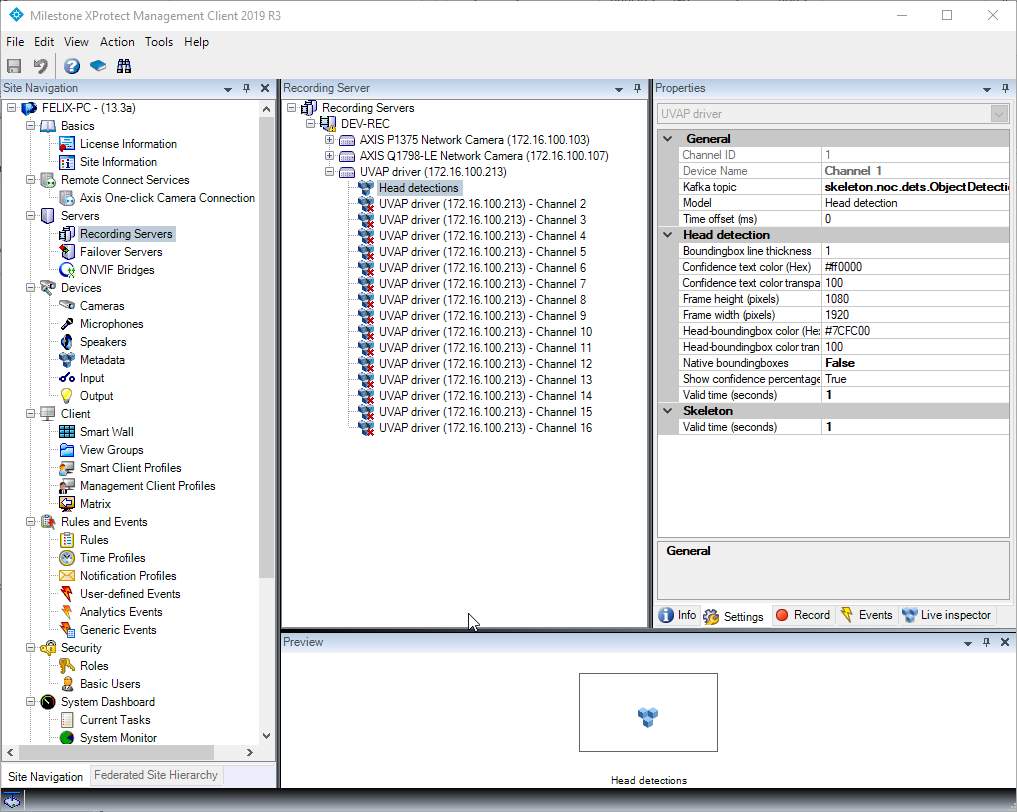
1. [Make sure the driver is installed in C:\Program Files\Milestone\MIPDrivers on the designated recording server hardware.](#_Installation)
2. Go to “Recording Servers” in the Management Client and right click the recording server to use. Press “Add Hardware".
3. Use the Manual mode, press Next. Choose any username and password and press Next again.
4. Select the “UVAPMipDriver” in the “Other” category at the bottom. Press Next.
5. Enter the address of the installed Kafka server where the UVAP is configured to store data. Set the port to the specified port for Kafka (default is 9092). Set the Hardware model to “UVAPMipDriver”. Press Next.
6. The system will try to add the hardware. If a Kafka server is found at the address and port, it will display “Success” under the Status column.

If it failed, go back and make sure the address and port is right, and check if Kafka is running.

1. Press Next a few times and finish. Select a device and start the configuration.

# Configuration

The configuration is set to each individual channel.



### General

Kafka topic: The name of the topic in Kafka

Model: The model used from the specified Kafka topic (Head detection or skeleton)

Time offset: Manual time offset setting for data and video synchronization.

