

VCA Documentation

v1.5.5r1



Chapter 1

Introduction

This is the user manual for the VCAserver video analytics framework developed by VCA Technology. This manual describes how to set up and configure the video analytics to detect events of interest, whilst minimising false alerts.

VCAserver is available as a server application for Windows or Linux and preloaded on the VCAbridge embedded hardware.

See [Getting Started](#) for the essentials necessary to get VCAserver configured and start generating metadata.

Chapter 2

Getting Started

This user guide documents each topic in detail, and each topic is accessible via the menu. However, to get started quickly, the essential topics are listed below.

2.1 Fundamentals

- **Install or set up** the platform on which VCAcore will run.
- Learn how to **navigate** VCAcore's interface.
- If running VCAcore on the VCAbridge platform, configure the network and time settings by using the **System Settings** page.
- Ensure that VCAcore is licensed for your required functionality by checking the Activation page.
- Create a **video source**.
- Create some **zones** and **detection rules**. **Calibrate** the channel if necessary.
- Create an **action** to send alerts to a third-party system.

2.2 User Credentials

Note that the default username and password for the VCAcore platform are:

- **Username:** admin
- **Password:** admin

2.3 Advanced Topics

Once the basic settings are configured, the following advanced topics may be relevant:

- Set up classifiers by using the **classification** function.

- Detect camera tampering or obscuration by using the **tamper detect** function.
- Learn how the device detects gross scene changes with the **scene change detection** function.
- Customise the annotation that is included in the video display, by using the **burnt-in annotation** settings.
- Adjust advanced settings such as alarm hold off time, detection point and camera shake cancellation by using the advanced settings page.

Chapter 3

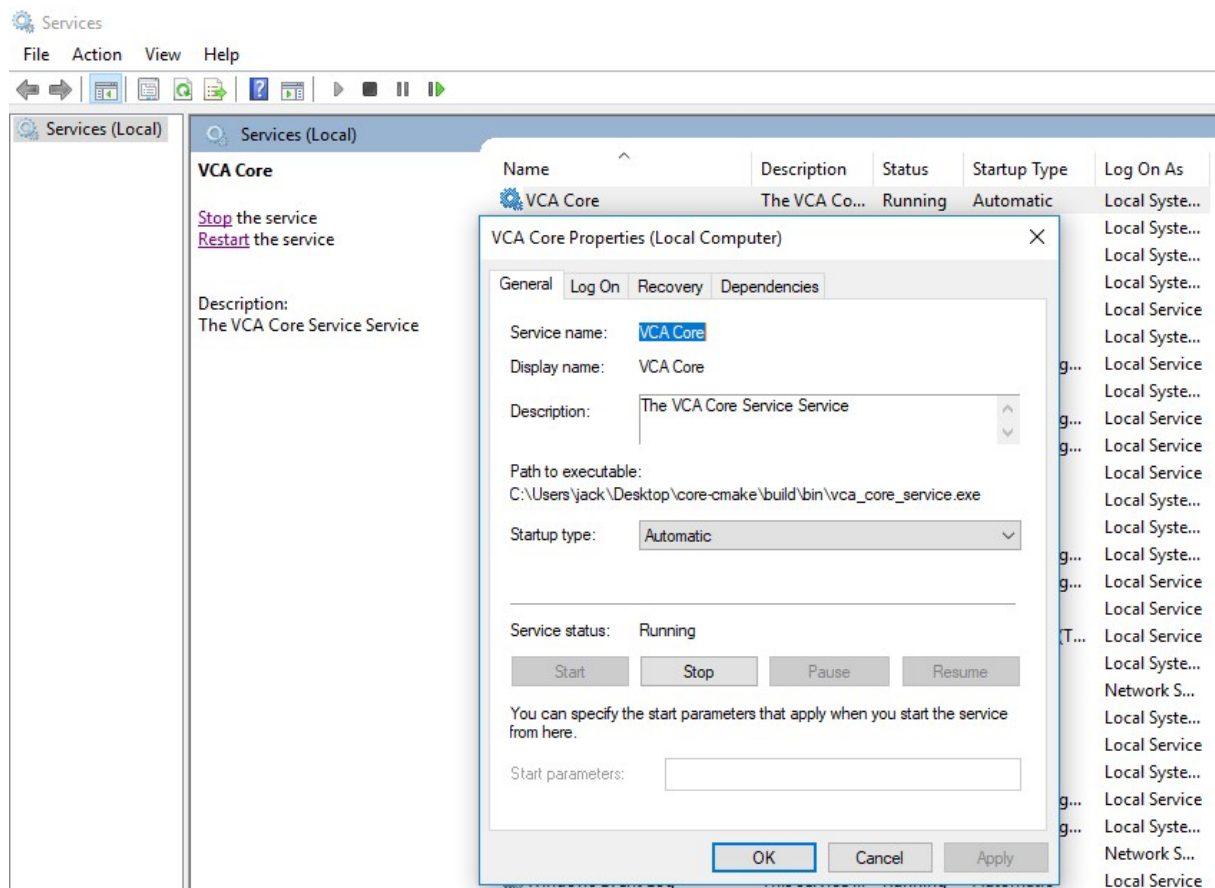
Installing VCAserver

Installation instructions for the various platforms supported by VCAcore vary slightly and are outlined below.

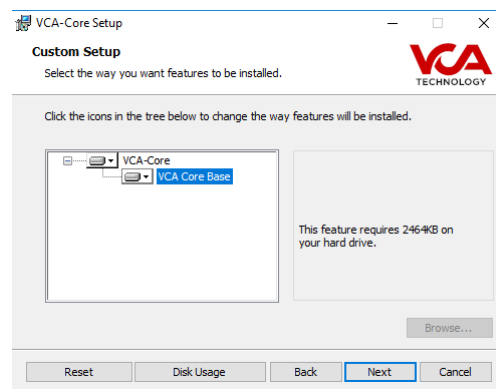
3.1 VCAserver (Windows 10)

VCAserver is installed as a service called **VCA core** and can be managed using the Windows service manager.

The configuration file for VCAcore is stored: C:\VCACore



The VCAserver MSI package installs the base analytics engine, interface and deep learning models.



3.2 VCAserver (Ubuntu 18.04)

VCAserver is installed as a `systemd` service, as such when installation is complete the VCAcore service will be started automatically. The VCAcore service can be managed using the a `systemd` service manager, e.g. `sudo systemctl restart vca-core.service`

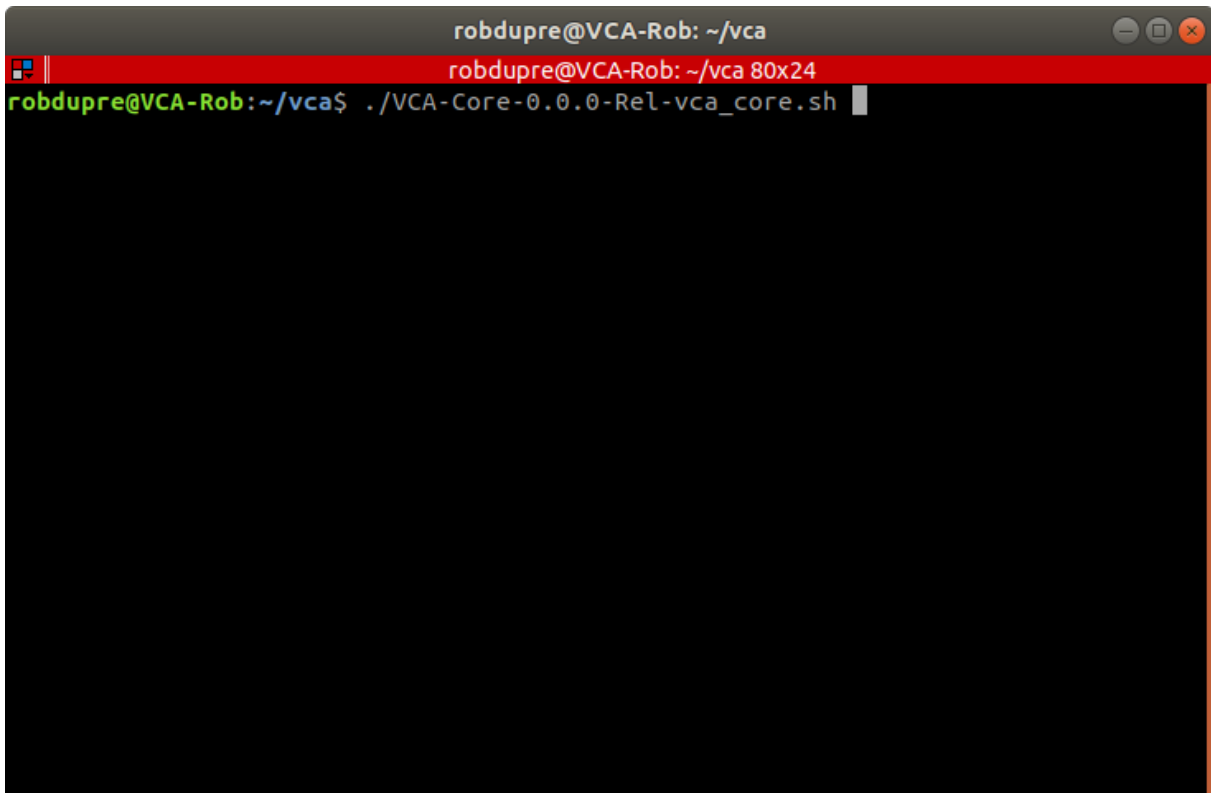
When installed the VCAserver configuration file is stored: `/var/opt/VCA-Core/`

VCAserver on Linux comes as a single archive file containing an `.sh` script, which handles the installation of the VCAcore components. Once the archive has been downloaded, navigate to folder and unpack the installation scripts from the archive:

- `VCA-Core-**VERSION_NUMBER**-vca_core`

Change file attributes to allow the script to run. `chmod +x ./VCA-Core-**VERSION_NUMBER**-vca_core` (for example e.g. `chmod +x ./VCA-Core-1.5.0-vca_core.sh`)

Next run the `.sh` script: `sudo ./VCA-Core-**VERSION_NUMBER**-vca_core`, (for example e.g. `sudo ./VCA-Core-1.5.0-vca_core.sh`).

A terminal window titled 'robdupre@VCA-Rob: ~/vca' with a red header bar. The prompt is 'robdupre@VCA-Rob: ~/vca 80x24'. The command being executed is './VCA-Core-0.0.0-Rel-vca_core.sh'. The terminal background is black, and the text is green and white. The window has standard Linux window controls (minimize, maximize, close) in the top right corner.

```
robdupre@VCA-Rob: ~/vca
robdupre@VCA-Rob: ~/vca 80x24
robdupre@VCA-Rob: ~/vca$ ./VCA-Core-0.0.0-Rel-vca_core.sh
```

VCAserver should be installed as a system service. The install directory is fixed to `/opt/VCA-Core/` and will request desired ports for VCAserver's recovery and web servers.

```
robdupre@VCA-Rob: ~/vca
robdupre@VCA-Rob: ~/vca 80x24
B. "Software" means the computer programs and documentation listed and described
  in The Appendix (Licensed Software) attached to this Agreement, as well as any
  archival copies of such computer programs and documentation permitted by this Ag
  reement.
C. "Install" means placing the Software on a computer's hard disk, CD-ROM, flash
  memory or other secondary storage device.

Do you accept the license? [yN]:
y

Install as system service (requires root permissions) [yN]:
y
The archive will be extracted to: /opt/VCA-Core

Using target directory: /opt/VCA-Core
Extracting, please wait...

Unpacking finished successfully

Enter VCACore web server port, or leave blank for default (8080):

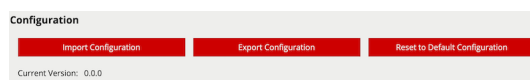
Enter VCACore recovery server port, or leave blank for default (9090):
```

Important notes: VCAserver is developed and tested against Windows 10 and Ubuntu 18.04. Although the application may run on other versions of Windows or Linux, support is limited to this version only.

3.3 Upgrading VCAserver

Periodically new versions of VCAserver will be released with new features, please check for software updates at the [VCA Support page](#).

When upgrading VCAserver, backup the system configuration (System -> Configuration), uninstall or delete the existing VCAserver version and run the new installation packages as above.

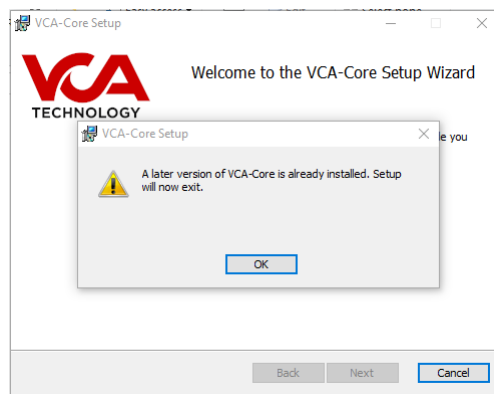


When the upgrade is complete the configuration is persisted and upgraded to work with the new version.

3.3.1 Downgrading VCAserver

Downgrading to a previous version of VCAserver is not supported. If a previous version is required, the existing installation and configuration must be deleted before the desired version is installed. Windows

systems will raise an error during install in this case.



3.4 Deep Learning Requirements

VCAserver supports GPU acceleration for the deep learning features with the following requirements:

1. A NVIDIA GPU with CUDA Compute Capability 7.5 or higher
2. The latest NVIDIA graphics drivers (at least 460.73 or higher). This can be checked using `nvidia-smi`.
3. NVIDIA's CUDA Toolkit 11.1 **must** be installed on the server. This can be checked with `nvcc -V`.

```
robdupre@rob-desktop: ~  
robdupre@rob-desktop: ~ 80x24  
robdupre@rob-desktop:~$ nvidia-smi  
Fri Apr 5 15:35:07 2019  
+-----+  
| NVIDIA-SMI 390.87                  Driver Version: 390.87          |  
+-----+-----+-----+-----+  
| GPU  Name            Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |  
| Fan  Temp  Perf    Pwr:Usage/Cap|     Memory-Usage | GPU-Util  Compute M. |  
+-----+-----+-----+-----+  
| 0   GeForce GTX 750 Ti    Off   | 00000000:01:00.0  On   |      N/A              |  
| 35%   42C   P8      2W / 38W   | 720MiB / 1999MiB |    58%      Default  |  
+-----+-----+-----+-----+  
+-----+  
| Processes:                                                       GPU Memory |  
|  GPU       PID    Type    Process name                       Usage    |  
+-----+-----+-----+-----+  
|    0       1722     G   /usr/lib/xorg/Xorg                     358MiB |  
|    0       2200     G                   compiz                      10MiB |  
|    0       2782     G                   compiz                      158MiB |  
|    0       3933     G   ...uest-channel-token=13833695216301488359  154MiB |  
|    0      16965     G   /snap/pycharm-community/123/jre64/bin/java    2MiB |  
|    0      29635     C       gimp-2.8                       29MiB |  
+-----+  
robdupre@rob-desktop:~$
```

In the absence of a correctly configured or installed GPU, VCAserver will either default to running the deep learning features on the CPU or, algorithms with a strict GPU requirement, will not be available.

Chapter 4

VCACore Service

VCAserver, on both Windows and Ubuntu, has a management service built in. This utility allows control of the VCAserver application via a Web UI to allow simple remote management.

The recovery service is always running and by default is accessible at: `http://[HOST_COMPUTER_IP]:9090/`.

The recovery service provides a range of functionality.

4.1 Info

Details of the logs and configuration locations for the currently running instance of VCAserver.

VCA Core Recovery	

Info	
Logs Path	
/opt/VCA-Core/logs	
Configuration Path	
/opt/VCA-Core/configuration	

A failure count is provided which will keep track of the number of times the VCACore application has restarted.

Lastly, the current status of the VCAserver application is also provided.

Failure count	0

Current Status	Status: running

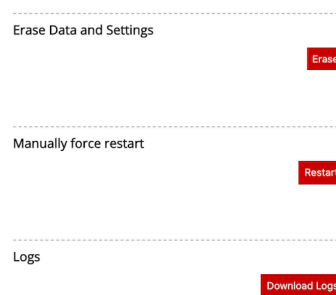
4.2 Managing VCAserver

The main function of the VCAcore Service is to manage the VCAserver application. By default VCAserver is always running. To stop the application, press the **Stop** button which will allow the service to perform additional management tasks.



Once stopped, the VCAcore Service is able to erase data and settings, resetting the VCAserver application and configuration back to a default state. An option is also available to download the log files.

Lastly to restart the VCAserver application click the **Restart** button.



Chapter 5

Navigation








This topic provides a general overview of the VCAserver configuration user interface.

5.1 Navigation Bar


The VCA user interface features a persistent navigation bar displayed at the top of the window.

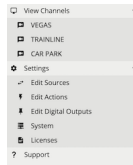



There are a number of elements in this navigation bar, each is described below:

- : The menu icon. Click the menu icon to access the [Side Menu](#) for convenient navigation through the application.
- : The view channels icon. Click the icon to access the [View Channels](#) page.
- : The configuration state icon. Displays the current saved state of the system. Changes to the system are saved and synchronised automatically, the configuration state icon indicates the current state of synchronisation. The configuration state can be in one of two states:
 - : All configuration changes have been successfully synchronized between web interface and the device.
 - : The configuration is currently being synchronised between the web interface and the device.
- : The settings icon. Click to access the [Settings Page](#).
- : The arm/disarm icon. Click to change the [Arm/Disarm State](#) of the device.

5.2 Side Menu

Clicking the  icon displays the side navigation menu:

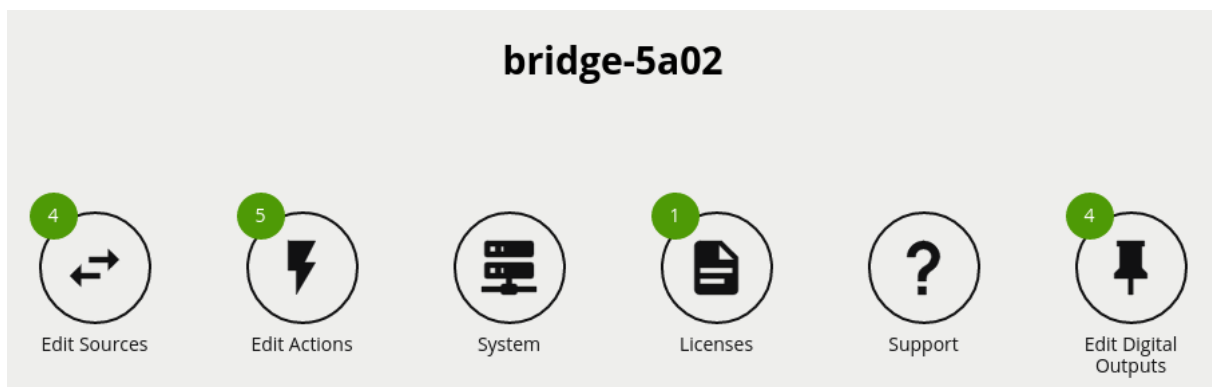


Every page in the VCA user interface is accessible through the side menu. The  icon next to a menu item indicates that the item has sub-items and can be expanded.

Items in the side menu are automatically expanded to reflect the current location within the web application.

5.3 Settings Page

The settings page displays a number of links to various configuration pages:



- **Edit Sources:** Go to the [Sources](#) configuration page.
- **Edit Actions:** Go to the [Actions](#) configuration page.
- **System** Go to the [System Settings](#) page.
- **Licenses:** Go to the [\[Activation\]](#) page to add or remove VCA
- **Support:** Go to the [Support](#) page.
- **Edit Digital Outputs** Go to the [Digital Output Device Configuration](#) page.

Chapter 6

Licensing

To create sources and take advantage of the VCAserver analytics, each channel will require a license.

6.1 License Server

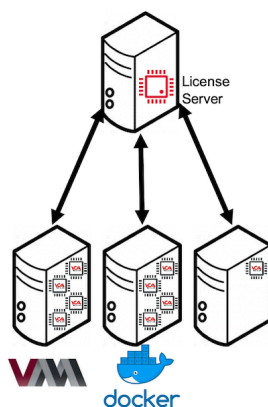
Licensing is managed by a license server that is installed alongside VCAserver. The server validates a license's **activation code** against a generated unique identifier specific to the physical hardware the server is running on called a **hardware code**. An **activation code** is only valid for a specific **hardware code** and the license server will only generate a **hardware code** on physical (non-virtualised) systems.

VCAserver connects to an instance of the license server to retrieve licenses for use with its sources. The connection is either to a local instance or to a remote instance of the license server across the network. As a result, two primary architecture configurations are possible:

Local: where the license server and VCAserver are running on a single physical hardware device.



Remote: where instances of VCAserver (either running on physical hardware or virtualised) connect to a license server (running on physical hardware) across the network. In this configuration, all connected instances of VCAserver will use a single license pool. All licenses will be validated against the license server's **hardware code**.



The license server uses port 15769. In cases where a remote configuration is required this port will need to be open between the VCAserver and license server.

6.2 License Management

To manage VCAserver's connection to the license server, navigate to the license settings page. This interface allows the user to specify the location of the License Server to use with this instance of VCAserver. Additionally, overview and management tools for the License Server's license pool are provided including the license types, their features, total and available licenses.

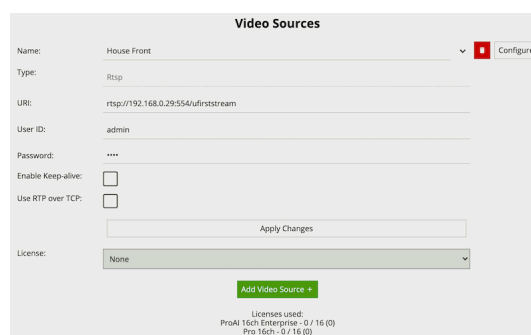
- **License Server Location:** Allows the user to specify a **Local** or **Remote** license server.
- **License Server :** The address of the License server. When **Local** this will be set to `localhost`, when **Remote** an network address will need to be specified where an instance of the License server is run.
- **Hardware Code:** The unique code for the License Server's hardware. Required to generate an activation code.
- **Activation Code:** Enter an activation code to enable additional functionality or channels.
- The list of installed licenses and features are displayed underneath.

On new installations, the license server will need a license added using an **activation code**. This code is linked to your hardware configuration, which will be provided by the reseller. In the case of an upgrade or on systems that have run the license server before, the system will persist the licenses.

Many licenses of different types can be added to an instance of VCAserver and specific **Source** of analytics can be assigned a license based on the required features.

6.3 Connecting to a License Server

When switching VCAserver's License Server from **Remote** to **Local** (or vice-versa), all licenses attached to sources must be removed by changing a **source's** license to **None**. A summary of the total Licenses used is also provided, this should read 0.



The screenshot shows the 'Video Sources' configuration window. It contains the following fields and controls:

- Name:** House Front (with a dropdown arrow and a red 'Configure' button)
- Type:** Rtp
- URI:** rtsp://192.168.0.29:554/ufirststream
- User ID:** admin
- Password:** (masked with four dots)
- Enable Keep-alive:** ☐
- Use RTP over TCP:** ☐
- Apply Changes** button
- License:** None (with a dropdown arrow)
- Add Video Source +** button
- Licenses used:** ProAI 16ch Enterprise - 0 / 16 (0)
Pro 16ch - 0 / 16 (0)

Once no licenses are being used, the user is able to change the **License Server Location**.

6.4 Managing License Pool

To add a license to the License Server's pool:

- Copy the hardware code and send it to the hardware reseller.
- The reseller sends an activation code by return.
- Apply the activation code to the device and verify the required features are activated.

Licenses can also be deleted from the License Server's pool (in the case of expired evaluation licenses or tokens):

- Set license to **None** on any sources that makes use of the license type to be deleted.
- Click red delete button next to license type.

6.5 More Information

For more information on the complete range of additional features available, please visit **VCA Technology**

Chapter 7

Sources


Sources are user configured inputs to VCAserver, which include video sources and non-video sources (e.g. digital inputs). The Edit Sources page allows users to add/remove sources and configure existing sources.

Video Sources

Name:

Video Source 1


<



Name:

Video Source 2

<



Add Video Source +

Licenses used:

Bridge 16ch - 2 / 16


Other Sources

Name:

Other Source

×

<



Add Other Source +

Common Properties:

- **Type:** The type of the source, e.g. File, RTSP, Interval etc.
- **Name:** The user configured name for the source.
- **License:** Drop down list of available licence types.

7.1 Video Sources

Video sources are automatically linked with a **channel** when added. The number of video sources which can be added to the system is dependant on the user's license. A list of the currently available license types (e.g. Pro) and the number of those licenses used is provided (e.g. 2 / 16).

Licence selection allows for a specific licence type to be associated with a channel. Licences can be changed on a video source at any time however, once a channel is configured with rules and functions linked to a particular licence type, changing the licence type for that channel is not advised.

7.1.1 File

File sources enable the streaming of video from a file located in a `test-clips` folder on the host machine. The folder is in a subdirectory of the installation location:

- Linux: `/opt/VCA-Core/share/test-clips/`
- Windows: `c:\Program Files\VCA-Core\share\test-clips\`

Any video files located in this folder will be presented in the File drop down menu. Please note that when files are added to this folder, the web interface will need to be refreshed for the UI to see the files in the drop down menu.

Name:	My File Source	▼	
Type:	File		
File:	las-vegas.mp4 ▼		
License:	Bridge 16ch ▼		


Properties:

- **File:** The name of the file.

7.1.2 RTSP

The RTSP source streams video from remote RTSP servers such as IP cameras and encoders. The minimum frame rate required for good quality tracking is 15fps. The suggested resolution for these RTSP streams is 480p or greater.

Note: resolutions greater than 480p will result in greater CPU resource usage and may not always result in greater tracking accuracy.

Name:	My RTSP Source	▼ 
Type:	Rtsp	
URI:	Enter URI of source	
User ID:	Enter user id/username if authentication is required	
Password:	Enter Password if required	
Enable Keep-alive:	<input type="checkbox"/>	
Use RTP over TCP:	<input type="checkbox"/>	
License:	Bridge 16ch ▼	

Properties:

- **URI:** The fully qualified URI to the remote RTSP endpoint.
- **User ID:** The username to access the RTSP endpoint (if applicable).
- **Password:** The password to access the RTSP endpoint (if applicable).
- **Enable Keep-alive:** Periodically sends commands to the RTSP source to keep the RTSP connection alive. Most RTSP sources require this option to be enabled. Disable this option only if the RTSP source does not support this behaviour.
- **Use RTP over TCP:** Force the RTP stream to be streamed via TCP (instead of UDP). Wherever possible, RTSP streams should be delivered over TCP and not UDP. This is because UDP transport is not robust against packet loss and dropped packets cause corrupt video which causes VCAserver to generate false positives.

7.1.3 Supported Video Sources

The range of video codecs supported by VCAserver is given below:

- H.264
- H.265
- MPEG4

Note: where supported, the following H.264 profiles can be decoded using hardware acceleration: ConstrainedBaseline, Main, High, MultiviewHigh and StereoHigh

When using an RTSP stream as a source please ensure it is encoded with one of the supported compression formats. Likewise, when using a file as a source please note that VCAserver is compatible with many video file containers (.mp4, .avi etc.) but the video file itself must be encoded with one of the above supported compression formats.

7.2 Other Sources

Various non-video sources are available to the user. Once added, these sources can then be assigned to **Actions** and in certain cases referenced in the **Rules**.

7.2.1 Interval

Interval sources can be used to generate events periodically, e.g. a heartbeat to check that the device is still running.



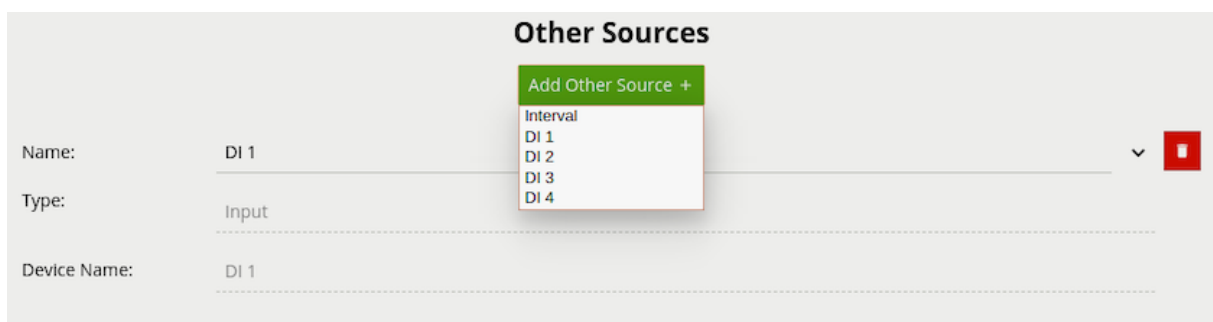
The screenshot shows a configuration form for an interval source. It has three fields: 'Name' with the value 'My Interval Source', 'Type' with the value 'Interval', and 'Interval' with the value '1' and the unit 'seconds'. There is a red square button with a white minus sign in the top right corner.

Properties:

- **Interval:** The interval in seconds between generated events.

7.2.2 Digital Input

If digital input hardware is available, these will show in the list of other sources.



The screenshot shows a configuration form for a digital input source. It has three fields: 'Name' with the value 'DI 1', 'Type' with the value 'Input', and 'Device Name' with the value 'DI 1'. There is a red square button with a white minus sign in the top right corner. A dropdown menu is open, showing the following options: 'Interval', 'DI 1', 'DI 2', 'DI 3', and 'DI 4'. The 'Add Other Source +' button is also visible.

Properties:

- **Device Name:** The name of the DI device.

7.2.3 Armed

The Armed source generates an event when the system becomes **armed**.

7.2.4 Disarmed

The Disarmed source generates an event when the system becomes **disarmed**. Note that any actions that this source is assigned to must be set to **Always Trigger**, otherwise the action will not be triggered due to the system being disarmed.

7.2.5 HTTP

The HTTP source creates an arbitrary REST API endpoint with a state variable that can be set **true** or **false**. This creates a virtual Digital Input which third party systems can enable or disable. The HTTP source can be referenced by the [Source Filter] in a rule graph.

Other Sources

Name:

New HTTP

▼

■

Type:

Http

Endpoint URL:

http://192.168.1.27:8080/api/observables/1/state

Add Other Source +

Properties:

- **Endpoint URL:** The REST API endpoint defining the state variable.

7.2.6 Schedule

The Schedule source allows the definition of a schedule of time when the source is either **on** or **off**. The Schedule other source can be referenced by the [Source Filter] in a rule graph. Additionally, the schedule source can be used to directly control the armed state of VCAserver.

Name:

New Schedule

▼

■

Type:

Schedule

Schedule:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
M																								
T																								
W																								
T																								
F																								
S																								
S																								

Set Arm/Disarm:

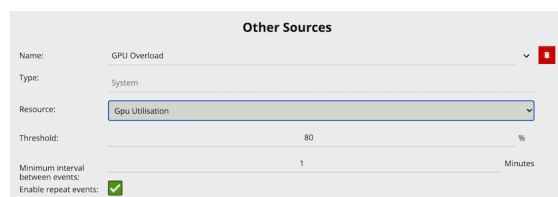
☒

Properties:

- **Schedule:** A click and drag interface which allows the definition of on periods (in green) and off periods (in grey). Each row represents one of the seven days in a week and each column represents a half hour period in that 24 hours.
- **Set Arm/Disarm:** When checked, the schedule source directly sets VCAserver's Armed state according to the schedule defined above.

7.2.7 System

The System source generates an event when the selected system resource goes above the user defined threshold. The source can be configured to continue to send events, whilst the resource remains above the threshold, at a set interval or to send a single event each time the threshold is reached.



The screenshot shows a configuration window titled "Other Sources". It contains the following fields:

- Name:** GPU Overload
- Type:** System
- Resource:** Gpu Utilisation
- Threshold:** 80 %
- Minimum interval between events:** 1 Minutes
- Enable repeat events:** ☒

Properties:

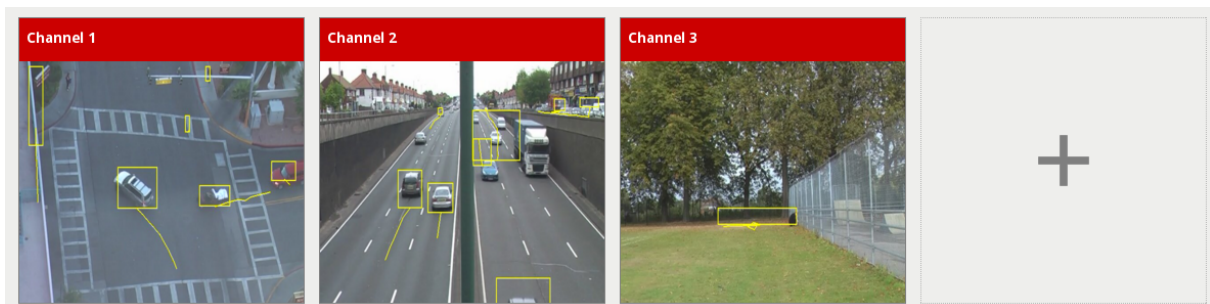
- **Resource:** A defined list of system resources that the system source can monitor.
- **Threshold:** The percentage (%) usage required to trigger an event.
- **Minimum interval between events:** The period of time before a repeat event is sent (if the resource is still above the threshold).
- **Enable repeat events:** When enabled the system will continue to trigger events (at the minimum interval) while the resource is above the threshold. When disabled, a single event will trigger each time the threshold is met.

Chapter 8

Channels

8.1 View Channels Page

The **View Channels** page displays a preview of each configured channel along with any event messages.




Click a thumbnail to view the channel and configure VCAserver related settings. Click the plus icon to go to the add **video source** page.

8.2 Channel Pages

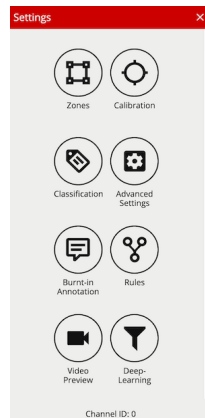
After clicking on a channel, a full view of the channel's video stream is displayed along with any configured **zones**, **counters** and **rules** and the channel settings menu open.



If the settings menu is closed, a tab with a  icon is displayed on the right hand side of the page. Click this to reopen the channel settings menu.

8.2.1 Channel Settings Menu

This menu contains various useful links for configuring various aspects of the channel:



- **Zones**
- **Calibration**
- **Classification**
- **[Advanced Settings]**
- **Burnt-in Annotation**
- **Rules**
- **[Video Preview]**
- **[Deep-Learning]**

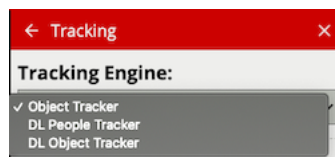
Chapter 9

Trackers

VCAserver supports a number of tracker technologies for use with a configured channel of video. The available trackers are listed below:

- Object Tracker
- Deep Learning People Tracker (DLPT)
- Deep Learning Object Tracker (DLOT)

Under the Trackers menu item is a drop down menu option for Tracking Engine; under which, one of the available trackers can be selected.



9.1 Initialisation

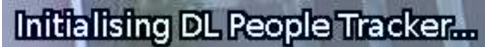
When a tracker is selected by the user, an initialisation phase will be required. This will vary based on the selected tracker.

- **Object Tracker:** when selected the tracker will need to 'learn the scene' to determine background from moving foreground objects. Whilst learning the scene the following message will be displayed in the live view, and no objects will be tracked during this time.

Learning Scene

- **DL People Tracker & DL Object Tracker:** when first selected, the DL engine will run a model generation process. This optimises the DL models to run on the available GPU hardware. Irrespective of which tracker is selected, the DL People tracker model, DL Object Tracker model and the DL Filter model will all be optimised in one go. This process can take up to 10 minutes per model and may increase with different GPU configurations. Once complete the optimised models are stored in the configuration folder. The process will not need to be run again unless the GPU

hardware is changed. Whilst optimisation is performed a message will be displayed in the live view, and no objects will be tracked during this time.



Please note: The DL Filter requires the same initialisation process but does not display a message.

Once initialised, VCAserver will begin analysing the video stream with the selected tracker. Settings specific to that tracker will also be displayed below the tracker engine selection option.

Regardless of the tracker selected, any tracked object can be passed through the available rules. However, in some cases, certain rules or algorithms will only be available with a specific tracker. For example, Deep Learning Filter and the abandoned and removed object rules are only available with the Object Tracker.

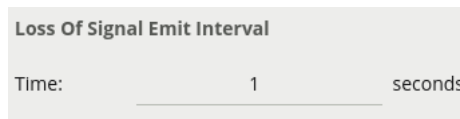
9.2 Universal Settings

Some settings are universal across all trackers, these are outlined below:

9.2.1 Loss Of Signal Emit Interval

The Loss Of Signal Emit Interval defines the amount of time between emissions when a channel loses signal to it's source.

The default setting is 1 second.



9.2.2 Tamper Detection

Learn more about [Tamper Detection](#).

9.3 Object Tracker

The Object Tracker is a motion based detection engine. Based on changes detected in the image, the algorithm separates the image into foreground and background, tracking any foreground object that is moving above a set threshold. The Object Tracker has the following settings:

9.3.1 Stationary Object Hold-on Time

The Stationary Object Hold-on Time defines the amount of time an object will be tracked by the engine once it becomes stationary. Since objects which become stationary must be “merged” into the scene

after some finite time, the tracking engine will forget about objects that have become stationary after the Stationary Object Hold-on Time.

Stationary Object Hold-on Time		
Time:	5	seconds

The default setting is 60 seconds.

9.3.2 Abandoned / Removed Object Threshold

This threshold amount of time an object must be classed as abandoned or removed before an Abandoned / Removed rule will trigger.

Abandoned Object Threshold		
Time:	5	seconds

The default setting is 5 seconds.

9.3.3 Minimum Tracked Object Size

The Minimum Tracked Object Size defines the size of the smallest object that will be considered for tracking.

For most applications, the default setting of 10 is recommended. In some situations, where extra sensitivity is required, the value can be manually specified. While lower values allow the engine to track smaller objects, it may increase the susceptibility to false detections.

Minimum Tracked Object Size		
Size:	10	blobmap pixels

9.3.4 Object Tracker Sensitivity

The Object Tracker Sensitivity value allows the object tracker to be tuned to ignore movement below a certain threshold. Combined with the Blob Map burnt in annotation, which visualises the area of the scene the object tracker is detecting movement, this value can be adjusted to filter out environmental noise.

Object Tracker Sensitivity

Threshold:

The default setting is 4.

9.3.5 Scene Change Detection

Learn more about [Scene Change Detection](#).

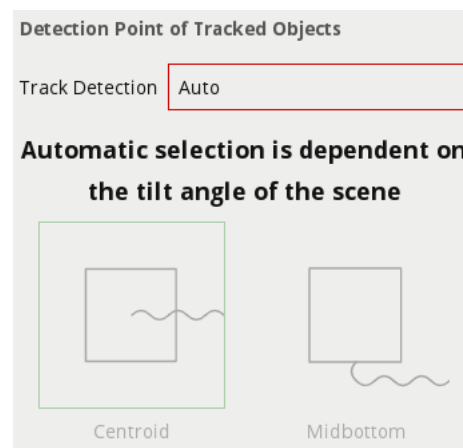
9.3.6 Detection Point of Tracked Objects

For every tracked object, a point is used to determine the object's position, and evaluate whether it intersects a zone and triggers a rule. This point is called the **detection point**.

There are 3 modes that define the detection point relative to the object:

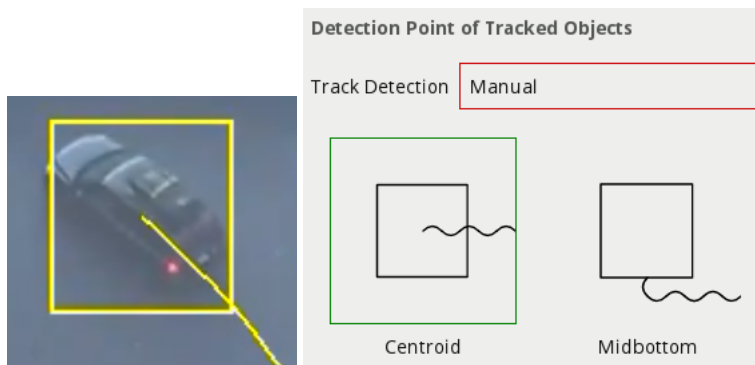
9.3.6.1 Automatic

In automatic mode, the detection point is automatically set based on how the channel is configured. It selects 'Centroid' if the camera is calibrated overhead, or 'Mid-bottom' if the camera is calibrated side-on or not calibrated.



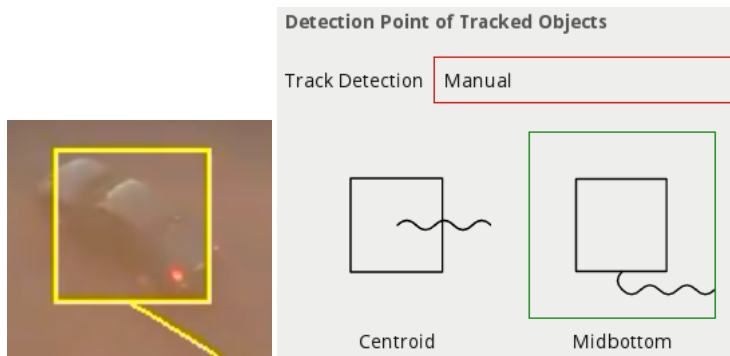
9.3.6.2 Centroid

In this mode, the detection point is forced to be the centroid of the object.



9.3.6.3 Mid-bottom

In this mode, the detection point is forced to be the middle of the bottom edge of the tracked object. Normally this is the ground contact point of the object (where the object intersects the ground plane).



9.3.7 Tamper Detection

Learn more about [Tamper Detection](#).

9.3.8 Loss Of Signal Emit Interval

See [Loss Of Signal Emit Interval](#)

9.4 Deep Learning People Tracker

The Deep Learning People tracker is designed to track people in situations where the camera field of view is relatively close.

The Deep Learning People Tracker is based on Pose Estimation technology, providing the location of a person in the field of view as well as additional key point metadata on the parts of the body. See [Deep Learning Requirements](#) for hardware requirements for this algorithm.

The Deep Learning People Tracker has the following settings:

9.4.1 Tamper Detection

Learn more about [Tamper Detection](#).

9.4.2 Loss Of Signal Emit Interval

See [Loss Of Signal Emit Interval](#)

9.5 Deep Learning Object Tracker

The Deep Learning Object Tracker is designed for accurate detection and tracking of people, vehicles and key objects in challenging environments where motion based tracking methods would struggle. The list of objects detected by the Deep Learning Object Tracker is given below:

Class Name	Description
person	A person, or tracked object with a person present (e.g bicycle)
cyclist	Person riding a bicycle, can be reported as two separate objects
motorcycle	A motorcycle
bicycle	A bicycle
car	A car
van	A van, including mini-vans, mini-buses and buses
truck	A truck, including lorries and commercial work vehicles,
bag	A backpack or holdall

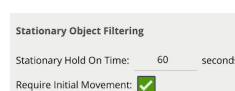
The Deep Learning Object Tracker is based on a classification and detection model, providing the location of an object in the field of view. See [Deep Learning Requirements](#) for hardware requirements for this algorithm.

The Deep Learning Object Tracker has the following settings:

9.5.1 Stationary Object Filtering

See [Stationary Hold On Time](#)

In addition to the Stationary Hold On Time, an additional setting Require Initial Movement, is available which will prevent objects which have not moved from being tracked.



9.5.2 Detection Point of Tracked Objects

See [Detection Point of Tracked Objects](#)

9.5.3 Tamper Detection

Learn more about [Tamper Detection](#).

9.5.4 Loss Of Signal Emit Interval

See [Loss Of Signal Emit Interval](#)

Chapter 10


Zones

Zones are the detection areas on which VCAserver **rules** operate. In order to detect a specific behaviour, a zone must be configured to specify the area where a rule applies.



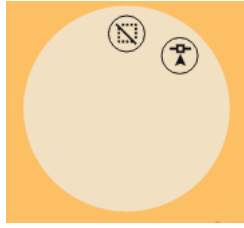
10.1 Adding a Zone

Zones can be added in multiple ways:





- Double-click anywhere on the video display.
- Click the **Create Zone** button in the zone settings menu.
- Right-click or tap-hold to display the context menu and select the add zone icon 

10.2 The Context Menu


Right-clicking or tap-holding (on mobile devices) displays a context menu which contains commands specific to the current context.



The possible actions from the context menu are:

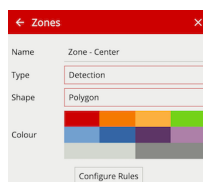
-  Adds a new zone.
-  Deletes an existing zone.
-  Adds a node to a zone.
-  Deletes an existing node from a zone.

10.3 Positioning Zones

To change the position of a zone, click and drag the zone to a new position. To change the shape of a zone, drag the nodes to create the required shape. New nodes can be added by double-clicking on the edge of the zone or clicking the add node icon  from the context menu.

10.4 Zone Specific Settings


The zone configuration menu contains a range of zone-specific configuration parameters:



- **Name:** The name of the zone, which appears in event notifications.
- **Type:** The type of the zone. Can be one of:
 - **Detection:** A zone which detects tracked objects and to which rules can be applied.
 - **Non-detection:** A zone which specifies an area that should be excluded from VCAserver analysis. Objects are not detected in non-detection zones. Useful for excluding areas of potential nuisance alarms from a scene (e.g. waving trees, flashing lights, etc).
- **Shape:** The shape of the zone. Can be one of:
 - **Polygon:** A polygonal detection area with at least three nodes. Rules apply to the whole area.
 - **Line:** A single- or multi-segment line with at least two nodes. Rules apply to the length of the line.
- **Colour:** The colour of the zone.
- **Configure Rules:** A shortcut button to navigate directly to the **rules** configuration page

10.5 Deleting a Zone

Zones can be deleted in the following ways:

- Select the zone and click the **Delete Zone** button from the zone settings menu.
- Select the zone, display the context menu and select the delete zone icon 

Chapter 11

Rules

VCAserver's rules are used to detect specific events in a video stream. There are three rule types which can be utilised to detect events and trigger actions:

- **Basic Inputs / Rule:** An algorithm that will trigger when a particular behaviour or event has been observed. E.g. Presence. Basic inputs can be used to trigger an action.
- **Filters:** A filter will trigger if the object which has triggered the input rule meets the filter requirements. E.g. is moving at a specific speed. Filters can be used to trigger an action.
- **Conditional Rule:** A logical link between one or more inputs to allow the detection of more complex behaviours. E.g. AND. Conditional rules can be used to trigger an action.

Within VCAserver, rule configurations can be as simple as individual **basic inputs** attached to a zone used to trigger an action. Alternatively rules can be combined into more complex logical rule configurations using **conditional rules** and **filters**. The overarching goal of the rules in VCAserver is to help eliminate erroneous alerts being generated, by providing functions to prevent unwanted behaviour from triggering an action.

More detail on the differences between these concepts is outlined below:

11.1 Basic Inputs

A basic input or rule can *only* be used to trigger an action or as an input to another rule type. Basic inputs always require a zone, and potentially some additional parameters. A basic input can be used on its own to trigger an action, although they are often used as an input to other filters or conditional rules.

The complete list of basic inputs are:

- Abandoned
- Appear
- Deep Learning Presence
- Direction
- Disappear
- Dwell
- Enter

- Exit
- Presence
- Stopped
- Tailgating
- Counting Line

Due to the nature of the deep learning algorithm which powers the Deep Learning Presence rule, it can not be used as an input to another filter or logical rule.

11.2 Filters

A filter cannot trigger an action on its own as it **requires** another basic input, filter or conditional rule to trigger. An example of this is the Object rule.

The complete list of filters are:

- Speed Filter
- Object Filter
- Colour Filter
- Deep Learning Filter
- Other Source Filter

Due to the nature of the deep learning algorithm which powers the Deep Learning Filter, it can not be used as an input to another filter or logical rule.

11.3 Conditional Rules

A conditional input, like a filter, is one that cannot trigger an action on its own. It **requires** the input of another basic input, conditional rule or filter to be meaningful. An example of this is the AND rule. The AND rule requires two inputs to compare in order to function.