### Head detection

Native bounding boxes:

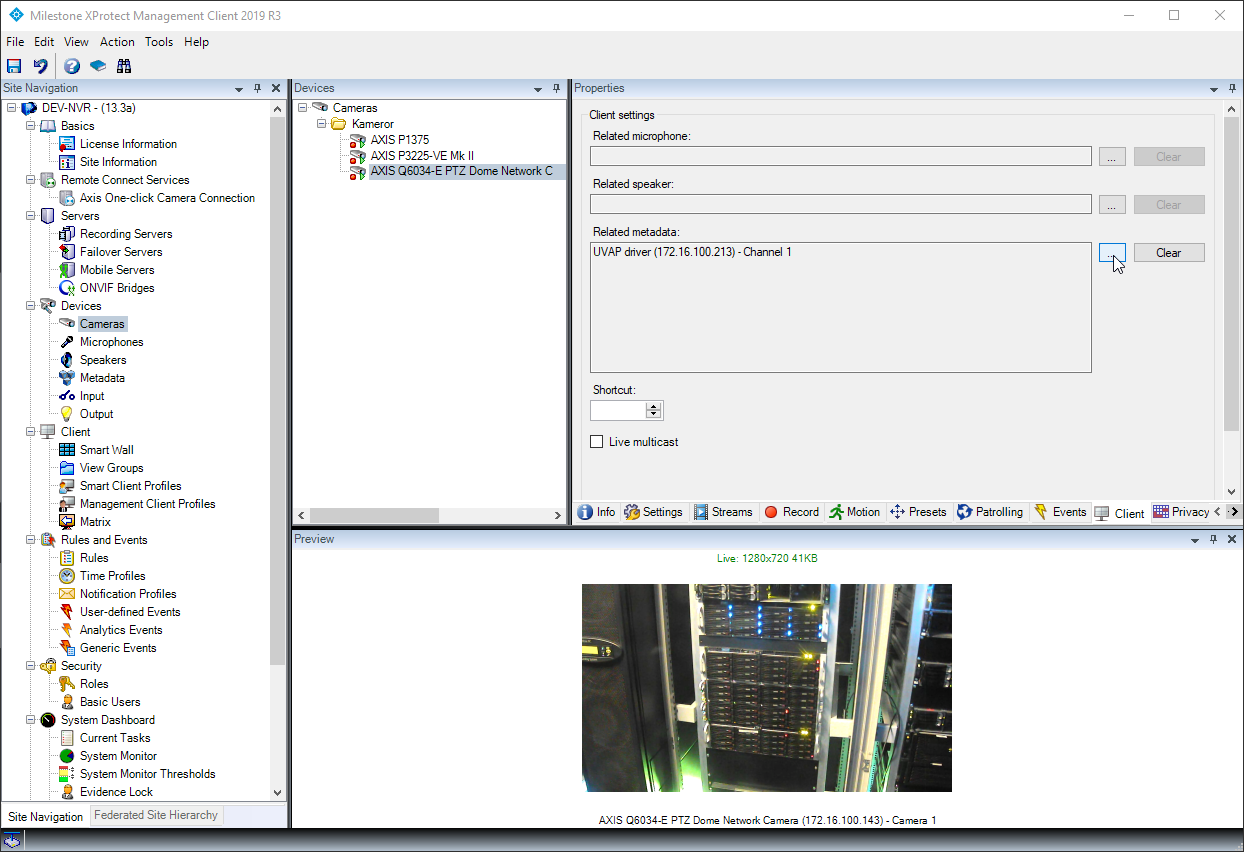
True = The data will be stored in Onvif-XML format. The Smart Client supports this natively and will draw bounding boxes without the UVAPMipPlugin overlay visualization. The native bounding boxes can be tweaked using the settings that speak for themselves. This option is for pure visualization and the post-data search is not supported.

False = The default and recommended value. The data will be stored in a slimmed json format. The data can be read by the UVAPSearchAgent plugin and is visualized by the UVAPMipPlugin overlay visualizer.

Frame height: Used for visualization when native bounding boxes is true.

Frame width: Used for visualization when native bounding boxes is true.

Make sure the device is **Enabled** when the configuration is complete. Go to the camera and add the configured metadata device as a related metadata device.