The complete list of conditional rules are:

- And
- Continuously
- Counter
- Not
- Or
- Previous

11.4 General Concepts

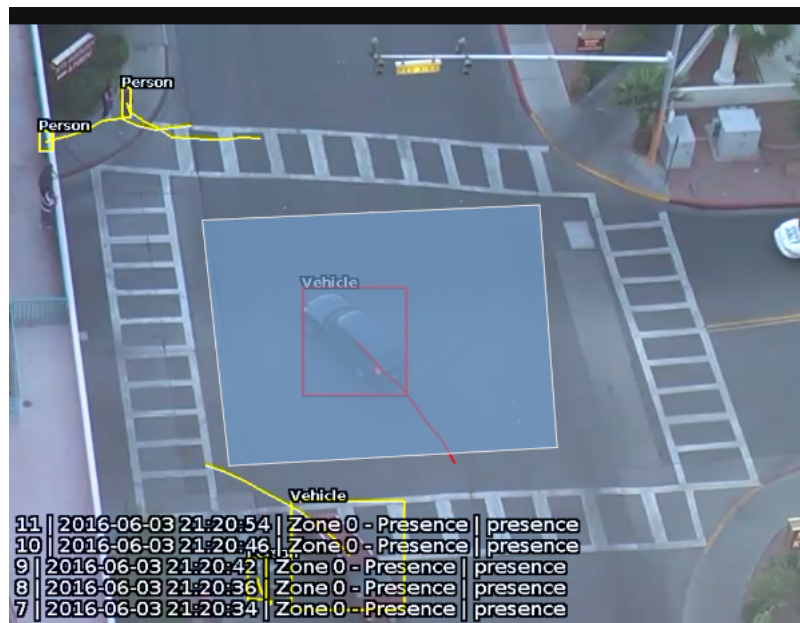
11.4.1 Object Display

As rules are configured they are applied to the channel in real time allowing feedback on how they work. Objects which have triggered a rule are annotated with a **bounding box** and a **trail**. Objects can

be rendered in two states:

- **Non-alarmed:** Default rendered in yellow. A detected object which does not meet any criteria to trigger a rule and raise an event.
- **Alarmed:** Default rendered in red. A detected object which has triggered one or more rules. Causes an event to be raised.

As seen below, when an event is raised, the default settings render details of the event in the lower half of the video stream. Object class annotations in this example are generated through **calibrated classification**

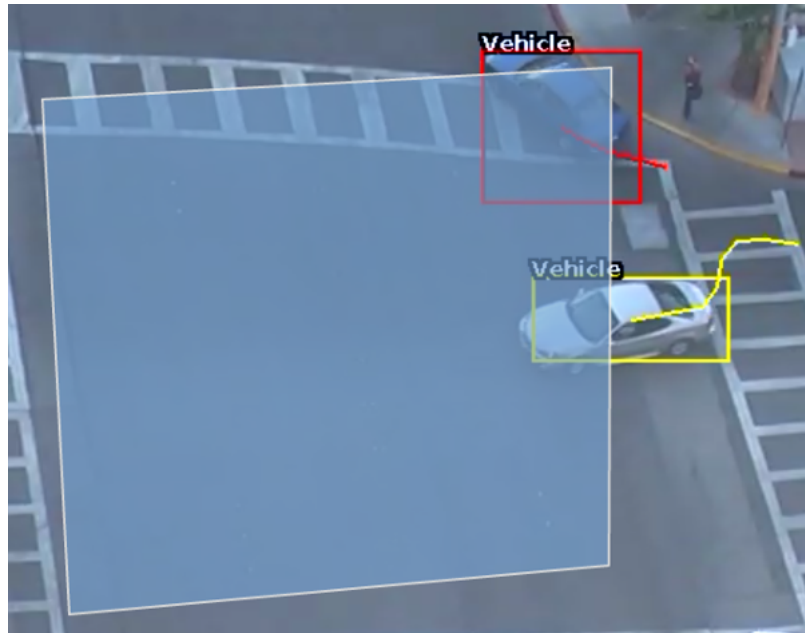


11.4.2 Object Trails

The **trail** shows the history of where the object has been. Depending on the **calibration** the trail can be drawn from the **centroid** or the **mid-bottom** point of the object. (See **Advanced Settings** for more information).

11.4.3 Trail Importance

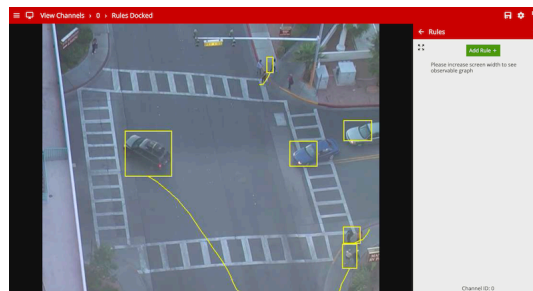
The trail is important for determining how a rule is triggered. The intersection of the trail point with a zone or line determines whether a rule is triggered or not. The following image illustrates this point: the blue vehicle's trail intersects with the detection zone and is rendered in red. Conversely, while the white vehicle intersects the detection zone, its trail does not (yet) intersect and hence it has not triggered the rule and is rendered in yellow.



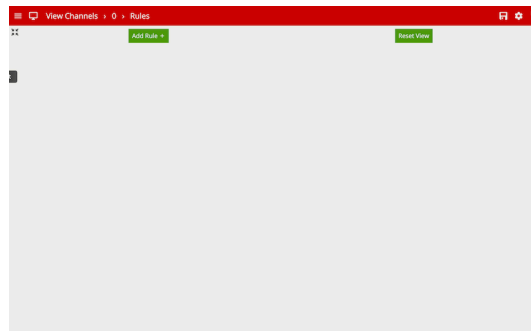
11.5 Rules Configuration

Rules are configured on a per channel basis by opening the rules menu when viewing the channel. Configuration is possible in two forms: the docked mode, in which both the rules and the video stream are visible or expanded view, in which a graph representation is provided to visualise the way the rules are connected.

The rules page opens in the 'docked' mode, alongside the live video stream.




The user may click on the expand button to switch to the expanded view. Please note that the rules graph is only visible in the expanded view.



In the expanded view, the user can add rules, and use the Rules Editor to connect the rules to one another. The graph on the right hand side updates in real time to reflect the user's changes.

11.5.1 Adding Rules

The first steps to defining a rule configuration is to add the basic inputs, configure the respective parameters and link to a zone. Click the **Add a Rule +** button and select the desired rule from the drop menu.

To delete a rule click the corresponding delete icon . Please note rules of any type cannot be deleted if they serve as an input to another rule. In this case the other rule must be deleted first.

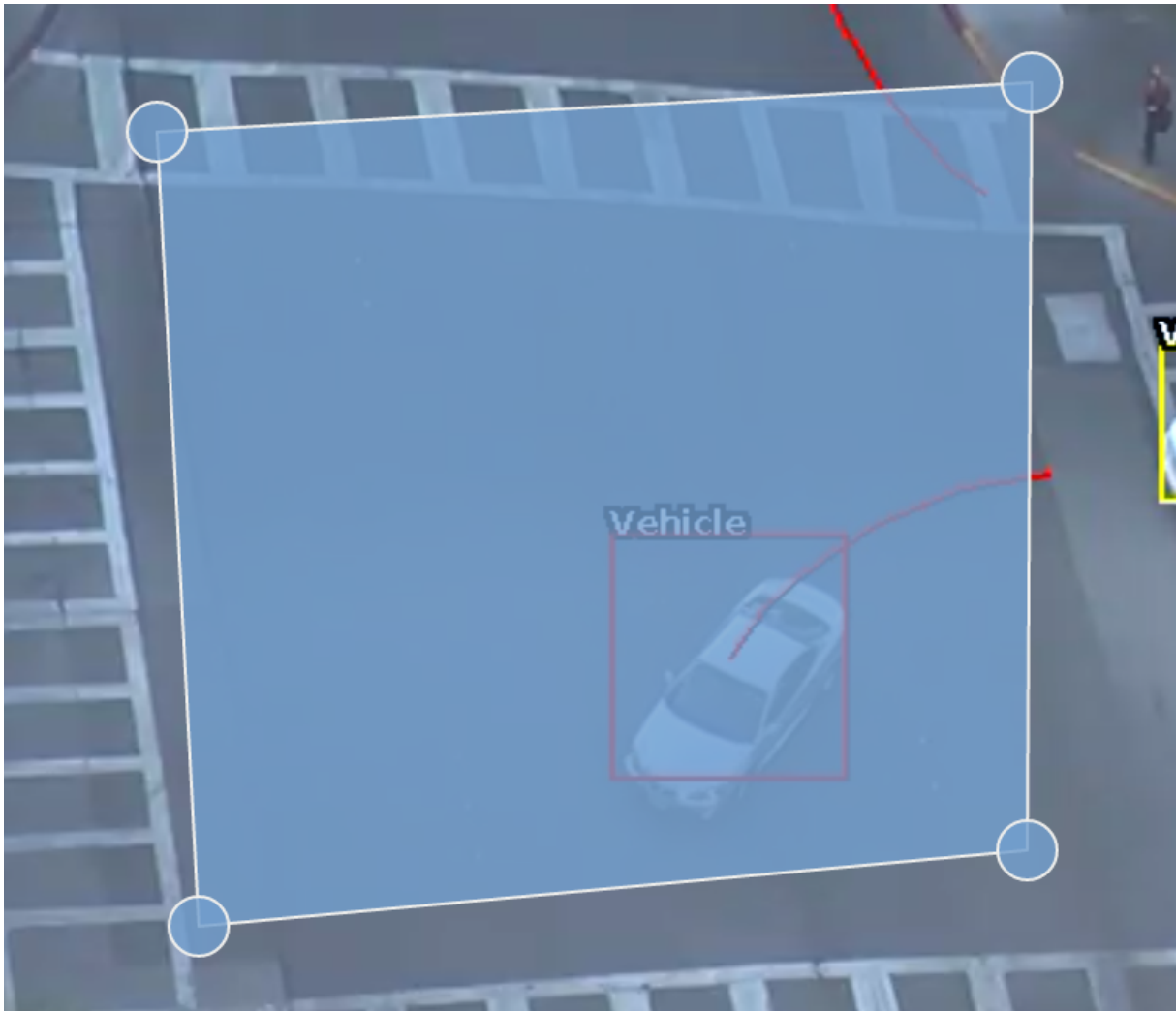
11.6 Basic Inputs

Below are the currently supported basic inputs, along with a detailed description of each.

11.6.1 Presence

A rule which fires an event when an object is first detected in a particular zone.


Note: The Presence rule encapsulates a variety of different behaviour, for example the Presence rule will trigger in the same circumstances as an Enter and Appear rule. The choice of which rule is most appropriate will be dependant on the scenario.



11.6.1.1 Graph View

Type: Presence
Name: Presence 5
Zone: None

11.6.1.2 Form View

Type:	Presence	
Name:	Presence 5	▼ 
Can Trigger Actions	<input checked="" type="checkbox"/>	
Zone:	None	▼

11.6.1.3 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	"Presence #"
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active
Zone	The zone this rule is associated with	None

11.6.2 Deep Learning Presence

A rule which fires an event when an object is detected in a particular zone, has been inside that zone for a defined `interval` (can be 0ms), and is classified as a certain class by the deep learning filter model. Classification will only occur after the object has been in the selected zone for the defined `Interval`.

Classification settings are configured in the Deep Learning page. See [Deep Learning Filter](#) for an in depth description on how the filter works.

Note: The Deep Learning Presence rule encapsulates a variety of different behaviour, for example the rule will try to trigger in the same circumstances as an Enter and Appear rule if `interval` is set to 0. If `interval` is greater than 0, the rule will trigger in the same circumstances as a Dwell rule. Additionally, the deep learning presence rule cannot be used as an input to any other rule type. As such it must work in isolation.

Note: The Deep Learning Presence rule will only work when the Object Tracker is selected under Trackers



11.6.2.1 Graph View



11.6.2.2 Form View

A configuration form for a rule. It has the following fields: "Type:" with a dropdown menu showing "Deep Learning Presence" and a red square icon; "Name:" with a text field containing "DL Presence"; "Can Trigger Actions:" with a checked checkbox; "Zone:" with a dropdown menu showing "Center"; and "Interval:" with a text field containing "1" and the unit "seconds".

11.6.2.3 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	"Deep Learning Presence #"
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active
Zone	The zone this rule is associated with	None
Interval	Period of time (in seconds)	1

11.6.3 Direction

The direction rule detects objects moving in a specific direction. Configure the direction and acceptance angle by moving the arrows on the direction control widget. The primary direction is indicated by the large central arrow. The acceptance angle is the angle between the two smaller arrows.

Objects that travel in the configured direction (within the limits of the acceptance angle), through a zone or over a line, trigger the rule and raise an event.

The following image illustrates how the white car, moving in the configured direction, triggers the rule whereas the other objects do not.



11.6.3.1 Graph View

Type: Direction
 Name: Direction 3
 Zone: Centre
 Angle: 358
 Angle Threshold: 27
 Can trigger actions: true

11.6.3.2 Form View

The screenshot shows a configuration window for a 'Direction' rule. It includes fields for 'Type' (Direction), 'Name' (Direction), 'Can Trigger Actions' (checked), 'Zone' (Center), 'Angle' (0 Degrees), and 'Acceptance' (90 Degrees). The 'Angle' and 'Acceptance' fields are represented by sliders with red circular markers.

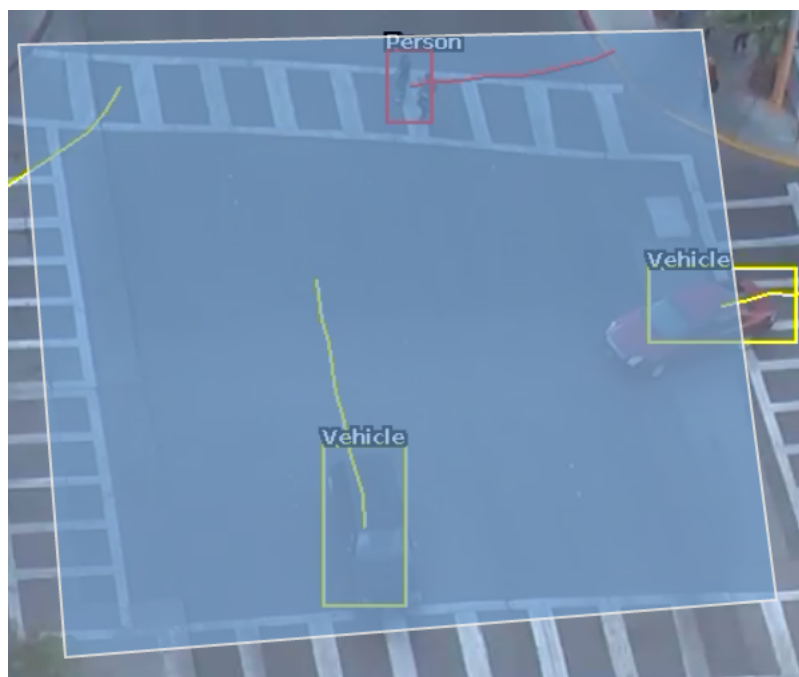
11.6.3.3 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	"Direction #"
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active
Zone	The zone this rule is associated with	None
Angle	Primary direction angle, 0 - 359. 0 references up.	0
Acceptance	Allowed variance each side of primary direction that will still trigger rule	0

11.6.4 Dwell

A dwell rule triggers when an object has remained in a zone for a specified amount of time. The interval parameter the time the object has to remain in the zone before an event is triggered.

The following image illustrates how the person, detected in the zone, is highlighted red as they have dwelt in the zone for the desired period of time. The two vehicles have not been present in the zone for long enough yet to trigger the dwell rule.



11.6.4.1 Graph View

Type: Dwell
Name: Dwell 1
Zone: Centre
Interval: 1000
Can trigger actions: true

11.6.4.2 Form View

Type: Dwell
Name: Dwell 1 ▼ ■
Can Trigger ☒
Actions
Zone: Centre ▼
Interval: 1 seconds

11.6.4.3 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	"Direction #"
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active
Zone	The zone this rule is associated with	None
Interval	Period of time in seconds)	1

11.6.5 Stopped


The stopped rule detects objects which are stationary inside a zone for longer than the specified amount of time. The stopped rule requires a zone to be selected before being able to configure an amount of time.

Note: The stopped rule does not detect abandoned objects. It only detects objects which have moved at some point and then become stationary.

11.6.5.1 Graph View

Type: Stopped
Name: Stopped 3
Zone: Centre
Time: 1
Can trigger actions: true

11.6.5.2 Form View

Type:	Stopped	
Name:	Stopped 4	▼ 
Zone:	Centre ▼	
Time	1 ▴ ▾ Secs	
Can Trigger Actions	<input checked="" type="checkbox"/>	

11.6.5.3 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	"Stopped #"
Zone	The zone this rule is associated with	None
Time	Period of time before a stopped object triggers the rule	0
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active

11.6.6 Enter and Exit

The enter rule detects when objects enter a zone. In other words, when objects cross from the outside of a zone to the inside of a zone.

Conversely, the exit rule detects when an object leaves a zone: when it crosses the border of a zone from the inside to the outside.

Note: Enter and exit rules differ from appear and disappear rules, as follows:

- Whereas the enter rule detects already-tracked objects crossing the zone border from outside to inside, the appear rule detects objects which start being tracked within a zone (e.g. appear in the scene through a door).
- Whereas the exit rule detects already-tracked objects crossing the zone border from inside to outside, the disappear rule detects objects which stop being tracked within the zone (e.g. leave the scene through a door).

11.6.6.1 Graph View



11.6.6.2 Form View

The form view displays two rule configuration sections. The first section is for an 'Enter' rule named 'Enter 3'. It has a 'Can Trigger Actions' checkbox checked with a green checkmark, and a 'Zone' dropdown menu set to 'Centre'. The second section is for an 'Exit' rule named 'Exit 4'. It also has a 'Can Trigger Actions' checkbox checked with a green checkmark, and a 'Zone' dropdown menu set to 'Centre'. Each section has a red trash icon in the top right corner.

11.6.6.3 Configuration Enter

Property	Description	Default Value
Name	A user-specified name for this rule	"Enter #"
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active

Property	Description	Default Value
Zone	The zone this rule is associated with	None

11.6.6.4 Configuration Exit

Property	Description	Default Value
Name	A user-specified name for this rule	"Exit #"
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active
Zone	The zone this rule is associated with	None

11.6.7 Appear and Disappear

The appear rule detects objects that start being tracked within a zone, e.g. a person who appears in the scene from a doorway.

Conversely, the disappear rule detects objects that stop being tracked within a zone, e.g. a person who exits the scene through a doorway.

Note: The appear and disappear rules differ from the enter and exit rules as detailed in the enter and exit rule descriptions.

11.6.7.1 Graph View

Type: Appear Name: Appear 3 Zone: Centre Can trigger actions: true	Type: Disappear Name: Disappear 4 Zone: Centre Can trigger actions: true
---	---

11.6.7.2 Form View

The screenshot shows a configuration interface for two types of rules: 'Appear' and 'Disappear'. Each rule configuration includes a 'Type' dropdown (set to 'Appear' or 'Disappear'), a 'Name' text field (e.g., 'Appear 3', 'Disappear 4'), a 'Can Trigger Actions' checkbox (checked with a green checkmark), and a 'Zone' dropdown menu (set to 'Centre'). A red square icon with a white 'X' is visible next to each rule configuration.

11.6.7.3 Configuration Appear

Property	Description	Default Value
Name	A user-specified name for this rule	"Appear #"
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active
Zone	The zone this rule is associated with	None

11.6.7.4 Configuration Disappear

Property	Description	Default Value
Name	A user-specified name for this rule	"Disappear #"
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active
Zone	The zone this rule is associated with	None

11.6.8 Abandoned and Removed Object

The abandoned and removed object rule triggers when an object has been either left within a defined zone. E.g. a person leaving a bag on a train platform, or when an object is removed from a defined zone. The abandoned rule has a duration property which defines the amount of time an object must have been abandoned for, or removed for, to trigger the rule.

Below is a sample scenario where a bag is left in a defined zone resulting in the rule triggering.



Below is a similar example scenario where the bag is removed from the defined zone resulting in the rule triggering.





Note: The algorithm used for abandoned and removed object detection is the same in each case, and therefore cannot differentiate between objects which have been abandoned or removed. This is because the algorithm only analyses how blocks of pixels change over time with respect to a background model. **Note:** The algorithm used for abandoned and removed object will only work when the Object Tracker is selected under Trackers

11.6.8.1 Graph View

Type: Abandoned
Name: Abandoned 3
Zone: Centre
Duration: 2
Can trigger actions: true

11.6.8.2 Form View

Type:	Abandoned	▼	
Name:	Abandoned 3		
Zone:	Centre ▼		
Duration	2	Secs	
Can Trigger Actions			

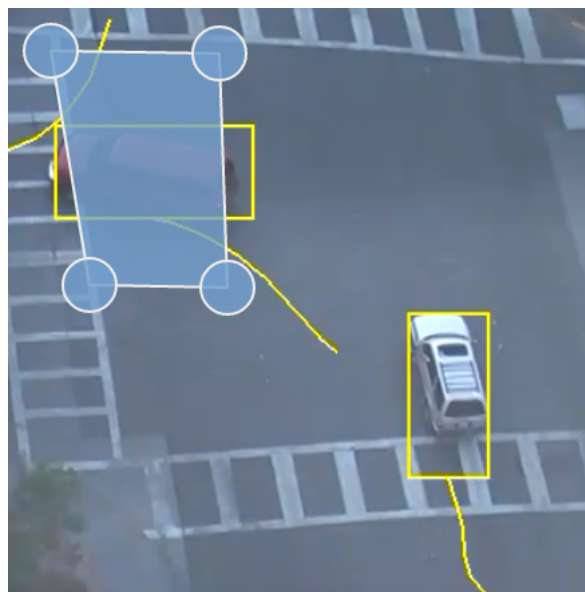
11.6.8.3 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	"Abandoned #"
Zone	The zone this rule is associated with	None
Duration	Period of time a object must have been abandoned or removed before the rule triggers	0
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active

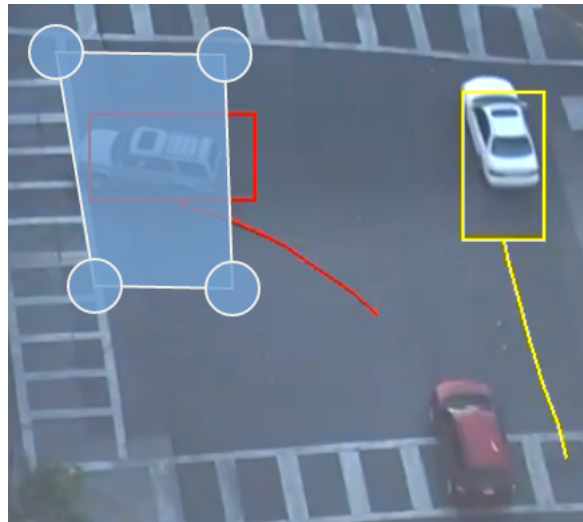
11.6.9 Tailgating

The tailgating rule detects objects which cross through a zone or over a line within quick succession of each other.

In this example, object 1 is about to cross a detection line. Another object (object 2) is following closely behind. The tailgating detection threshold is set to 5 seconds. That is, any object crossing the line within 5 seconds of an object having already crossed the line will trigger the object tailgating rule.



Object 2 crosses the line within 5 seconds of object 1. This triggers the tailgating filter and raises an event.



11.6.9.1 Graph View

Type: Tailgating
 Name: Tailgating 3
 Zone: Centre
 Duration: 5
 Can trigger actions: true

11.6.9.2 Form View

Type: Tailgating

Name: Tailgating 3

Zone: Centre

Duration: 5 Secs

Can Trigger Actions: ☒

11.6.9.3 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	"Tailgating #"
Zone	The zone this rule is associated with	None
Duration	Maximum amount of time between first and second object entering a zone to trigger the rule	0

Property	Description	Default Value
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active

11.6.10 Counting Line

A counting line is a detection filter optimized for directional object counting (e.g. people or vehicles) in busier detection scenarios. Examples of such applications may include:

- People counting with overhead cameras in a retail environment.
- Vehicle counting with overhead cameras on public highways.

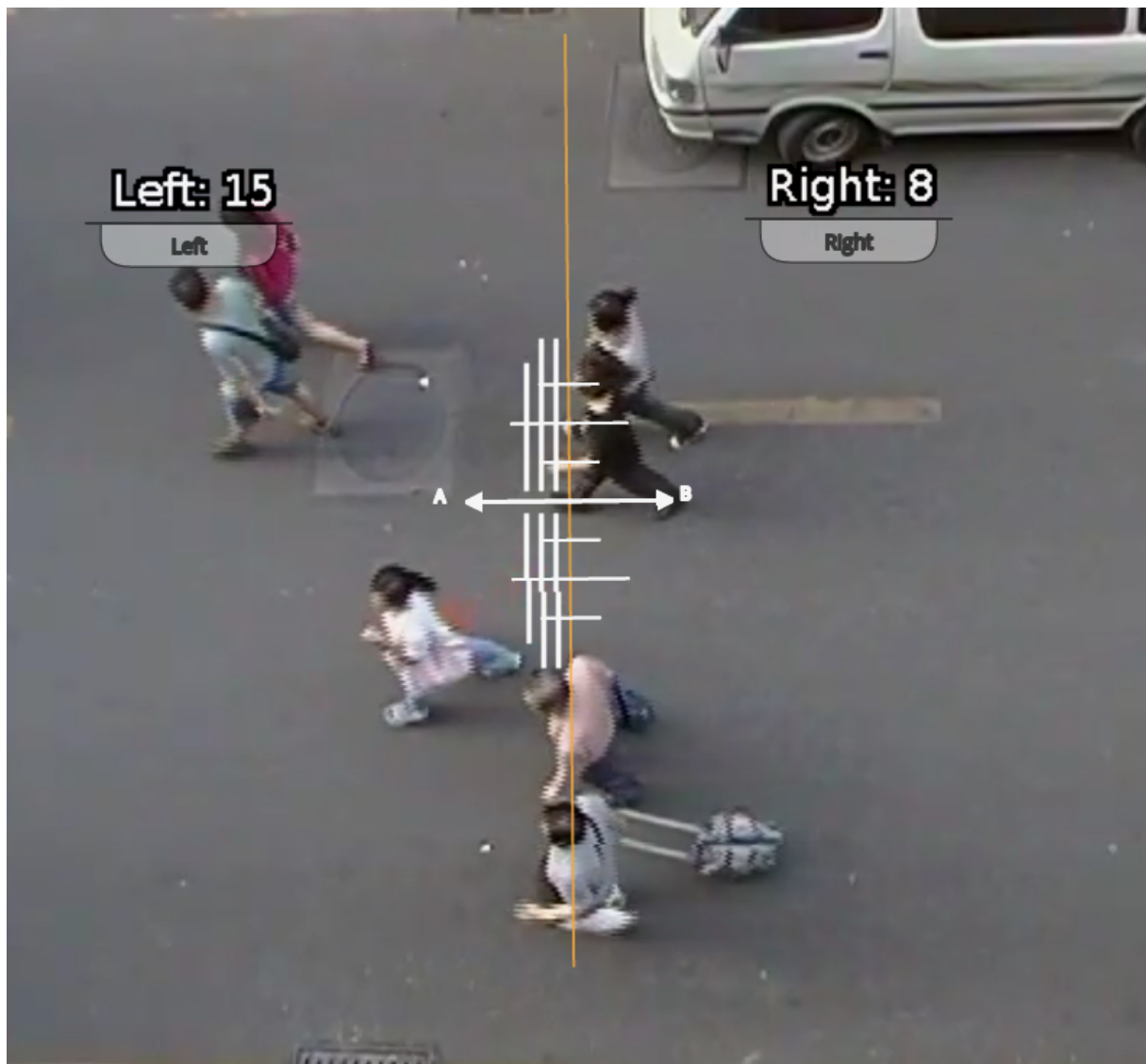
In some scenes, such as entrances with camera installed overhead, the counting line typically will generate a higher accuracy count than using the aforementioned counters connected to a presence rule.

An event is generated every time an object crosses the line in the configured direction. If multiple objects cross the line together, multiple corresponding events are generated. These events can be directly used to trigger actions if the **Can Trigger Actions** property is checked.

Counting lines are attached to zones configured with a **Line** shape. See [Zones](#) for more information. If a counting line is configured with a zone not defined with a **Line** shape, the zone property will be automatically changed (it will not be possible to change the zone shape back until the counting line stops referencing the zone in question).

Counting lines have a specified direction indicated by the arrow in the UI (direction **A** or **B**), the direction of this arrow is governed by the configured zone. Each instance of the rule counts in a single direction. To count in both directions a second counting line rule must be added to the same zone with the opposite direction selected. An example rule graph of a two way counting line configured with a counter is provided to illustrate this below.

NOTE: The maximum number of counting line filters that can be applied per video channel is 5.



11.6.10.1 Calibrating the Counting Line

In order to generate accurate counts, the counting line requires calibration. Unlike the object tracking function engine, this cannot be performed at a general level for the whole scene using the 3D **Calibration** tool. This is because the counting line is not always placed on the ground plane; it may be placed at any orientation at any location in the scene. For example, a counting line could be configured vertically with a side-on camera view.

Instead of the 3D calibration tool, the counting line has its own calibration setting. Two bars equidistant from the centre of the line represent the width of the expected object. This allows the counting line to reject noise and also count multiple objects.



To calibrate the counting line:

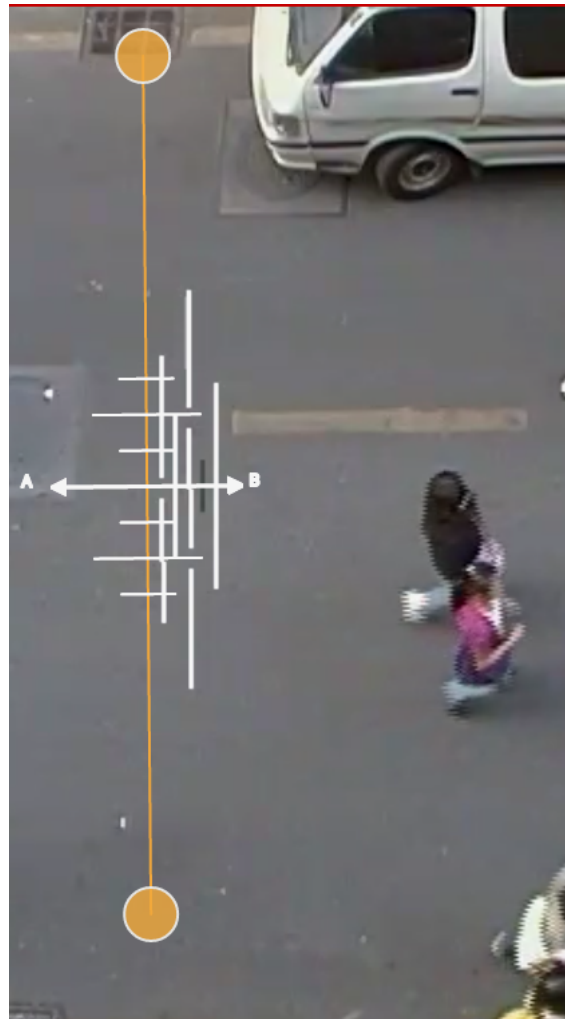
- Select the counting line rule.
- Check the **Enable width calibration** option.
- Drag a calibration marker to adjust the distance between the calibration markers, until the distance is approximately the size of the objects to be counted. Alternatively, move the **Width** slider to achieve the same result.
- The calibration width is displayed within the counting line rule, this value can be edited directly to change the calibration width.
- The smaller markers on either side of the large markers indicate the minimum and maximum width of a single object.

NOTE: if the **Width** slider is set to zero then the **Enable width calibration** checkbox is automatically disabled.

11.6.10.2 Counting Line Calibration Feedback

To enable the user to more accurately configure the calibration for the counting line, the widths of detected objects are displayed as an overlay next to the counting line when objects pass over it. By default this display option is enabled. However, if it does not appear, ensure that the option is enabled on the **Burnt-in Annotation** settings.

The calibration feedback is rendered as lines on either side of the counting line on the **Zones** configurations page. Each line represents an object detected by the counting algorithm. The width of the line shows the width of the object detected by the line. The last few detections are displayed for each direction with the latest one appearing closest to the counting line.



Each detection is counted as a number of objects based on the current width calibration. This is displayed as follows:

- Black line: Event not counted
- Solid white line: Event counted as one object
- Broken white line: Event counted as multiple objects indicated by the number of line segments.

Using the feedback from the calibration feedback annotation, the width calibration can be fine tuned to count the correct sized objects and filter out spurious detections.

11.6.10.2.1 Shadow Filter The counting line features a shadow filter which is designed to remove the effects of object shadows affecting the counting algorithm. Shadows can cause inaccurate counting results by making an object appear larger than its true size or by joining two or more objects together. If shadows are causing inaccurate counting, the shadow filter should be enabled by selecting the **Shadow Filter** check box for the line. It is recommended that the shadow filter only be enabled when shadows are present because the algorithm can mistake certain parts of an object for shadows and this may lead to worse counting results. This is especially the case for objects that have little contrast compared to the background (e.g. people wearing black coats against a black carpet).

11.6.10.3 Graph View

Type: Line Counter

Name: Line_Counter

Can trigger actions: true

11.6.10.4 Form View

Type: Line Counter

Name: Line_Counter

Zone: Centre Line

Direction: Direction A

Enable width calibration ☒

Width: 0.146

Can Trigger Actions ☒

11.6.10.5 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	Line_Counter #
Zone	The zone this rule is associated with	None
Direction	Enable counting in the 'A' or 'B' direction (one direction per counting line)	None
Enable Width Calibration	Width calibration to allow more accurate counting	None
Width	Width calibration value	0
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active

11.6.10.6 Typical Logical Rule Combination

The below example has two line counters, ****Line_Counter A**** and ****Line_Counter B**** attached to the zone ****Center Line**** each with differing directions selected. ****Line_Counter A**** is configured to increment the counter, whilst ****Line_Counter B**** is configured to decrement the counter value.

Only the counter rule ****Counter**** is set to ****Can Trigger Actions****, meaning only this com-

ponent of the logical rule will be available as a source for actions. In this case an action using this rule as a source will trigger every time the counter changes.

The image shows a configuration interface for three components: Line Counter A, Line Counter B, and a Counter.

Line Counter A Configuration:

- Type: Line Counter
- Name: Line_Counter A
- Zone: Centre Line
- Direction: Direction A
- Enable width calibration: ☒
- Width: 0.146
- Can Trigger Actions: ☐

Line Counter B Configuration:

- Type: Line Counter
- Name: Line_Counter B
- Zone: Centre Line
- Direction: Direction B
- Enable width calibration: ☒
- Width: 0.237
- Can Trigger Actions: ☐

Counter Configuration:

- Type: Counter
- Name: Counter
- Can Trigger Actions: ☒
- Increment: Line_Counter A
- Decrement: Line_Counter B
- Occupancy: (Add Occupancy Input +)
- Reset Counter: (Reset Counter)

Diagram:

```

graph TD
    A["Type: Line Counter  
Name: Line_Counter A  
Can trigger actions: false"] --> C["Type: Counter  
Name: Counter  
Increment Inputs: Line_Counter A  
Decrement Inputs: Line_Counter B  
Occupancy Inputs:  
Can trigger actions: true"]
    B["Type: Line Counter  
Name: Line_Counter B  
Can trigger actions: false"] --> C
  
```

11.7 Filters

Below is a list of the currently supported filters, along with a detailed description of each.

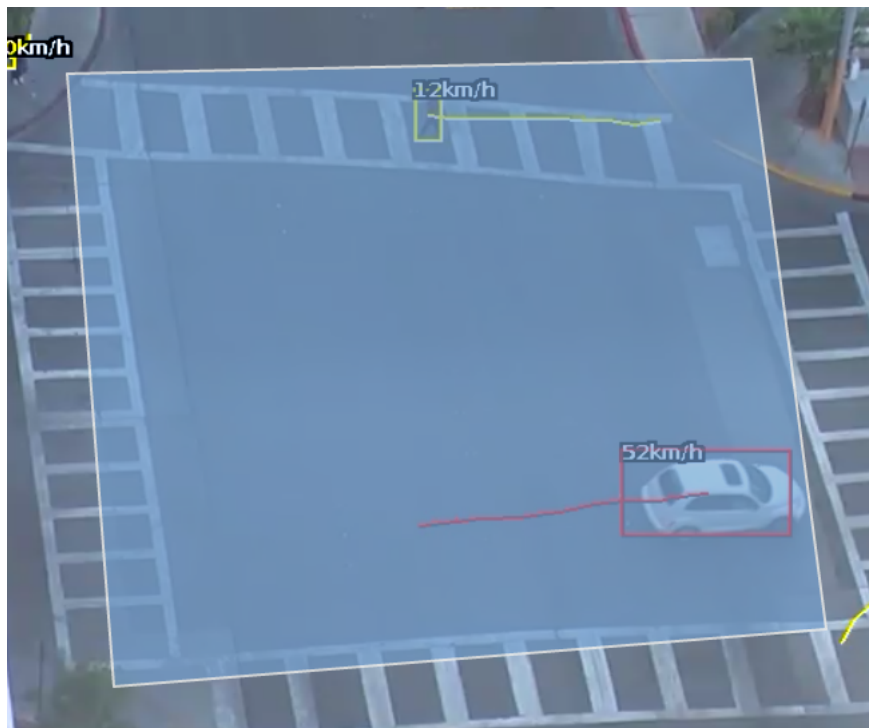
When filters are used to trigger an action the rule type property is propagated from the filter input. For example, if the input to the speed filter is a *presence* rule, then actions generated as a result of the speed filter will have a *presence* event type.

11.7.1 Speed Filter

The speed filter provides a way to check if the speed of an object, which has triggered an input, is moving within the range of speeds defined by a lower and upper boundary.

Note: The channel must be **calibrated** in order for the speed filter to be available.

Commonly, this rule is combined with a presence rule, an example rule graph is provided to illustrate this below. The following image illustrates how such a rule combination triggers on the car moving at **52 km/h**, but the person moving at **12 km/h** falls outside the configured range (25-100 km/h) and thus does not trigger the rule.



11.7.1.1 Graph View

Type: Speed
Name: Speed 3
Min Speed: 50
Max Speed: 200
Can trigger actions: true

11.7.1.2 Form View

Type: Speed

Name: Speed Filter 25-100kmph

Can Trigger Actions ☒

Input: Presence Rule

Min Speed: 25 kmph

Max Speed: 100 kmph

11.7.1.3 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	"Speed #"
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active
Input	The input rule	None
Min Speed	The minimum speed (km/h) an object must be going to trigger the rule	0
Max Speed	The maximum speed (km/h) an object can be going to trigger the rule	0

11.7.1.4 Typical Logical Rule Combination

The logical rule example below checks if an object triggering the presence rule **Presence Rule** attached to zone **Centre**, is also travelling between 25 and 100 km/h as specified by the speed rule **Speed Filter 25-100 km/h**.

Only the Speed Filter is set to **Can Trigger Actions**, meaning only this component of the logical rule will be available as a source for actions. Additionally, any action generated by the speed filter will have the event type Presence.

Type: Presence
Name: Presence Rule

Can Trigger Actions: ☐
Zone: Center

Type: Speed
Name: Speed Filter 25-100kmph

Can Trigger Actions: ☒
Input: Presence Rule

Min Speed: 25 kmph
Max Speed: 100 kmph

Type: Presence
Name: Presence Rule
Zone: Center
Can trigger actions: false

Type: Speed
Name: Speed Filter 25-1...
Min Speed: 25
Max Speed: 100
Can trigger actions: true

11.7.2 Object Filter

The object filter provides the ability to filter out objects which trigger a rule, if they are not classified as a certain class (e.g. person, vehicle). The available classes which can be used to filter, depend on which tracker is currently selected. In cases where the class is assigned via a deep learning model (DLF, DLOT, DLPT), the confidence threshold can also be used to further filter out objects which the model is not confident about its class.

The object classification filter must be combined with another rule(s) to prevent unwanted objects from triggering an alert, an example rule graph is provided to illustrate this below.



The previous image illustrates how the object classification filter configured with **Vehicle** class, only triggers on Vehicle objects. The person in the zone is filtered out since the **Person** class is not selected in the filter list.

Note: when using the Object Tracker, the channel must be **calibrated** for the object classification filter to be available.

11.7.2.1 Graph View

Type: Object Filter
Name: Vehicle Filter
Filters:
Vehicle
Can trigger actions: true

11.7.2.2 Form View

Type: Object Filter
Name: Vehicle Filter
Can Trigger Actions: ☒
Input: Presence
Classes: ☐ Person, ☐ Cyclist, ☒ Car, ☐ Van, ☐ Truck, ☐ Motorcycle, ☐ Bicycle, ☐ Bag
Confidence Threshold: 70 %

11.7.2.3 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	"Object Filter #"
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active
Input	The input rule	None
Classes	The object classes allowed to trigger an alert	None

11.7.2.4 Typical Logical Rule Combination

The logical rule example below checks if the object triggering the presence rule **Presence Rule** attached to zone **Centre**, is also classified as a **Vehicle** as specified by the Object Filter **Vehicle Filter**.

Only the Object filter is set to **Can Trigger Actions**, meaning only this component of the logical rule will be available as a source for actions. Additionally, any action generated by the speed filter will have the event type Presence.

Type: Presence
Name: Presence Rule
Can Trigger Actions: ☐
Zone: Center

Type: Object Filter
Name: Vehicle Filter
Can Trigger Actions: ☒
Input: Presence Rule
Classes:

☐ Person
☒ Vehicle
☐ Clutter
☐ Group Of People

Type: Presence
Name: Presence Rule
Zone: Center
Can trigger actions: false

Type: Object Filter
Name: Vehicle Filter
Filters: Vehicle
Can trigger actions: true

11.7.3 Colour Filter

The colour filter rule utilises the Colour Signature algorithm, providing the ability to filter out objects based on an object's colour components.

The Colour Signature algorithm groups the pixel colours of an object. When a Colour Filter rule is added to a channel, any object that is tracked by VCAserver will also have its pixels grouped into 10 colours. By default this information is added to VCAserver's metadata, available as tokens, via the SSE metadata service or that channel's RTSP metadata stream.

The colour filter allows you to select one or more of these colour bins, and will trigger only if the subject object contains one or more of those selected colours.

The below image shows an example tracked object with the colour signature annotations enabled. Here the top four colours which make up more than 5% of the object are represented by the colour swatch attached to the object. In this case a person being tracked in the scene with high visibility safety clothing. Here the colour filter is set to trigger on Yellow, detecting the person but ignoring the shadow.

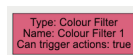
Typically, the colour filter would be combined with another rule(s) to prevent unwanted objects from triggering an alert, an example rule graph is provided to illustrate this below.



The previous image illustrates how the colour filter prevents objects, which do not contain the specified colours, from generating an event. In this case only the person generates an event but not the train line.

Note: the channel must have the **Colour Signature** enabled for the colour filter to work.

11.7.3.1 Graph View



11.7.3.2 Form View

Type: Colour Filter
Name: Colour 1 ✕ ▼
Can Trigger ☒
Actions
Input: Trainline
Colours:

☐

☐

☐

☐

☐

☐

☐

☐

☐

☒

☐

☐

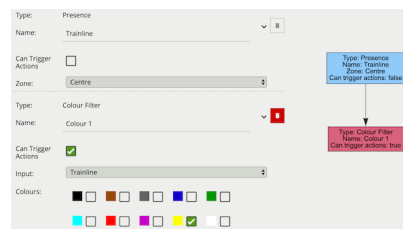
11.7.3.3 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	"Object Filter #"
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active
Input	The input rule	None
Colours	The colours allowed to trigger an alert	All Unchecked

11.7.3.4 Typical Logical Rule Combination

The logical rule example below checks if the object triggering the presence rule **Train line** attached to zone **Centre**, also contains the colour **Yellow** as one of the top four colours by percentage.

Only the Colour filter is set to **Can Trigger Actions**, meaning only this component of the logical rule will be available as a source for actions. Additionally, any action generated by the colour filter will have the event type Presence.

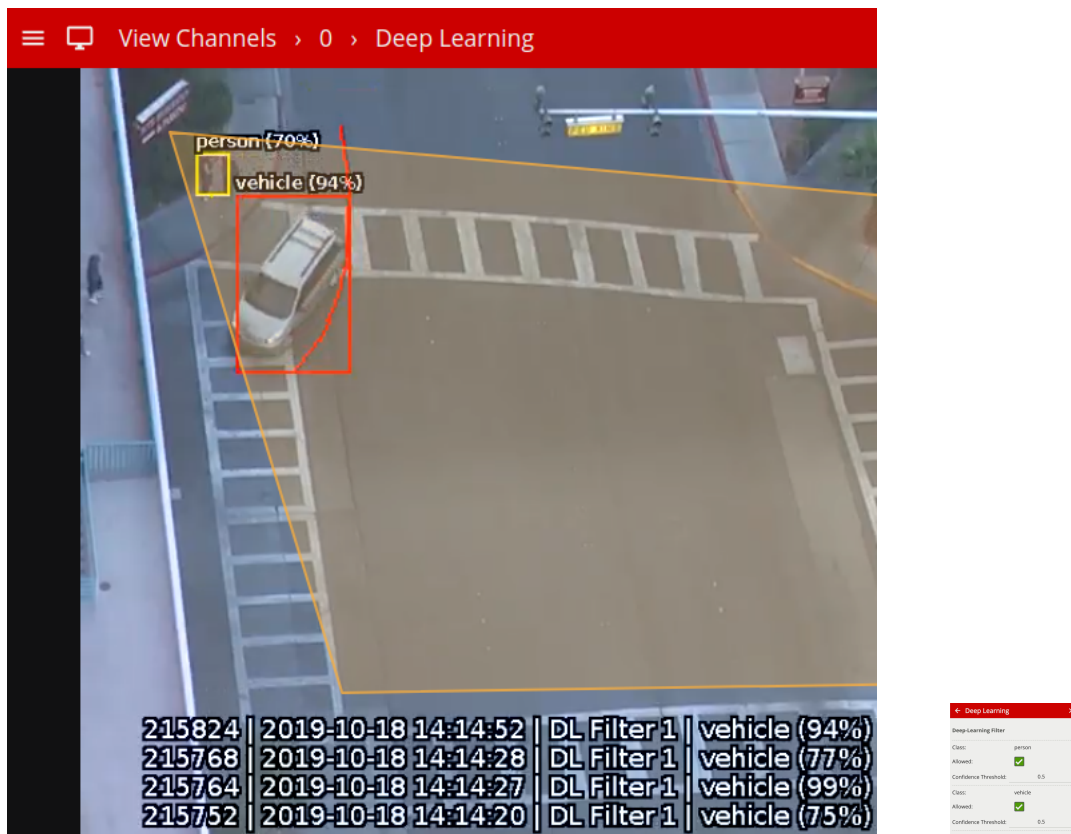


11.7.4 Deep Learning Filter

The deep learning filter provides the ability to filter out objects, which trigger a rule, if they are not classified as a certain class by the deep learning model.

The deep learning filter settings are configured in the Deep Learning page. See [Deep Learning Filter](#) for an in depth description on how the filter works.

Typically the deep learning filter would be combined with another rule(s), to prevent unwanted objects from triggering an alert, an example rule graph is provided to illustrate this below. Please note that the deep learning filter cannot be used as an input to any other rule type. As such it must be the last rule in a graph.



The previous image illustrates how the deep learning filter, configured with just **vehicle** class, only triggers on the vehicle object. The person in the zone is filtered out since the **person** class setting is not **enabled** in the Deep Learning settings.

Note: The Deep Learning Filter will only work when the Object Tracker is selected under Trackers

11.7.4.1 Graph View



11.7.4.2 Form View

Type: DeepLearningFilter

Name: DL Filter 1

Can Trigger Actions ☒

Input: None

11.7.4.3 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	"DL Filter #"
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active
Input	The input rule	None

11.7.4.4 Typical Logical Rule Combination

The logical rule example below checks if the object triggering the presence rule **Presence Rule**, attached to zone **Centre**, is one of the classes of interest defined in the Deep Learning settings page (see above settings page image).

Only the deep learning filter is set to **Can Trigger Actions**, meaning only this component of the logical rule will be available as a source for actions. Additionally, any action generated by the deep learning filter will have the event type Presence.

The screenshot displays the configuration for a logical rule combination. It consists of two main sections for configuring rules and a diagram on the right.

Top Section (Presence Rule):

- Type: Presence
- Name: Presence Rule
- Can Trigger Actions: ☐ (unchecked)
- Zone: Center

Bottom Section (DeepLearningFilter):

- Type: DeepLearningFilter
- Name: DL Filter 1
- Can Trigger Actions: ☒ (checked)
- Input: Presence Rule

Diagram:

The diagram shows a flow from the Presence rule to the DeepLearningFilter. The Presence rule box contains: Type: Presence, Name: Presence Rule, Zone: Center, Can trigger actions: false. An arrow points down to the Deep Learning Filter box, which contains: Type: Deep Learning Filter, Name: DL Filter 1, Can trigger actions: true.

11.7.5 Other Source Filter

The other source filter provides the ability to use **Other Sources** to filter an input rule in a rule graph. The other source filter will only trigger an event in cases when the selected other source evaluates as on, whilst the input rule triggers an event.