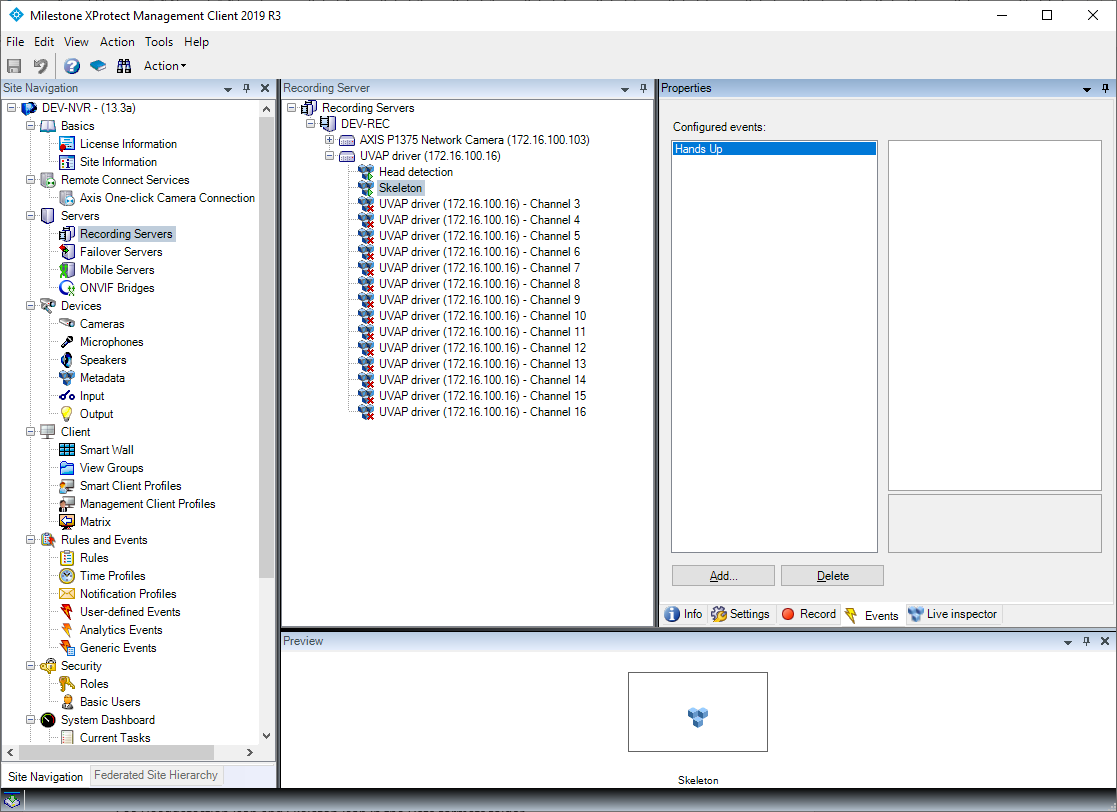


# Hands Up event

This driver contains one built-in real-time event for demonstration purposes. It only works for the skeleton model. The driver checks if a detected skeleton has both wrists above both shoulders.

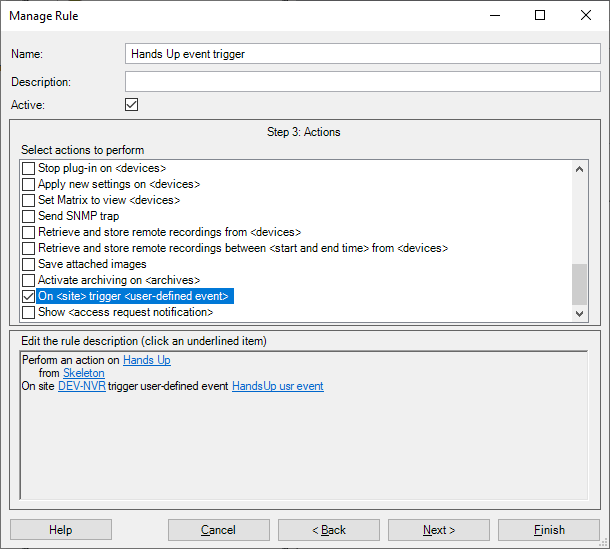
## Setup

Select the metadata device using the skeleton model that’s used for the event. Go to the Events tab and press the Add-button. Select the Hands Up event and press OK. Save the configuration.



Create a user-defined event. This event is going to be triggered by the rule-engine and used when creating alarm definitions.

Go to the Rules section and create a new rule. Use the Hands Up trigger in Devices -> Configurable events. Press Next. If you want a schedule/timeprofile related to this rule, fill this in. Press Next again and select the ‘On <site> trigger user-defined event <event>’. Select the user-defined event you created earlier. Press Finish.



That’s it! Create a alarm definition and use the user-defined event as the trigger.