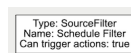


Valid **Other Sources** and the valid on scenario are outlined in the table below:

Other Source Type	on Condition	off Condition
HTTP	The observable state is set true	The observable state is set false
Schedule	The current system clock falls into a scheduled 'on' period	The current system clock falls into a scheduled 'off' period

Typically the other source filter would be used to limit a rule(s) from firing if an external requirement is not met. For example, using a **Schedule** source with the source filter only triggers events if the input rule fires during set periods of time. Alternatively, using a **HTTP** source would only trigger an event when the input rule triggers and the **HTTP** source state is set to true. An example rule graph is provided to illustrate this below.

11.7.5.1 Graph View



11.7.5.2 Form View

Type: SourceFilter
 Name: Schedule Filter
 Can Trigger Actions: ☒
 Input: Presence Centre
 Source: New Schedule - Sch

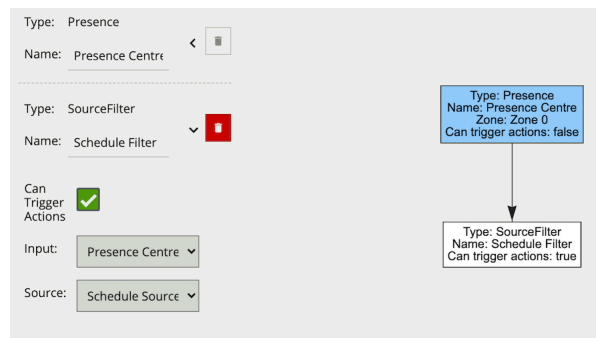
11.7.5.3 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	"Other Source #"
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active
Input	The input rule	None
Source	The other source	None

11.7.5.4 Typical Logical Rule Combination

The logical rule example below will only generate an event if the current system time falls within an on period, defined in the source **Schedule Source** and the input rule **Presence Centre** attached to zone **Zone 0**, triggers an event.

Only the other source filter is set to **Can Trigger Actions**, meaning only this component of the logical rule will be available as a source for actions. Additionally, any action generated by the other source filter will have the event type Presence.



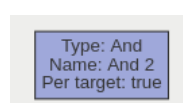
11.8 Conditional Rule Types

Below is a list of the currently supported conditional rules, along with a detailed description of each.





11.8.1 And

A logical operator that combines two rules and only fires events if both inputs are true.

11.8.1.1 Graph View



11.8.1.2 Form View

Type:	And	
Name:	And 2	 
Can Trigger Actions	<input checked="" type="checkbox"/>	
Input A:	None	
Input B:	None	
Per Target	<input checked="" type="checkbox"/>	

11.8.1.3 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	"And #"
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active
Input A	The first input	None
Input B	The second input	None
Per Target	Fire one event per tracked object	Active

If we consider a scene with two presence rules, connected to two separate zones, connected by an AND rule, the table below explains the behaviour of the **Per Target** property. Note that object here refers to a tracked object, as detected by the VCA tracking engine.

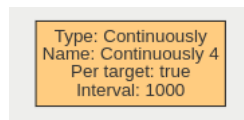
State	Per Target	Outcome
Object A in Input A, Object B in input B	On	Two events generated, one for each object
Object A in Input A, Object B in input B	Off	Only one event generated

Additionally, it is important to note that if the rule fires when **Per Target** is switched off, it will not fire again until it is 'reset', i.e. until the AND condition is no longer true.


11.8.2 Continuously

A logical operator that fires events when its input has occurred continuously for a user-specified time.

11.8.2.1 Graph View



11.8.2.2 Form View

Type:	Continuously	▼	
Name:	Continuously 4		
Can Trigger Actions	<input checked="" type="checkbox"/>		
Input:	None ▼		
Per Target	<input checked="" type="checkbox"/>		
Interval	1000	ms	

11.8.2.3 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	"Continuously #"
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active
Input	The input rule	None
Per Target	Fire one event per tracked object. See description below for more details	Active
Interval	The time in milliseconds	1000 ms

Considering a scene with a Presence rule associated with a zone and a Continuously rule attached to that Presence rule, when the **Per Target** property is on, the rule will generate an event for each tracked object that is continuously present in the zone. When it is off, only one event will be generated by the rule, even if there are multiple tracked objects within the zone. Additionally, when **Per Target** is off, the rule will only generate events when there is change of state, i.e. the rule condition changes from true to false or vice versa. When **Per Target** is off, the state will change when:

- Any number of objects enter the zone in question and remain in the zone
- **All** objects leave the zone in question

11.8.3 Counter

Counters can be configured to count the number of times a rule is triggered. For example, the number of people crossing a line. The counter rule is designed to be utilised in two ways:

- **Increment / Decrement:** whereby a counter is incremented by the attached rule(s) (+1 for each rule trigger), and decremented by another attached rule(s) (-1 for each rule trigger).
- **Occupancy:** whereby the counter reflects the number of objects that are currently triggering the attached rule(s).

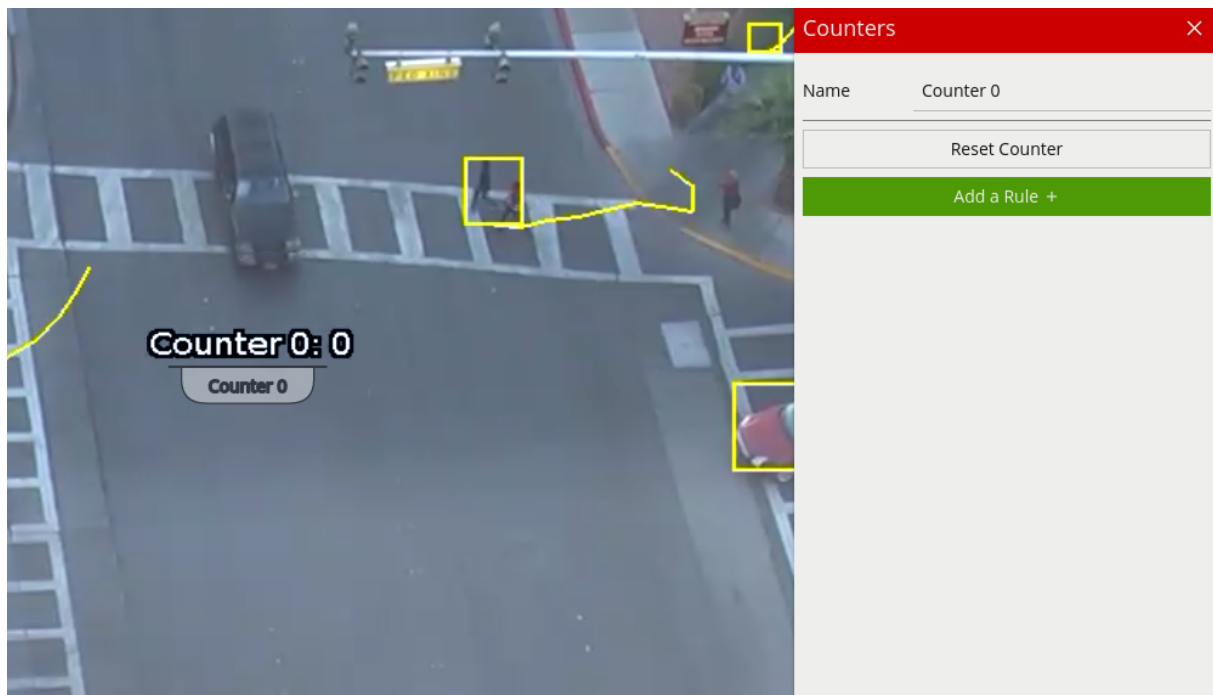
More than one rule can be assigned to any of a counter's three inputs. This allows, for example, the occupancy of two presence rules to be reflected in a single counter, or more than one entrance / exit gate to reflect in a single counter. An example rule graph is provided to illustrate this below.

Broadly speaking a single counter should not be used for both purposes occupancy and increment / decrement.

The Counter's Threshold Operator allows the user to limit when a counter generates an event. Based on the selected behaviour and a defined Threshold Value, the counter can be configured to only send events in specific scenarios. Threshold Operators include:

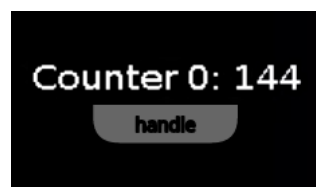
- Greater than or equal to
- Less than or equal to
- Greater than
- Less than
- Equal to
- Not Equal to
- None

Note: events created by a counter will not trigger the **Deep-Learning Filter**, even if enabled on the channel.

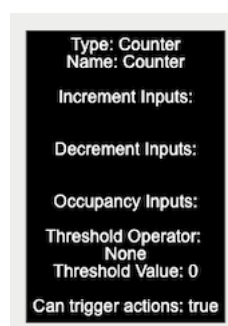


11.8.3.1 Positioning Counters

When added, a counter object is visualised on the video stream as seen below. The counter can be repositioned by grabbing the 'handle' beneath the counter name and moving the counter to the desired location.



11.8.3.2 Graph View



11.8.3.3 Form View

The screenshot shows a configuration form for a 'Counter' type. It includes fields for 'Type' (set to Counter), 'Name' (set to Counter), 'Can Trigger Actions' (checked), 'Increment' (with an 'Add Increment Input +' button), 'Decrement' (with an 'Add Decrement Input +' button), 'Occupancy' (with an 'Add Occupancy Input +' button), 'Threshold Operator' (set to None), 'Threshold Value' (set to 0), and a 'Reset Counter' button.

11.8.3.4 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	"Counter #"
Increment	The rule which, when triggered, will add one to the counter	None
Decrement	The rule which, when triggered, will subtract one from the counter	None
Occupancy	Sets counter to current number of the rule's active triggers	None
Threshold Operator	Defines when a Counter will trigger events based on the threshold	None
Threshold Value	The value used by the Threshold Operator to define the behaviour	0
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active
Reset Counter	A button allowing the counter value to be reset to 0	None

* E.g. if a Presence rule is set as the occupancy target and two objects are currently triggering that Presence rule, the counter will show the value of 2.

11.8.3.5 Typical Logical Rule Combination

The below counter example increments a counter based on two enter rules, **Enter Centre** and **Enter Top** attached to the zones **Centre** and **Top** respectively, this means that when either of these enter rules triggers the counter will be incremented by + 1. The counter also decrements based on the exit rule **Exit**, which will subtract 1 from the counter each time an object exits the zone **Centre**. The Threshold Operator and Threshold Value, limit the counter to only generate events when the count is more than 20.

Only the counter rule **Counter** is set to **Can Trigger Actions**, meaning only this component of the logical rule will be available as a source for actions. In this case an action using this rule as a source will trigger every time the counter changes.

The interface displays the configuration for a Counter rule and its associated Enter and Exit rules.

Counter Rule Configuration:

- Type: Counter
- Name: Counter
- Can Trigger Actions: ☒
- Increment: Enter Centre, Enter Top
- Decrement: Exit
- Occupancy: Add Occupancy Input +
- Threshold Operator: Greater than or equal to
- Threshold Value: 20
- Reset Counter

Enter Rules:

- Type: Enter, Name: Enter Centre
- Type: Enter, Name: Enter Top

Exit Rule:

- Type: Exit, Name: Exit

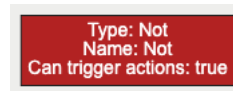
Diagram:

The diagram illustrates the logical flow of the Counter rule. It shows three input boxes on the left: 'Type: Enter, Name: Enter Top, Zone: None, Can trigger actions: true', 'Type: Enter, Name: Enter Centre, Zone: None, Can trigger actions: true', and 'Type: Exit, Name: Exit, Zone: None, Can trigger actions: true'. Arrows from the 'Enter Top' and 'Enter Centre' boxes point to a central box labeled 'Type: Counter, Name: Counter, Can trigger actions: true'. This central box lists 'Increment Inputs: Enter Top, Enter Centre', 'Decrement Inputs: Exit', and 'Threshold Operator: Greater than or equal to, Threshold Value: 20'.

11.8.4 Not

A logical operator that generates an event when the input rule becomes false.

11.8.4.1 Graph View



11.8.4.2 Form View

A configuration form for the "Not" rule. It includes a "Type" dropdown set to "Not", a "Name" text field containing "Not", a "Can Trigger Actions" checkbox that is checked, and an "Input" dropdown set to "None". A red square icon is visible in the top right corner.

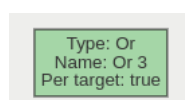
11.8.4.3 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	"Not #"
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active
Input	The input rule	None



11.8.5 Or

A logical operator that combines two rules and fires events if either input is true.

11.8.5.1 Graph View



11.8.5.2 Form View

Type:	Or	 
Name:	Or 3	
Can Trigger Actions	<input checked="" type="checkbox"/>	
Input A:	None	
Input B:	None	
Per Target	<input checked="" type="checkbox"/>	

11.8.5.3 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	"Or #"
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active
Input A	The first input	None
Input B	The second input	None
Per Target	Fire one event per tracked object	Active

If we consider a scene with two Presence rules connected to two separate zones, connected by an OR rule, the table below explains the behaviour of the **Per Target** property.

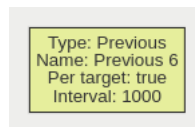
State	Per Target	Outcome
Object A in Input A, No object in input B	On	Two events generated, one for each object
No object in Input A, Object B in input B	On	Only one event generated (for Object B)
Object A in Input A, No object in input B	On	Only one event generated (for Object A)
Object A in Input A, No object in input B	Off	Only one event generated
No object in Input A, Object B in input B	Off	Only one event generated
Object A in Input A, No object in input B	Off	Only one event generated

Additionally, it is important to note that if the rule fires when **Per Target** is switched off, it will not fire again until it is 'reset', i.e. until the OR condition is no longer true.




11.8.6 Previous

A logical operator that triggers for input events which were active at some point in a past window of time. This window is defined as between the current time and the period before the current time (specified by the Interval value).

11.8.6.1 Graph View



11.8.6.2 Form View

Type:	Previous	
Name:	Previous 6	 
Can Trigger Actions	<input checked="" type="checkbox"/>	
Input:	<div>None </div>	
Per Target	<input checked="" type="checkbox"/>	
Interval	1000	ms

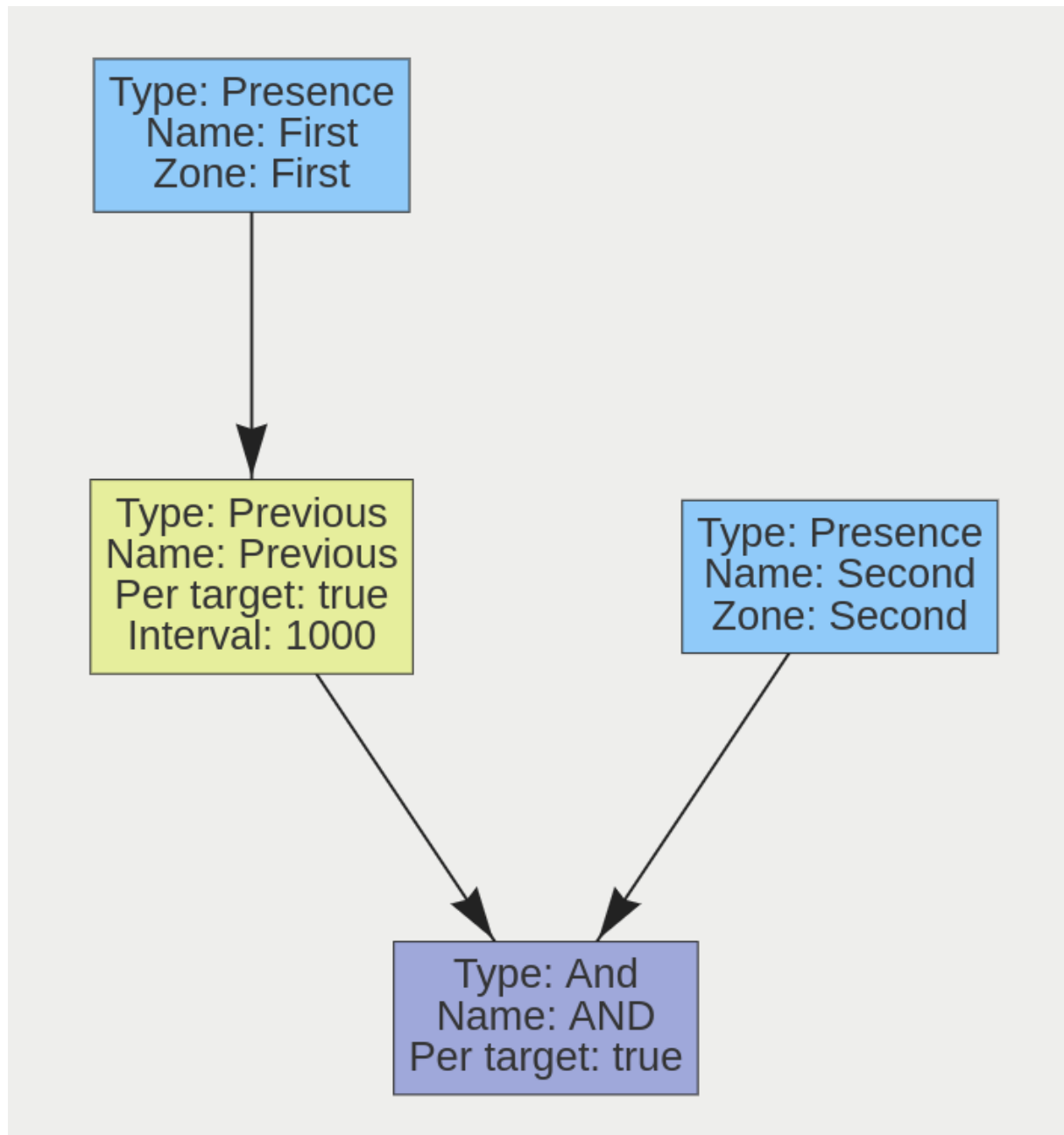
11.8.6.3 Configuration

Property	Description	Default Value
Name	A user-specified name for this rule	"Previous #"
Can Trigger Actions	Specifies whether events generated by this rule trigger actions	Active
Input	The input rule	None
Per Target	Fire one event per tracked object	Active
Interval	The time in milliseconds	1000 ms

11.9 Combined Rule Examples

11.9.1 Double-knock Rule

The 'double-knock' logical rule triggers when an object enters a zone which had previously entered another defined, zone within a set period of time. The `interval` on the Previous rule decides how much time can elapse between the object entering the first and second zone. The graph for a double-knock logical rule is as follows:

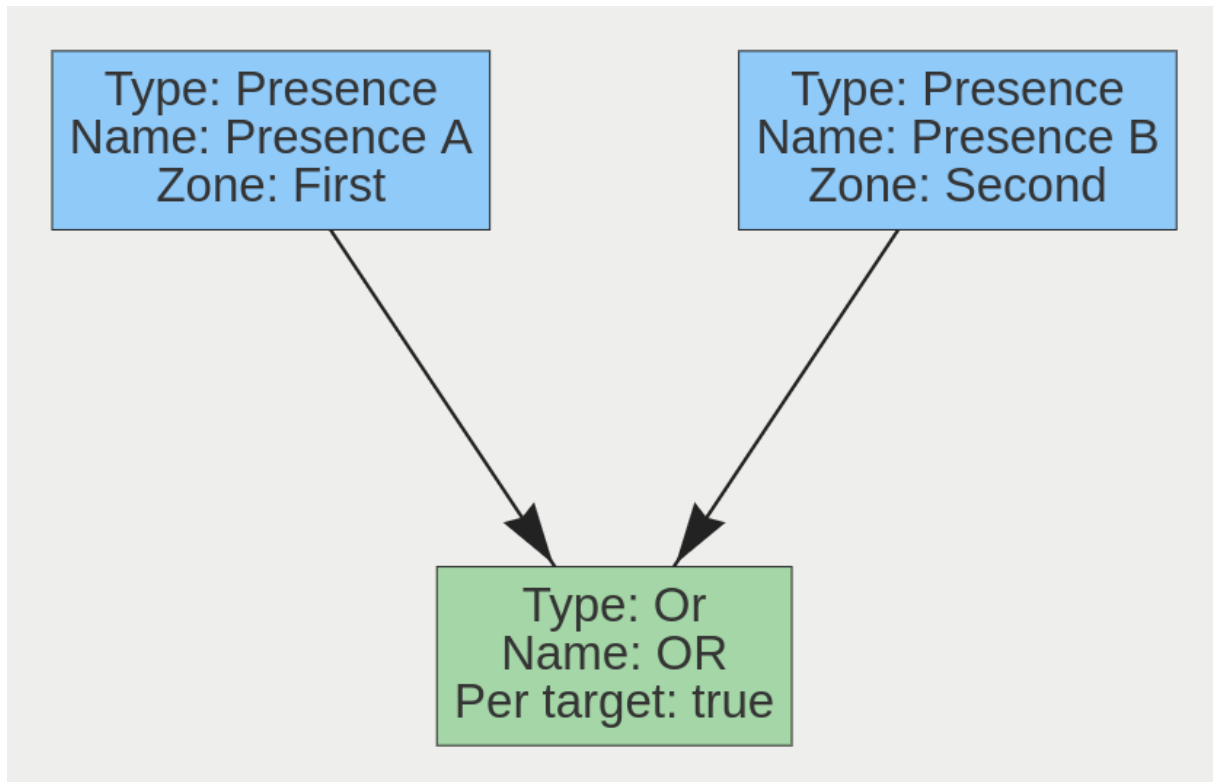


The rule may be interpreted as follows: 'An object is in Zone 2, and was previously in Zone 1 in the last 1000 milliseconds'. This rule can be used as a robust way to detect entry into an area. Since the object

has to enter two zones in a specific order, it has the ability to eliminate false positives that may arise from a simple Presence rule.

11.9.2 Presence in A or B

This rule triggers when an object is present in either Zone A or Zone B. Its graph is as follows:



A typical use case for this rule is having multiple areas where access is prohibited, but the areas cannot be easily covered by a single zone. Two zones can be created, associated with two separate Presence rules, and they can then be combined using an Or rule.

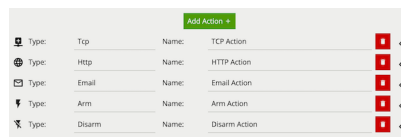
11.10 Usage notes

- Use the expanded view to build the Logical Rules graph, checking the graph view for correctness.
- Use the docked view to evaluate and tweak the graph once its overall structure is correct.
- The **Per Target** property on a rule will affect all rules below it.
- The Logical Rules Editor prevents the deletion of logical rules that have rules depending on them.

Chapter 12

Actions

Actions are user configured outputs which can be triggered by a variety of events that occur within VCAserver.



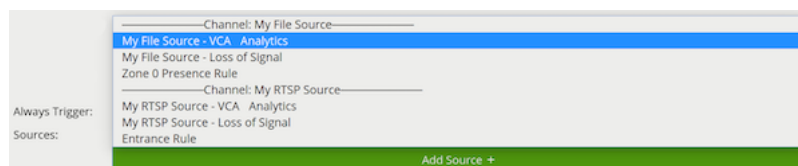
			Add Action +
Type:	Tcp	Name:	TCP Action
Type:	Http	Name:	HTTP Action
Type:	Email	Name:	Email Action
Type:	Arm	Name:	Arm Action
Type:	Disarm	Name:	Disarm Action

Common Properties:

- **Type:** The type of the action, e.g. TCP, Email, Digital Output etc.
- **Name:** The user configured name for the action.
- **Always Trigger:** If this setting is checked, the action will trigger even if the system is **disarmed**.

12.1 Event Sources

Any action can have multiple event sources assigned to it. Once an event source is assigned to an action, any event of that type will trigger the action. Available event sources are grouped according to **Video Sources** including either, customer defined **logical rules** (with the **Can Trigger Action** box checked), or **loss of signal** events and any configured **Digital Input**, **Armed**, **Disarmed** or **Interval** sources.



12.2 Action Types

12.2.1 TCP

The TCP action sends data to a remote TCP server when triggered. The format of the body is configurable with a mixture of plain text and **Tokens**, which are substituted with event-specific values at the time an event is generated.

The screenshot shows the configuration for a 'TCP Alert' action. The 'Type' is set to 'Tcp' and the 'Name' is 'TCP Alert'. The 'URI' is '192.168.1.100' and the 'Port' is '9000'. The 'Body' is set to 'Custom' and contains the template string: `start-time: {{time.start.iso8601}} end-time: {{time.end.iso8601}} event-id: {{id}} event-name: {{name}}`. Below the body, there are dropdown menus for 'Line endings' (set to 'Unknown') and 'Select a token to add it to the template string'. The 'Always Trigger' checkbox is unchecked. The 'Sources' section has an 'Add Source +' button and a 'Test' button.

- **URI:** The IP address or hostname of the remote TCP server where the event data should be transmitted.
- **Port:** The port on which the remote TCP server is listening for incoming TCP connections.
- **Body:** The body of the TCP message to transmit. Can be a mixture of plain text and any supported **Tokens**, which will be replaced with event-specific data at the time an event is generated. The TCP Body supports escape characters, allowing for inputs such as NULL terminated strings (`\x00`). In turn these escape characters can be escaped through the use of an additional `\` e.g. (`\\x00`).

See the **Tokens** topic for full details about the token system and example templates.

12.2.2 Email

The email action sends events in pre- or user-configured formats to remote email servers.

Type:

Email

Name:

Email Action

✕
▢
▾

Server:

aspmx.l.google.com

Port:

25

Username:

wile.e.coyote@acme.com

Password:

.....

From:

wile.e.coyote@acme.com

To:

road.runner@acme.com

Cc:

Enter 'Cc' address list

Bcc:

Enter 'Bcc' address list

Subject:

Test email from VCAbridge

Body:

Select a token to add it to the template string ▾

Custom ▾

Event Name: {{name}}
Event start: {{start.iso8601}}
Event end: {{end.iso8601}}

Line endings: Unix (LF) ▾

Select a token to add it to the template string ▾

Enable Authentication:
☒

Verify Certificate:
☒

Send Snapshots:
☒

Snapshot Quality:

Average ▾

Interval between snapshots:

500 ms

Number of snapshots before event:

1

Number of snapshots after event:

1

Encryption:

None ▾

Always Trigger:
☐

Sources:

Add Source +

Test

- **Server:** SMTP server address.
- **Port:** SMTP server port.
- **Username:** The username of the email account used to send emails on the SMTP server.
- **Password:** The password of the email account used to send emails on the SMTP server.
- **From:** The email address of the sender.

- **To:** The email address of the recipient.
- **Cc:** Email address/addresses of any carbon-copy recipients.
- **Bcc:** Email address/addresses of any blind carbon-copy recipients.
- **Subject:** The tokenised template of the email subject.
- **Body:** Tokenised template of the email body.
- **Enable Authentication:** Check to enable SMTP authentication.
- **Verify Certificate:** Check to verify the remote SSL certificate.
- **Send Snapshots:** Check to attach annotated snapshots to the email.
- **Snapshot Quality:** Select the quality of the snapshots attached to the email.
- **Interval between snapshots:** Set the snapshot capture rate in 250 milliseconds (ms) increments.
- **Number of Snapshots sent before event:** Set the number of pre-event snapshots to attach to the email.
- **Number of Snapshots sent after event:** Set the number of post-event snapshots to attach to the email.
- **Encryption:** The type of encryption used for SMTP communication. Valid options are None, SSL and STLS.

12.2.3 HTTP

The HTTP action sends a `text/plain` HTTP request to a remote endpoint when triggered. The URL, HTTP headers and message body are all configurable with a mixture of plain text and **Tokens**, which are substituted with event-specific values at the time an event is generated. Additionally, snapshots from the camera can be sent as a `multipart/form-data` request with the configured snapshots included as `image/jpeg`'s. HTTP actions are sent using the HTTP/1.1 standard.

Type: Http Name: HTTP Action

URI: http://api.genericvms.com/create-event&event-name={{name}}

Port: 80

Headers: None

Body: Custom

Event Name: {{name}}
 Event Type: {{type.string}}
 Event Time: {{start.iso8601}}

Line endings: Unix (LF)

Method: POST

Enable Authentication: ☒

Username: ddmin

Password: *****

Send Snapshots: ☐

Snapshot Quality: Average

Interval between snapshots: 500 ms

Number of snapshots before event: 4

Number of snapshots after event: 4

Multipart request name: vca

Always Trigger: ☐

Sources: Add Source +

Test

- URI:** The remote URI to request when executing the HTTP action. As illustrated in the figure, the URI can contain **Tokens**, which will be replaced with event-specific data at the time an event is generated. If specifying user credentials in plain text is undesirable, they can be specified in the Header section encoded as a base 64 string as part of a standard HTTP Authorization header.

- **Port:** The remote server port.
- **Headers:** Specifies any HTTP headers to send in the HTTP request. Examples may include `Authorization` or `Content-Type` headers. Any necessary headers will normally be specified by the remote server API. Each header should be placed on a new line. When the headers are transmitted a CRLF (`\r\n`) is automatically inserted between each header, and between the last header and the message body.
- **Body:** Specifies the body of the HTTP request. Can be a mixture of plain text and any supported **Tokens**, which will be replaced with event-specific data at the time an event is generated.
- **Method:** The HTTP request method (verb). Can be one of GET, POST, PUT, DELETE, HEAD. This setting will normally be specified by the remote server API.
- **Enable Authentication:** Check to enable authentication (supports both HTTP Basic and HTTP Digest).
- **Username:** The username to use for authentication.
- **Password:** The password to use for authentication.
- **Send Snapshots:** Check to attach annotated snapshots to the HTTP request.
- **Snapshot Quality:** Select the quality of the snapshots attached to the HTTP request.
- **Interval between snapshots:** Set the snapshot capture rate in 250 milliseconds (ms) increments.
- **Number of Snapshots sent before event:** Set the number of pre-event snapshots to attach to the HTTP request.
- **Number of Snapshots sent after event:** Set the number of post-event snapshots to attach to the HTTP request.
- **Multipart request name:** Sets the text assigned to multipart name. This name will need to be reflected in any scripts that handle this HTTP request, for example in php; this would be by using `$_FILES['vca']` where `vca` is the string set in the multipart field.

See the **Tokens** topic for full details about the token system and example templates.

12.2.4 Digital Output

A digital output is a logical representation of a digital output hardware channel. To configure the properties of a physical digital output channel, such as activation time, refer to the **Digital IO** page.

The screenshot shows a configuration window for a Digital Output. At the top, there's a header bar with 'Type: Digital Output', 'Name: DO3', and a red square icon with a dropdown arrow. Below this, the 'Device:' is set to 'DO 3'. The 'Always Trigger:' checkbox is checked with a green checkmark. Under the 'Sources:' section, 'Gangnam 720 - Analytics' is listed with a red square icon. Below the sources, there is a green button labeled 'Add Source +' and a green button labeled 'Test'.

- **Device:** The physical digital output channel assigned to the logical digital output.

12.2.5 Arm

The Arm action sets the device state to **armed** when triggered.

12.2.6 Disarm

The Disarm action sets the device state to **disarmed** when triggered.

12.3 Arm/Disarm State

The Arm/Disarm functionality provides a means of disabling/enabling all of the configured actions. For example, users may wish to disable all actions when activity is normal and expected (e.g. during normal working hours) and re-enable the actions at times when activity is not expected.

The Arm/Disarm state can be toggled manually by clicking the  icon in the **Navigation Bar** or by using the Arm or Disarm actions.

Chapter 13

Calibration

Camera calibration is required in order for VCAserver to classify objects into different object classes. Once a channel has been calibrated, VCAserver can infer real-world object properties such as speed, height and area and classify objects accordingly.

Camera calibration is split into the following sub-topics:

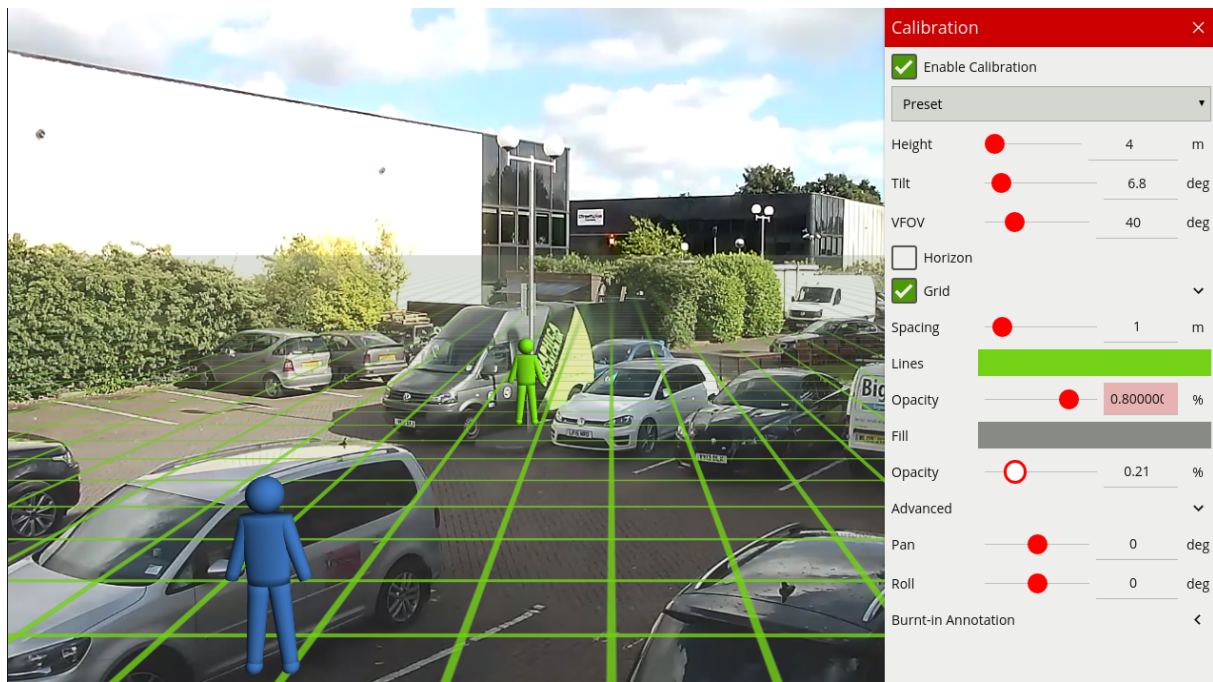
- Enabling Calibration
- Calibration Controls
- Calibrating a Channel
- Advanced Calibration Parameters

13.1 Enabling Calibration

By default calibration is disabled. To enable calibration on a channel, check the **Enable Calibration** checkbox.

13.2 Calibration Controls

The calibration page contains a number of elements to assist with calibrating a channel as easily as possible. Each is described below.



13.2.1 3D Graphics Overlay

During the calibration process, the features in the video image need to be matched with a 3D graphics overlay. The 3D graphics overlay consists of a green grid that represents the ground plane. Placed on the ground plane are a number of 3D mimics (people-shaped figures) that represent the dimensions of a person with the current calibration parameters. The calibration mimics are used for verifying the size of a person in the scene and are 1.8 metres tall.

The mimics can be moved around the scene to line up with people (or objects which are of a known, comparable height) to a person.

13.2.2 Mouse Controls

The calibration parameters can be adjusted with the mouse as follows: - Click and drag the ground plane to change the camera tilt angle. - Use the mouse wheel to adjust the camera height. - Drag the slider to change the vertical field of view.

Note: The sliders in the control panel can also be used to adjust the camera tilt angle and height.

13.2.3 Control Panel Items

The control panel (shown on the right hand side in the image above) contains the following controls:

- **Height:** Adjusts the height of the camera
- **Tilt:** Adjusts the tilt angle of the camera
- **VFOV:** Adjusts the *vertical* field of view of the camera. **Note:** A correct value for the camera vertical field of view is important for accurate calibration and classification.

- **Horizon:** Enables/disables the horizon display. Useful to line up against a horizon in a deep scene.
- **Grid:** Enables/disables the ground plane grid display. The expand/collapse control (<) exposes additional settings to vary the colour, opacity and size of the ground plane grid.
- **Advanced:** Exposes advanced settings for controlling the pan and roll of the camera.
- **Burnt-in Annotation:** Exposes the **Burnt-in Annotation** controls for convenience.

13.2.4 Context Menu Items

Right-clicking the mouse (or tap-and-hold on a tablet) on the grid displays the context menu:



Performing the same action on a mimic displays the mimic context menu:



The possible actions from the context menu are:

- **II** Pause the video. Pausing the video can make it easier to align mimics up with objects in the scene.
- **▶** Re-starts playing the video after it was previously paused.
- **†** Adds an extra mimic to the ground plane.
- **-** Removes the currently selected mimic from the ground plane.

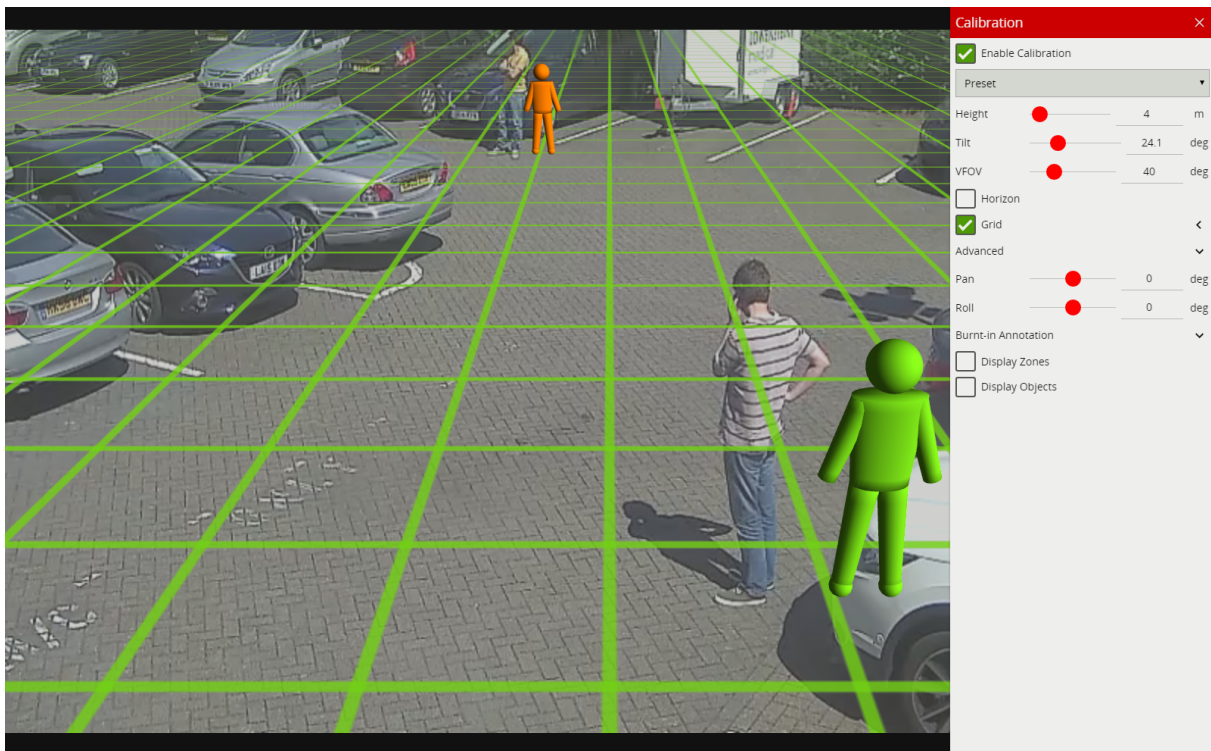
13.3 Calibrating a Channel

Calibrating a channel is necessary in order to estimate object parameters such as height, area, speed and classification. If the height, tilt angle and vertical field of view corresponding to the installation are known, these can simply be entered as parameters in the appropriate fields in the control panel.

If however, these parameters are not explicitly known this section provides a step-by-step guide to calibrating a channel.

13.3.1 Step 1: Find People in the Scene

Find some people, or some people-sized objects in the scene. Try to find a person near the camera, and a person further away from the camera. It is useful to use the play/pause control to pause the video so that the mimics can be accurately placed. Place the mimics on top of or near the people:



13.3.2 Step 2: Enter the Camera Vertical Field of View

Determining the correct vertical field of view is important for an accurate calibration. The following table shows pre-calculated values for vertical field of view for different sensor sizes.

		Focal Length(mm)							
		1	2	3	4	5	6	7	8
CCD Size (in)	CCD Height(mm)								
1/6"	1.73	82	47	32	24	20	16	14	12
1/4"	2.40	100	62	44	33	27	23	19	17
1/3.6"	3.00	113	74	53	41	33	28	24	21
1/3.2"	3.42	119	81	59	46	38	32	27	24
1/3"	3.60	122	84	62	48	40	33	29	25
1/2.7"	3.96	126	89	67	53	43	37	32	28
1/2"	4.80	135	100	77	62	51	44	38	33
1/1.8"	5.32	139	106	83	67	56	48	42	37

	Focal Length(mm)	1	2	3	4	5	6	7	8
2/3"	6.60		118	95	79	67	58	50	45
1"	9.60		135	116	100	88	77	69	62
4/3"	13.50			132	119	107	97	88	80

	Focal Length(mm)	9	10	15	20	30	40	50
CCD Size (in)	CCD Height(mm)							
1/6"	1.73	11	10	7				
1/4"	2.40	15	14	9	7			
1/3.6"	3.00	19	12	11	9	6		
1/3.2"	3.42	21	16	13	10	7		
1/3"	3.60	23	20	14	10	7	5	
1/2.7"	3.96	25	22	15	11	8	6	
1/2"	4.80	30	27	18	14	9	7	5
1/1.8"	5.32	33	30	20	15	10	8	6
2/3"	6.60	40	37	25	19	13	9	8
1"	9.60	56	51	35	27	18	14	11
4/3"	13.50	74	68	48	37	25	19	15

If the table does not contain the relevant parameters, the vertical FOV can be estimated by viewing the extremes of the image at the top and bottom. Note that without the correct vertical FOV, it may not be possible to get the mimics to match people at different positions in the scene.

13.3.3 Step 3: Enter the Camera Height

If the camera height is known, type it in directly. If the height is not known, estimate it as far as possible and type it in directly.

13.3.4 Step 4: Adjust the Tilt Angle and Camera Height

Adjust the camera tilt angle (and height if necessary) until both mimics are approximately the same size as a real person at that position in the scene. Click and drag the ground plane to change the tilt angle and use the mouse wheel or control panel to adjust the camera height.

The objective is to ensure that mimics placed at various locations on the grid line up with people or people-sized- objects in the scene.

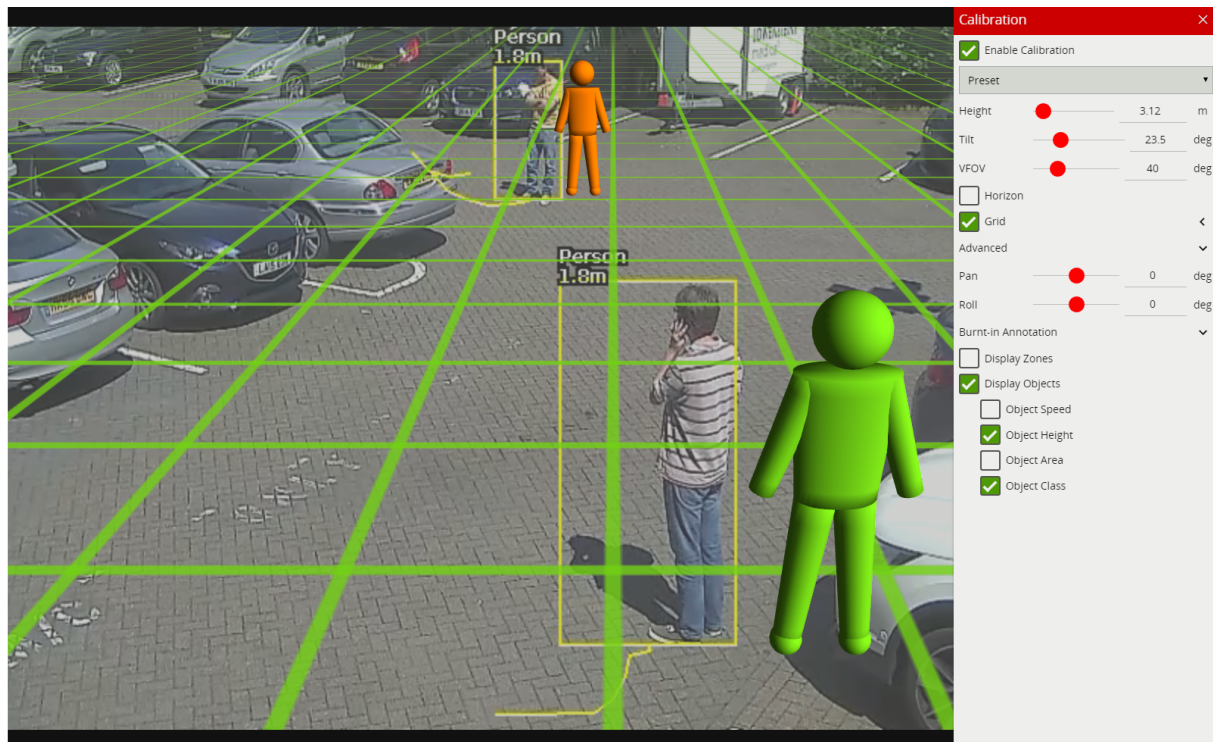
Once the parameters have been adjusted, the object annotation will reflect the changes and classify the objects accordingly.

13.3.5 Step 5: Verify the Setup

Once the scene is calibrated, drag or add mimics to different locations in the scene and verify they appear at the same size/height as a real person would. Validate that the height and area reported by

the VCAserver annotation looks approximately correct. Note that the burnt-in -annotation settings in the control panel can be used to enable and disable the different types of annotation.

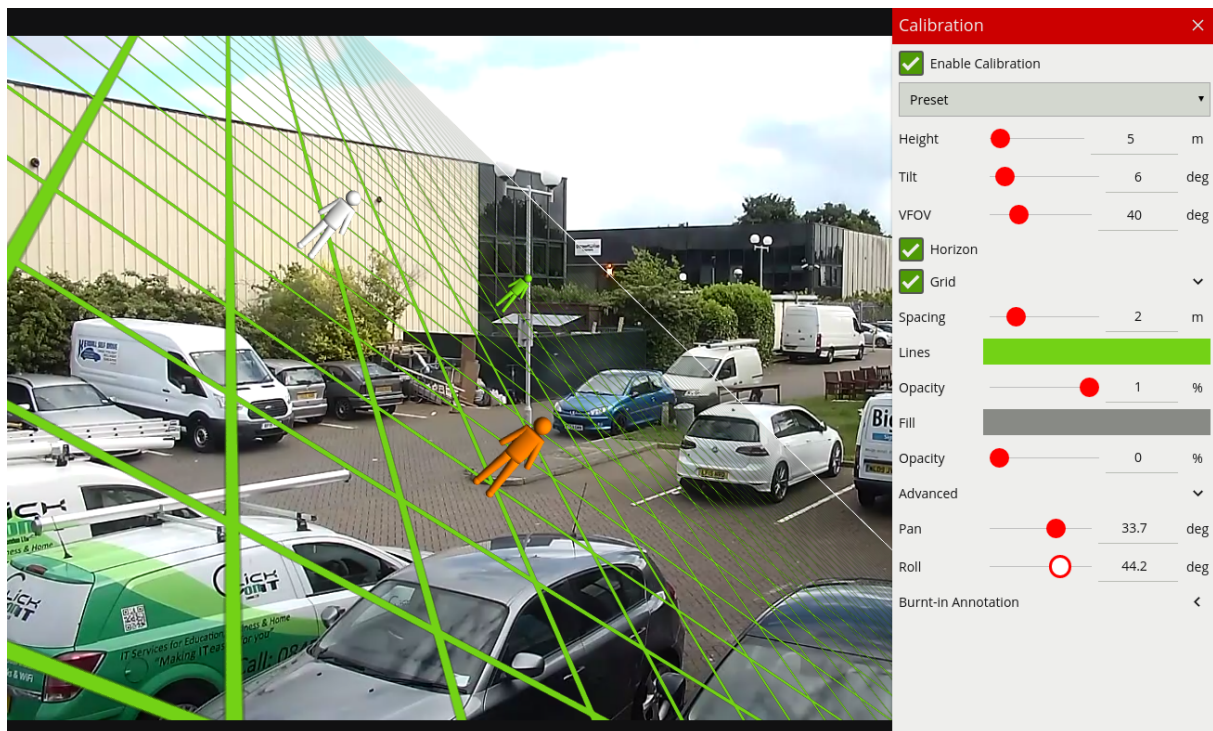
Repeat step 4 until the calibration is acceptable.



Tip: If it all goes wrong and the mimics disappear or get lost due to an odd configuration, select one of the preset configurations to restore the configuration to normality.

13.4 Advanced Calibration Parameters

The advanced calibration parameters allow the ground plane to be panned and rolled without affecting the camera calibration parameters. This can be useful to visualize the calibration setup if the scene has pan or roll with respect to the camera.



Note: the pan and roll advanced parameters only affect the orientation of the 3D ground plane so that it can be more conveniently aligned with the video scene, and does not actually affect the calibration parameters.