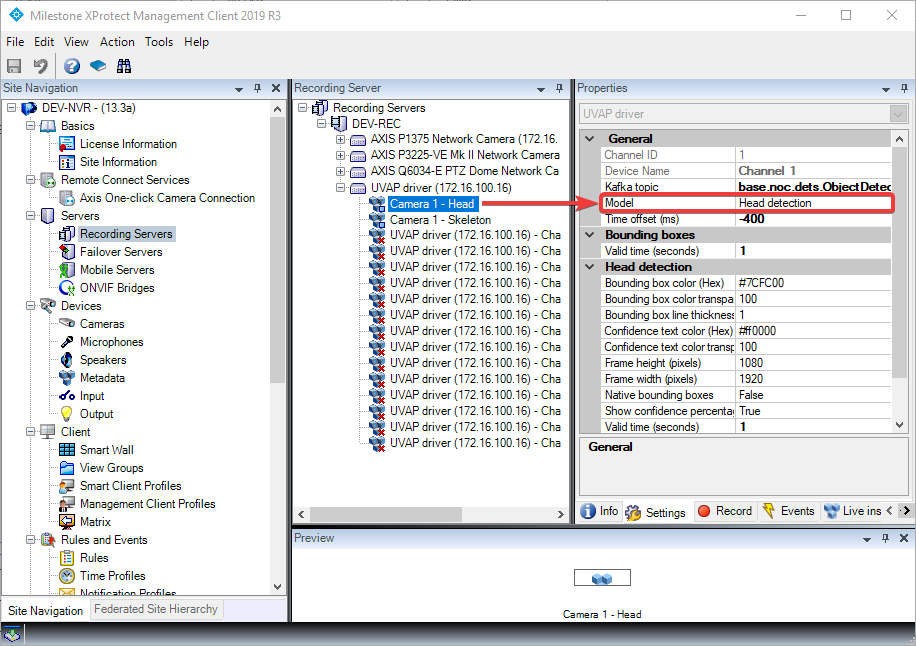
# Data formats

See Headdetection.json and Skeleton.json in the Data formats folder.

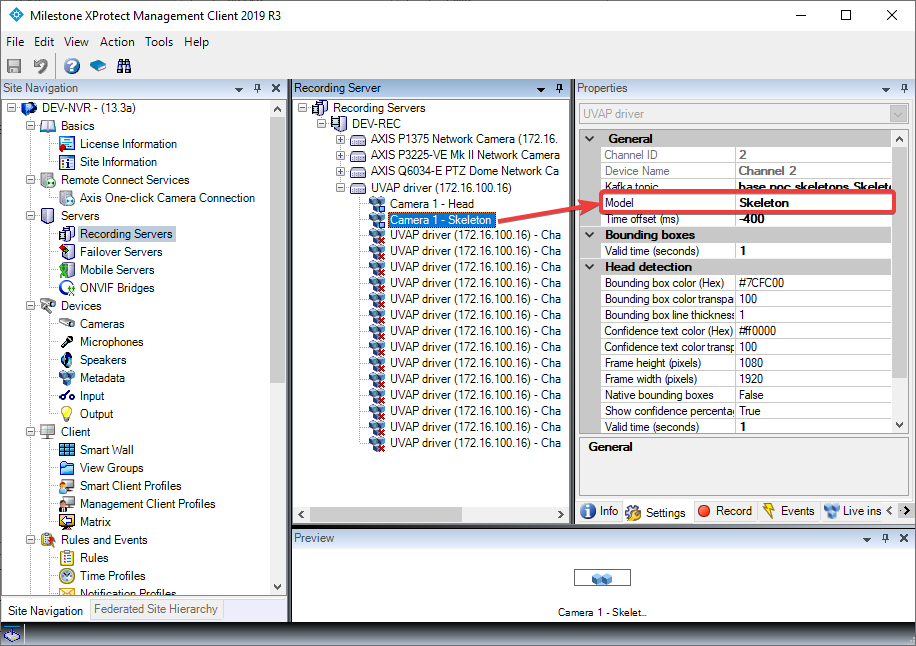
## Limitations

Due to limitations in the SDK, the metadata devices must follow these naming guidelines:

Devices using the head detection format must contain the text “head”, in example “Camera 5 – UVAP Head detection”. This applies to the skeleton format as well and those devices must contain the word “skeleton”.

Head detection model device, the device contains the word “head”:

Skeleton model device, the device name contains the word “skeleton”:



# Dependencies

.NET Framework 4.7 or later (Requires install) <https://dotnet.microsoft.com/>

Confluent.Kafka 1.3.0 (Included in build) <https://github.com/confluentinc/confluent-kafka-dotnet/>

librdkafka.redist 1.3.0 (Included in build) <https://github.com/edenhill/librdkafka>

Newtonsoft.Json 12.0.3 (Included in build) <https://www.newtonsoft.com/json>