Chapter 14

Classification

VCAserver can define a moving object's class, using either deep learning models or by using properties extracted from an object in a **calibrated** scene.

Both methods of classification are applied through the use of filters in the **rules** interface. Classification filters allow an object, which has triggered a rule, to be evaluated against it's predicted class and filtered out if needed.

14.1 Object Classification

Once a camera view has been **calibrated**, each detected object in that view will have a number of properties extracted including object area and speed.

VCAserver's object classification performs classification by comparing these properties to a set of configurable object classifiers. VCAserver comes pre-loaded with the most common object classifiers, and in most cases these will not need to be modified.

Channels running the Deep Learning People Tracker or the Deep Learning Object Tracker cannot be calibrated. Therefore, Object Classification is not available when these trackers are selected.

14.1.1 Configuration

In some situations it might be desirable to change the classifier parameters, or add new object classifiers. The classification menu can be used to make these changes.

Add Classifier +

⋮	Name:	Person	Speed:	0	to	20	km/h	Area:	0.5	to	2	m ²	■
<hr style="border-top: 1px dashed #ccc;"/>													
⋮	Name:	Vehicle	Speed:	0	to	200	km/h	Area:	4	to	100	m ²	■
<hr style="border-top: 1px dashed #ccc;"/>													
⋮	Name:	Clutter	Speed:	0	to	50	km/h	Area:	0	to	0.4	m ²	■
<hr style="border-top: 1px dashed #ccc;"/>													
⋮	Name:	Group Of People	Speed:	0	to	20	km/h	Area:	2.1	to	3.9	m ²	■

Each of the UI elements are described below:

- ⋮ : Click and drag to rearrange the order of the classification groups.
- **Name:** Specifies the name of the classification group.
- **Speed:** Sets the speed range for the classification group. Objects which fall within the speed and area ranges will be classified with this group.
- **Area:** Sets the area range for the classification group. Objects which fall within the speed and area ranges will be classified with this group.
- ■ : Deletes the classification group.

To add a new classifier click the **Add Classifier** button Add Classifier +.

Calibration must be **enabled** on each channel object classification is to be used on. If not enabled, any rules that include an object filter will not trigger.

14.1.2 Classification (Object)

Objects are classified according to how their calibrated properties match the classifiers. Each classifier specifies a speed range and an area range. Objects which fall within both ranges of speed and area will be classified as being an object of the corresponding class.

Note: If multiple classes contain overlapping speed and area ranges then object classification may be ambiguous, since an object will match more than one class. In this case the actual classification is not specified and may be any one of the overlapping classes.

The classification data from object classification can be accessed via **template tokens**.

14.2 Deep Learning Filter

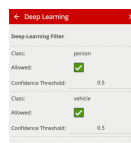
VCAserver also supports classification through the use of the deep learning filter. In this case an object, which has triggered a rule, can be analysed using the deep learning filter and a predicted class and confidence level returned. The available object classes are defined by the model.

On VCAserver the Deep Learning Filter can use GPU acceleration, see [Deep Learning Requirements](#) for hardware requirements.

Without GPU acceleration the deep learning filter will use the CPU, enabling the deep learning filter on multiple channels which are generating a high volume of events, (more than 1 per second) may result in poor performance of the system and is not advised.

14.2.1 Configuration

The Deep Learning page allows the user to configure the deep learning filter in VCAserver.



Each of the possible object classes has additional parameters:

- **Allowed:** Whether this object type will be allowed to pass through the filter. If this is unchecked, any objects classified as this type will not trigger any actions.
- **Confidence Threshold:** A value (0.0 - 1.0) representing the minimum confidence level required for classification. Any objects with a lower classification score than this minimum value will be filtered out and will not trigger any actions.

14.2.2 Classification (DL)

When an object triggers the deep learning filter, the analysed object will either be defined as one of the detectable object classes or as background. While an object is triggering a rule (i.e. a Presence rule) the deep learning filter will continue to evaluate its prediction, until the deep learning filter returns an object of interest as defined by the configuration.

If the filter classifies an object which generated an event as background, the event will be filtered out and any attached actions will not be triggered.

The classification data from the deep learning filter can also be accessed via [template tokens](#).

14.3 Deep Learning People Tracker

By the nature of the Deep learning People Tracker's detection methodology, every tracked object is, by definition, classified as `Person`. The Deep Learning People Tracker will not track an object unless it is classified as a `Person`. Additionally, no calibration is required for the tracker's classification to work.

See [Deep Learning Requirements](#) for hardware requirements for this algorithm.

14.4 Deep Learning Object Tracker

By the nature of the Deep learning Object Tracker's detection methodology, every tracked object is by definition classified as one of the following classes:

- Person
- Car
- Van (includes Van, Minibus, Bus)
- Truck (includes Truck, Lorry)
- Cyclist
- Motorbike
- Bag

The Deep Learning Object Tracker will not track an object unless it is classified as one of the above classes. Additionally, no calibration is required for the tracker's classification to work.

See [Deep Learning Requirements](#) for hardware requirements for this algorithm.

14.5 Object Classification and the Deep Learning Filter

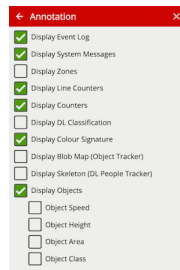
The Deep Learning Filter, Deep Learning Object Tracker and Deep Learning People Tracker does not require the source input to have been **calibrated**, or the **object classifier** to be configured. Similarly, the settings of the Deep Learning Filter are entirely independent from Object Classification.

Classification methods (deep learning based or object based) are designed to be used independently. However, Object Classification can be used in tandem with the Deep Learning Filter, when an appropriate rule graph is constructed. However, when using both together care should be taken. For example, as the Deep Learning Filter is trained to detect specific objects, if custom object classes have been configured in the object classifier, e.g. **small animal**, the Deep Learning filter may erroneously filter those alerts out, as **small animal** is not a class the Deep Learning Filter is trained to recognise. In these cases, use of the Deep Learning filter is not recommended.

Chapter 15

Burnt-in Annotation

Burnt-in Annotations allow VCAserver metadata to be overlaid on to the raw video stream. The burnt-in annotation settings control which VCAserver metadata (objects, events, etc) is rendered into the video stream.

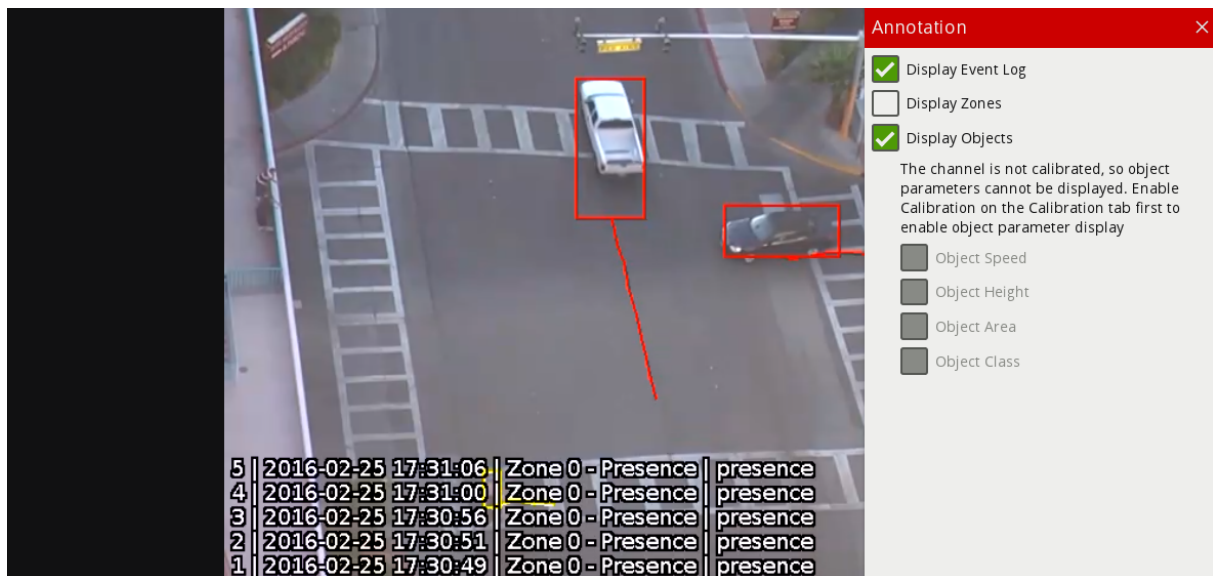


Note:

- To display object parameters such as speed, height, area and classifications, the channel must first be **calibrated**.
- To display DL Classification annotations, the channel must have an active Deep Learning Filter rule configured or the DL People Tracker enabled.
- To display colour signature annotations, the channel must have an active Colour Filter rule configured.
- Some annotations only apply to certain trackers, in such cases the required tracker is listed in brackets.

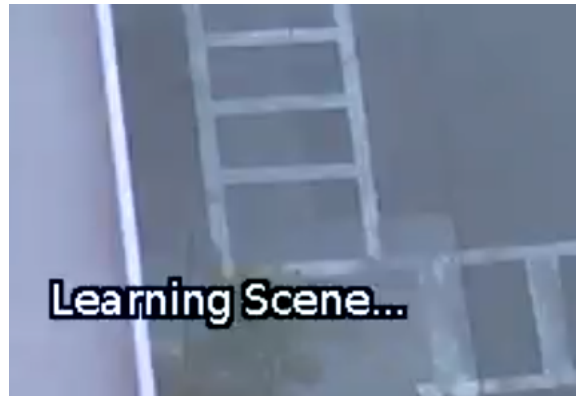
15.1 Display Event Log

Check the **Display Event Log** option to show the event log in the lower portion of the image.



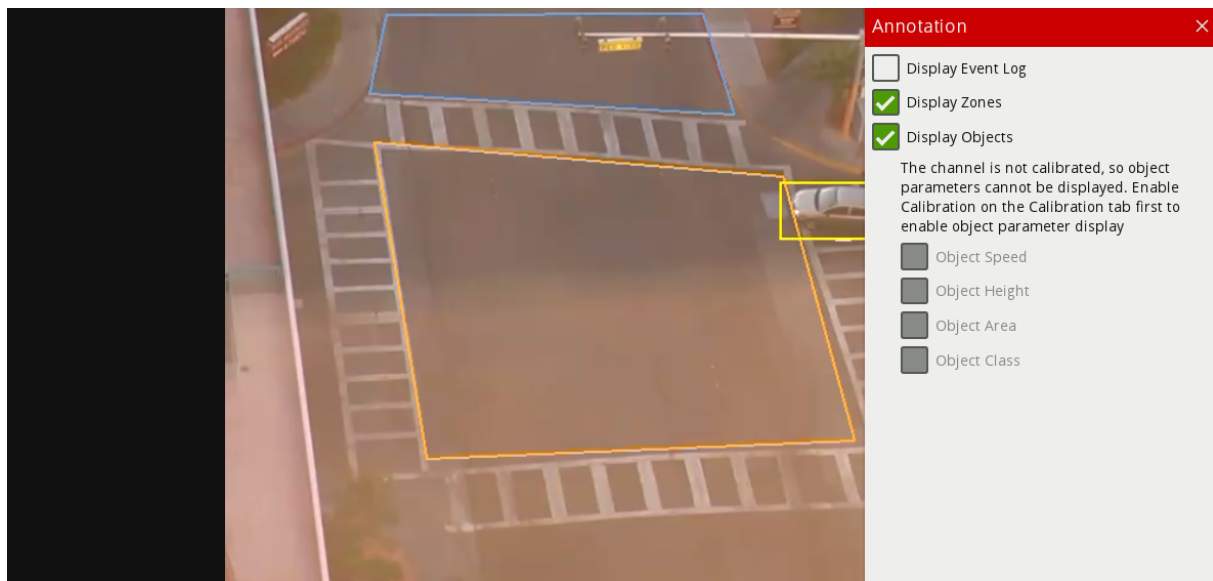
15.2 Display System Messages

Check the **Display System Messages** option to show the system messages associated with Learning Scene and Tamper.



15.3 Display Zones

Check the **Display Zones** option to show the outline of any configured zones.



15.4 Display Line Counters

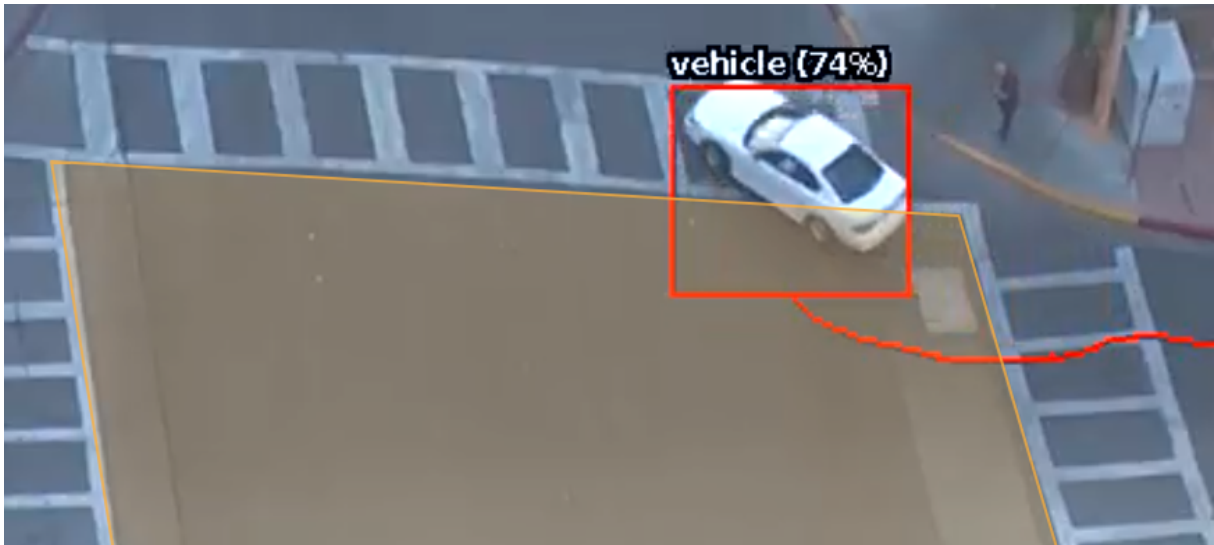
Check the **Display Line Counters** option to display the line counter calibration feedback information. See the [Rules](#) for more information.

15.5 Display Counters

Check the **Display Counters** option to display the counter names and values. See the [Counters](#) topic for more information.

15.6 Display Deep Learning Classification

Check the **Display DL Classification** option to show the class and confidence of objects evaluated by a deep learning model.



15.7 Display Colour Signature

Check the **Display Colour Signature** option to show the current top four colours, of a possible ten, found in a given bounding box.



15.8 Display Blob Map (Object Tracker)

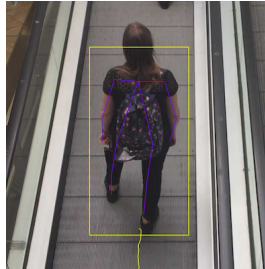
Check the **Display Blob Map (Object Tracker)** option to visualise the motion the object tracker is detecting in real time. Motion is represented as cyan blocks.



15.9 Display Skeleton (DL Person Tracker)

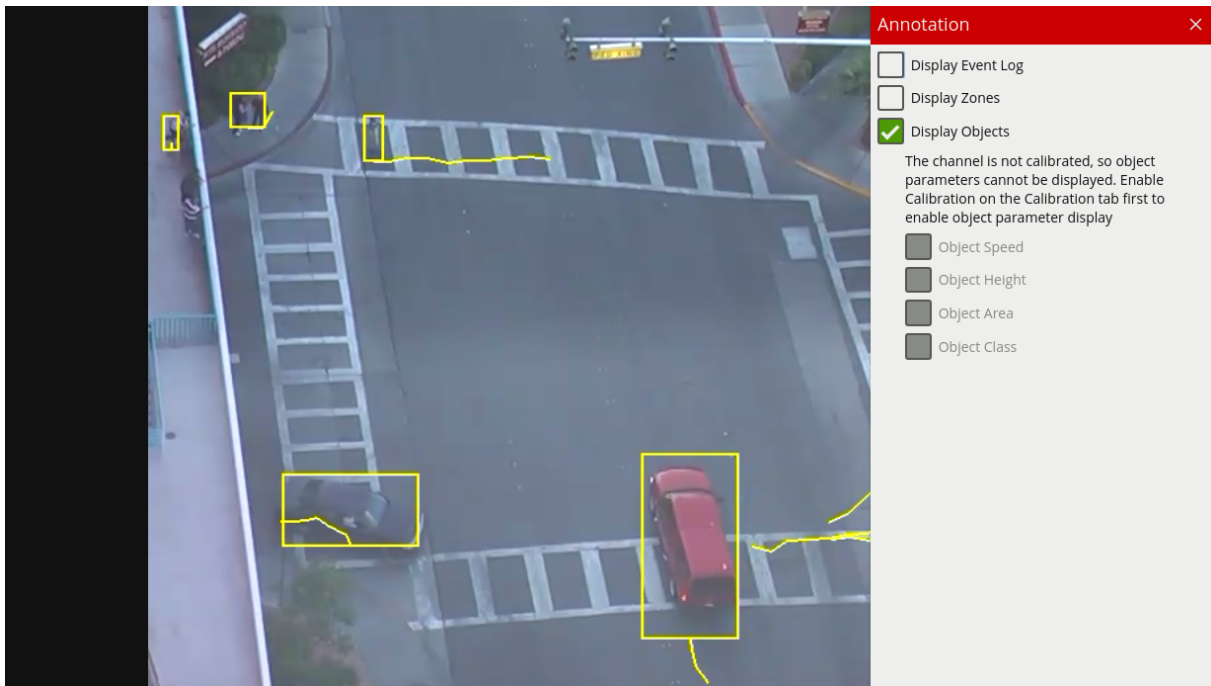
Check the **Display Skeleton (DL Person Tracker)** option to visualise the body part key point metadata, detected by the DL People Tracker.

These annotations may not appear every frame and not all the body parts of a tracked person may be detected each frame.



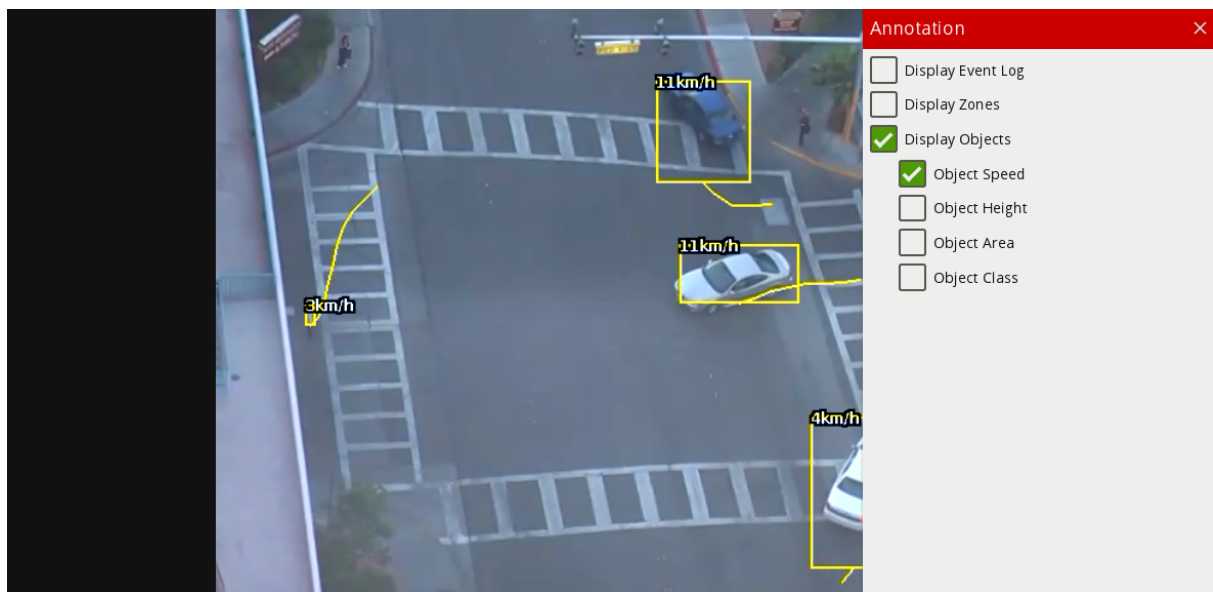
15.10 Display Objects

Check the **Display Objects** option to show the bounding boxes of tracked objects. Objects which are not in an alarmed state are rendered in yellow. Objects rendered in red are in an alarmed state (i.e. they have triggered a rule).



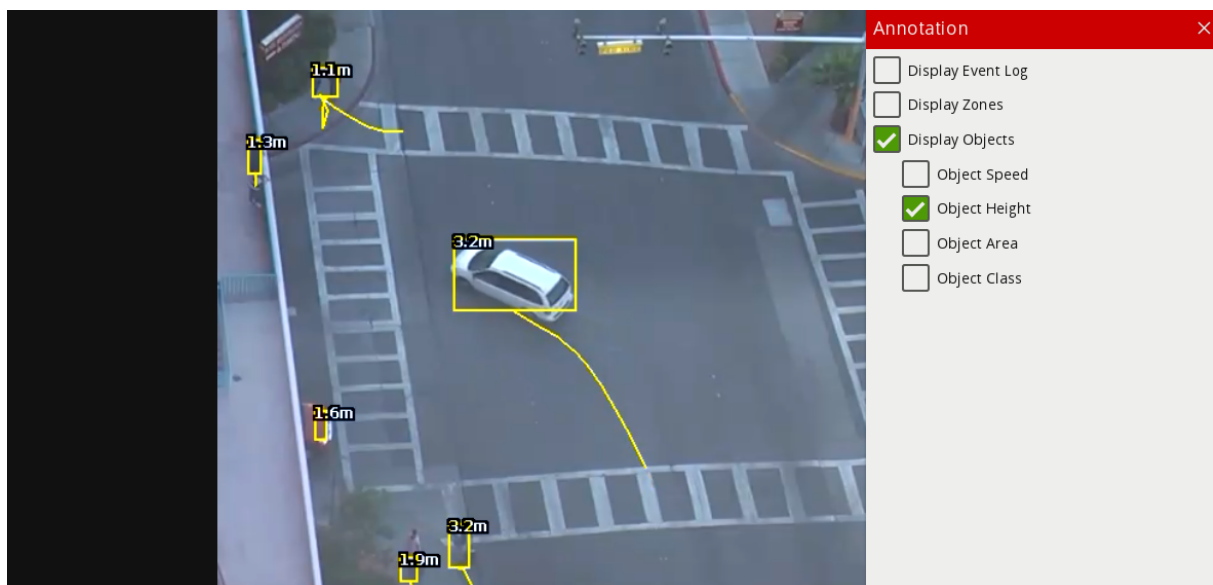
15.10.1 Object Speed

Check the **Object Speed** option to show the object speed.



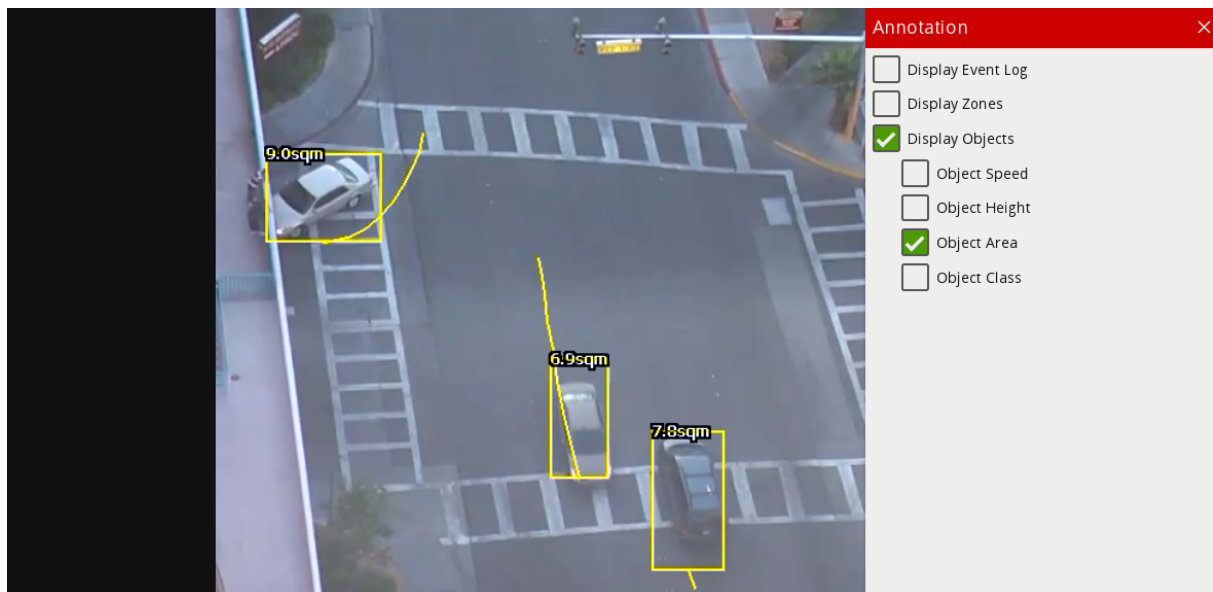
15.10.2 Object Height

Check the **Object Height** option to show the object height.



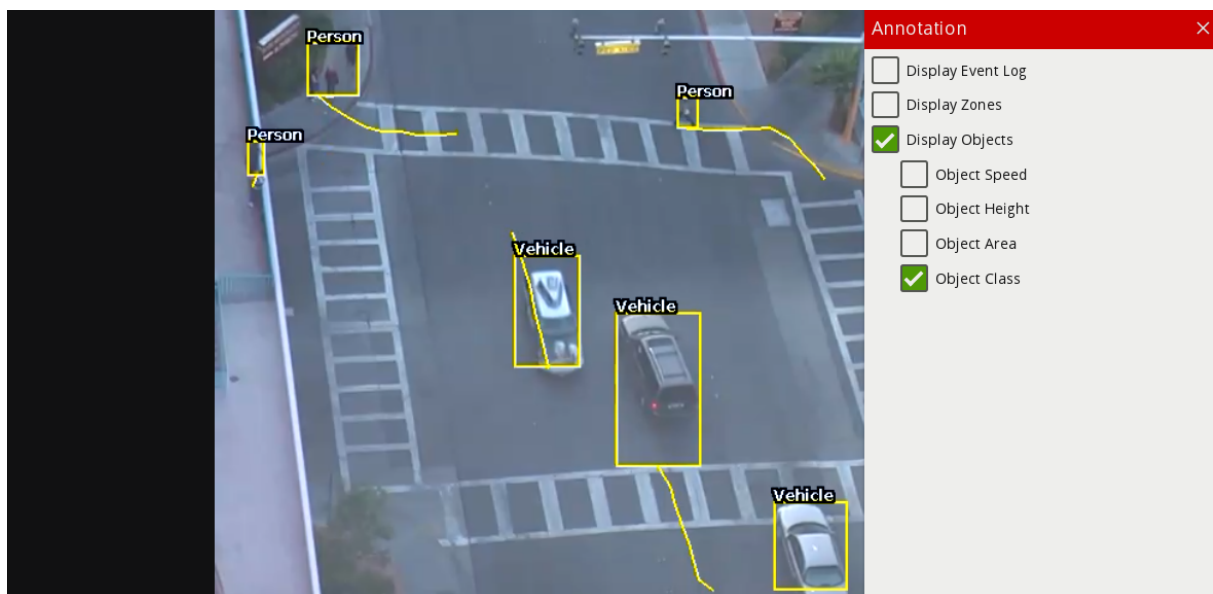
15.10.3 Object Area

Check the **Object Area** option to show object area.



15.10.4 Object Classification

Check the **Object Class** to show the object **Classification**.



Chapter 16

Tamper Detection

The Tamper Detection module is intended to detect camera tampering events such as bagging, de-focusing and moving the camera. This is achieved by detecting large persistent changes in the image.

Tamper Detection
Enabled: ☒
Duration: 20 Seconds
Area Threshold: 40 %
Suppress Alarm on Lights on/off: ☐
Suppress false alarms caused by major light changes. Enabling can reduce sensitivity to genuine alarms.

16.1 Enabling Tamper Detection

To enable tamper detection click the **Enabled** checkbox.

Tamper Detection
Enabled: ☐

16.2 Advanced Tamper Detection Settings

In the advanced tamper detection settings it is possible to change the thresholds for the area of the image which must be changed and the length of time it must be changed for before the tamper event is triggered.

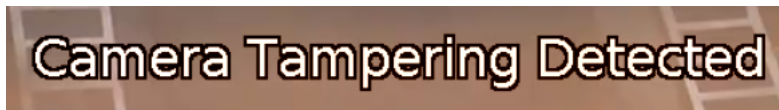
- **Duration:** the length of time that the image must be persistently changed before the alarm is triggered.

- **Area Threshold:** the percentage area of the image which must be changed for tampering to be triggered.
- **Suppress Alarm on Lights on/off:** Large fast changes to the image lighting such as switching on/off indoor lighting can cause false tamper events. Enable this option if this is likely to be a problem in the area where the camera is installed. However, this option will reduce sensitivity to genuine alarms so it is not recommended to be used if rapid light changes are not likely to be a problem

If false alarms are a problem the duration and/or area should be increased so that large transient changes such as close objects temporarily obscuring the camera do not cause false alarms.

16.3 Notification

When a tamper event is detected, a tamper event is generated. This event is transmitted through any output elements as well as being displayed in the video stream:



Chapter 17

Scene Change Detection

The scene change detection module resets the object tracking algorithm when it detects a large persistent change in the image. This prevents the tracking engine from detecting image changes as tracked objects which could be potential sources of false alarms.

The kinds of changes the scene change detection module detects are as follows:

- Sudden movement of a camera (e.g. due to repositioning or the use of a pan-tilt camera).
- Sudden obscuration of a camera (e.g. a vehicle parks in front of a camera obscuring most of its view).
- Gross illumination changes (e.g. lights being switched on/off, dazzling from car headlights).
- Day/night transitions (e.g. when a camera switches from colour to black/white during a night-day transition)

17.1 Scene Change Settings

There are 3 options for the scene change detection mode:

- **Automatic:** Detects scene changes automatically. This is the recommended setting unless the automatic mode is causing difficulties (e.g. re-learning the scene when unnecessary).
- **Manual:** Allows the user to adjust the parameters used by the scene change detection algorithm.
- **Disabled:** Disables the scene change detection.

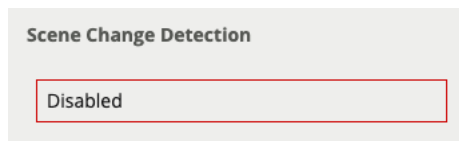
17.1.1 Automatic

This is the default setting and will automatically use the recommended settings. It is recommended to use the automatic setting unless the scene change detection is causing difficulties.



17.1.2 Disabled

Scene change detection is disabled.



Note that when the scene change detection is disabled, gross changes in the image will not be detected. For example, if a truck parks in front of the camera the scene change will not be detected and false events may occur as a result.

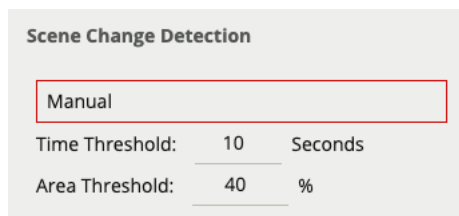
17.1.3 Manual

Allows user configuration of the scene change detection algorithm parameters.

If automatic mode is triggering in situations where it's not desired (e.g. it's too sensitive, or not sensitive enough) then the parameters can be adjusted to manually control the behaviour.

In the manual mode the following settings are available:

- **Time Threshold:** the length of time that the image must be persistently changed before the scene change is triggered and the tracking algorithm is reset.
- **Area Threshold:** the percentage area of the image which must be changed for the scene change to be triggered and the tracking algorithm to be reset.



When both the time and area thresholds are exceeded the scene is considered to have changed and will be reset.

If false scene change detections are a problem, the time and/or area should be increased so that large transient changes such as a close object temporarily obscuring the camera do not cause false scene change detections.

17.2 Notification

When a scene change is detected, the scene is re-learned and a message is displayed in the event log and annotated on the video

Learning Scene

Chapter 18

System Settings

The system settings page facilitates administration of system level settings such as network configuration, and authentication.

18.1 Network Settings

The network configuration of the device can be changed in the network settings configuration section:

Web Port:	<input type="text" value="8080"/>	<input type="button" value="Save"/>
RTSP Port:	<input type="text" value="8554"/>	<input type="button" value="Save"/>

- **Web Port:** Sets the port that the device's web server will listen on. The **Save** button must be clicked to apply the change.
- **RTSP Port:** Sets the port that the device's RTSP server will listen on. The **Save** button must be clicked to apply the change.

18.2 System Information

The system information section shows the Uptime of VCAbridge (how long the application has been running without restarting) as well as the device CPU and Memory usage:

System Information	
Uptime:	0 days 0 hours 0 minutes 29 seconds
CPU:	14%
Memory:	3913MB/17024MB

18.3 GPU Devices

The GPU devices section shows information on all the detected graphics processing units. Name and vendor information are provided for reference, with the current temperature, overall utilisation and memory usage:

GPU Devices	
NVIDIA GeForce GTX 1050 Ti	
Vendor:	NVIDIA
Temperature:	25C
Utilisation:	17%
Memory:	383MB / 4295MB

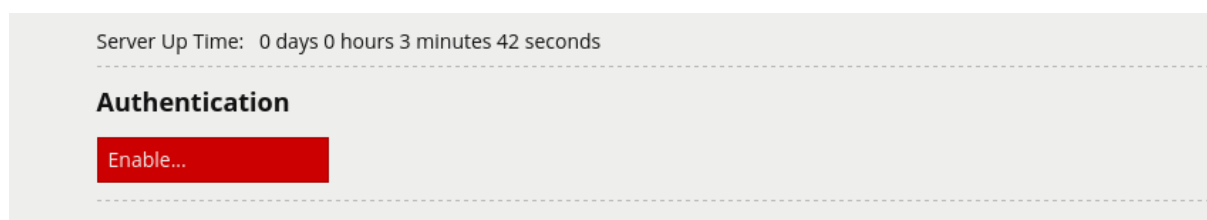
These values, combined with the system information can be used to determine if the current configuration is overly stressing the available hardware.

18.4 Authentication Settings

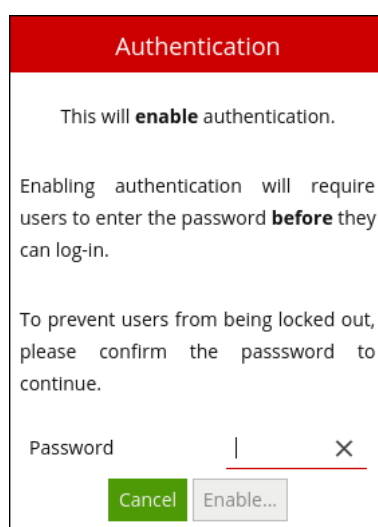
VCAserver can be protected against unauthorised access by enabling authentication. By default, authentication is enabled and the default credentials must be entered when accessing the device for the first time. Authentication applies to all functions including the web interface and API, RTSP server and discovery interfaces.

18.4.1 Enabling Authentication

Click the **Enable** button to enable authentication.

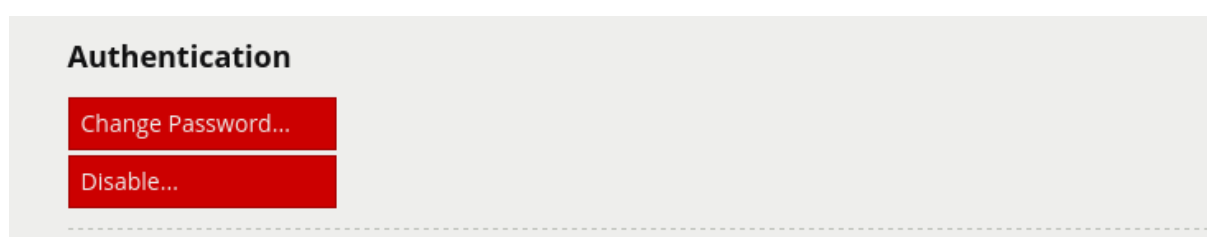


The password must be confirmed before authentication can be enabled in order to prevent the user being locked out of the device.

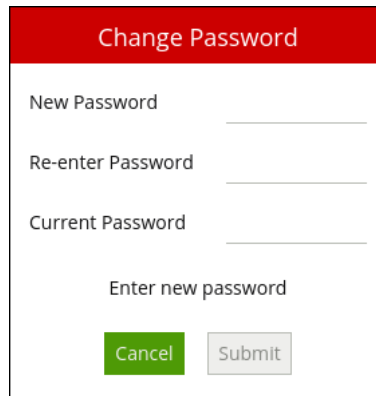


18.4.2 Changing the Password

Click the **Change Password** button to change the password.



Enter the new password, and confirm the current password in order to apply the changes.

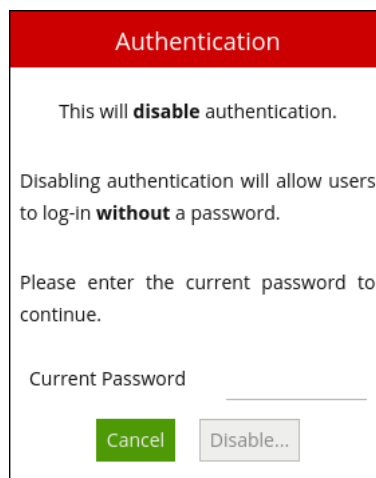


A dialog box titled "Change Password" with a red header. It contains three input fields labeled "New Password", "Re-enter Password", and "Current Password". Below the fields is the text "Enter new password". At the bottom are two buttons: "Cancel" (green) and "Submit" (grey).

WARNING: If the password is forgotten, the device will not be accessible. The only way to recover access to a device without a valid password is to perform a physical reset as described in the Forgotten Password section.

18.4.3 Disabling Authentication

Click the **Disable** button to disable authentication and allow users to access the device without entering a password. The password is required to disable authentication.



A dialog box titled "Authentication" with a red header. It contains the following text: "This will **disable** authentication.", "Disabling authentication will allow users to log-in **without** a password.", and "Please enter the current password to continue." Below the text is an input field labeled "Current Password". At the bottom are two buttons: "Cancel" (green) and "Disable..." (grey).

18.4.4 Default Credentials

The default credentials are as follows:

- **Username:** admin
- **Password:** admin

18.4.5 Forgotten Password

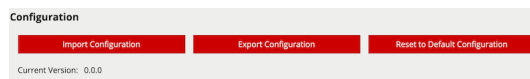
If a system becomes inaccessible due to a lost password, the only way to recover access to the device is to delete the configuration file VCAserver is using. This process differs between platforms:

- **VCAserver:** Navigate to configuration folder using the the filesystem browser and delete the entire configuration folder. The configuration file will be stored in the one of the following places:
 1. Windows: C:\VCACore\configuration
 2. Linux: /var/opt/VCA-Core/

18.4.6 Configuration

Under configuration, buttons to allow the management of VCAserver's configuration are provided:

- **Import Configuration:** Allows the import of a previously saved configuration file.
- **Export Configuration:** Exports the current configuration to a file `config.json`
- **Reset to Default Configuration:** removes all configured channels, rules, actions etc, resetting VCAserver to the default state



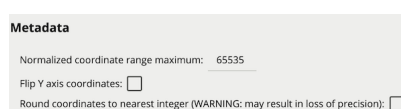
Current version information is also provided.

18.4.7 Metadata

VCAserver produces metadata accessible through various APIs but also through the action's token system. One aspect of that metadata is the **X** and **Y** coordinates for objects in a camera view.

Under the metadata section, the definition of aspects of this metadata can be specified:

- **Normalised coordinate range maximum:** The value representing the right hand side of the screen. For example can be set to 100 to signify location as a percentage of the field of view
- **Flip Y axis coordinates:** The ability specify the meaning of 0 in the **Y** axis, changing it from the top of the field of view to be the bottom.
- **Round coordinates to nearest integer:** The ability to round a given coordinate to the nearest whole number. This should be used with care especially in cases where the normalised coordinate range maximum has been set to a low value.



18.4.8 Digital Input

If digital inputs are available, the input sensors can be configured in two different modes:

- **Relay:** Enables the pull-up resistor, for use when inputs are connected to switches.
- **Voltage:** Disables the pull-up resistor.

Chapter 19

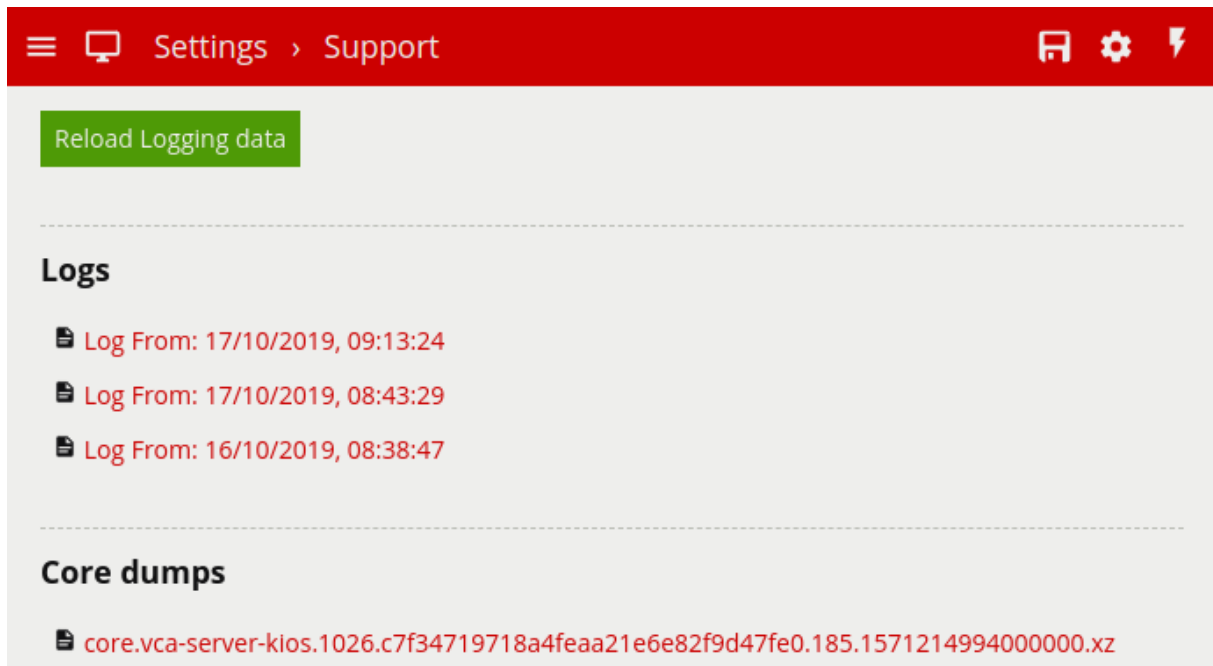
Support

The support page provides a repository for tools which can be utilised to help debug issues.

19.1 Logs

The Logs section provides a list of download links to the currently available logs. Logs are user-readable text files which log VCAserver messages. These logs can be submitted to the VCA Technology support staff to help resolve issues. New log files are created when VCAserver is started. If a log file reaches a certain size then it will be split into separate files.

The list of log files can be reloaded using the **Reload Logging data** button. Only a limited number of files can be stored with the oldest being replaced if that storage limit is met.



19.2 Core Dumps

The Core Dumps section provides a list of download links to the currently available core dumps. This additional debugging file is available for the VCAserver on Linux only. Core dumps can be uploaded to the VCA Technology support staff to provide more in depth system state information.

The list of core dump files can be reloaded using the **Reload Logging data** button. Only a limited number of files can be stored, with the oldest being replaced if that storage limit is met.

Chapter 20

Digital IO

VCAserver supports digital input and output hardware for interfacing with third party systems. Digital inputs can be used as triggers for events in VCAserver, and digital outputs can be triggered by a VCAserver or a system event.

Configuration of the digital inputs and outputs consists of three tasks:

- Configure the physical digital IO channels
- Assign physical digital input channels to logical **Sources**
- Assign physical digital output channels to logical **Actions**

20.1 Digital Output Device Configuration

From the **Settings Page** select **Edit Digital Outputs** to access the digital IO device configuration page.

Device Name:	DO 1	<
Device Name:	DO 2	<
Device Name:	DO 3	▼
Default State:	Normally Open	
Trigger Duration (ms):	3000	⦿
Test device		

The digital output device configuration page contains a section for each digital output device. Note that the number of digital output channels available depends on the specific hardware device in use.

20.1.1 Digital Outputs

Digital outputs can be triggered by a range of analytics event sources. Each digital output channel has the following properties:

- **Device Name:** The name of the digital output channel.
- **Default State:** The default state of the digital output when it's not triggered. Can be one of Normally Open or Normally Closed. When configured as Normally Open, the digital output will be open (low) when inactive and closed (high) when activated. When configured as Normally Closed, the behaviour is inverted. Refer to the table for more information.

Default State	DO Inactive	DO Active
Normally Open	Open (Low)	Closed (High)
Normally Closed	Closed (High)	Open (Low)

- **Trigger Duration(ms):** The duration of activation of the digital output in milliseconds. When the digital output is triggered, it will be activated for the specified duration, after which time it will be deactivated. Triggers received while the digital output is already active are ignored.
- **Test Device:** Clicking the button activates the digital output for the specified duration. This is useful to test that external devices are correctly connected to the digital output.

Once the digital output hardware has been configured, digital output hardware channels must be assigned to [Actions](#).

20.2 Sources and Actions

In order for digital IO channels to interact with VCAserver and system events, [Sources](#) must be created for digital inputs and [Actions](#) for digital outputs.

20.3 Digital Input Mode

See the [system settings page](#) for more digital input configuration options.

Chapter 21

Template Tokens

VCAserver can be set up to perform a specific action when an analytic event occurs. Examples include sending an email, TCP or HTTP message to a server.

VCAserver allows templated messages to be written for email, TCP and HTTP actions which are automatically filled in with the metadata for the event. This allows the details of the event to be specified in the message that the action sends, e.g. the location of the object, type of event, etc.

21.1 Syntax

The templating system uses `mustache`, which is widely used and [well-documented online](#). A brief overview of the templating syntax will be provided here. Templated messages can be written by using tokens in the message body. For example:

```
Hello {{name}}!
```

is a template with a `name` token. When the template is processed, the event metadata is checked to see if it has a `name` entry. If it does, the `{{name}}` token is replaced with the name of the event. If it isn't present, the token will be replaced with blank space.

If an event with the name `Presence` occurs, the processed template will be `Hello Presence!` but if it doesn't have a name, it will be `Hello !`

Some tokens may also have sub-properties which can be accessed as follows:

```
It happened at {{start.hours}}!
```

21.1.1 Conditionals

Tokens can also be evaluated as boolean values, allowing simple conditional statements to be written:

```
{{#some_property}}Hello, world!{{/some_property}}
```

In this example, if `some_property` is present in the event metadata, then "Hello, world!" will appear in the message. Otherwise, nothing will be added to the message.

If `some_property` is a boolean, then its value will determine whether or not the conditional is entered.
If `some_property` is an array property, it will only evaluate as true if the array is not empty.

21.1.2 Arrays

Finally, tokens can also be arrays which can be iterated over. For example:

```
{{#object_array}}  
{{name}} is here!  
{/object_array}}
```

This template will iterate through each item in `object_array` and print its name, if it has a `name` property. For example, the array `[{"name": "Bob"}, {"name": "Alice"}, {"name": "Charlie"}]` will result in the following output:

```
Bob is here!  
Alice is here!  
Charlie is here!
```

21.2 List of tokens

Lower case names represent tokens that can be used with the `{{token}}` syntax. Upper case names represent boolean or array properties that should be used with the `{{#token}}...{/token}}` syntax.

21.2.1 `{{name}}`

The name of the event

21.2.2 `{{id}}`

The unique id of the event

21.2.3 `{{type.string}}`

The type of the event. This is usually the type of rule that triggered the event

21.2.4 `{{type.name}}`

This is a boolean property that allows conditionals to be performed on the given type *name*.

For example, to print something only for events of type “presence”:

```
{{#type.presence}}My text{/type.presence}}
```


21.2.5 `{{start}}`

The start time of the event. It has the following sub-properties:

- `start.iso8601`
- `start.year`
- `start.month`
- `start.day`
- `start.hours`
- `start.minutes`
- `start.seconds`
- `start.milliseconds`
- `start.microseconds`
- `start.nanoseconds`
- `start.epoch`
- `start.offset.sign`
- `start.offset.hours`
- `start.offset.minutes`

The `iso8601` property is a date string in the ISO 8601 format.

The `offset` property is the time zone offset.

21.2.6 `{{end}}`

The end time of the event. Same properties as `{{start}}`

21.2.7 `{{host}}`

The hostname of the device that generated the event

21.2.8 `{{ip}}`

The IP address of the device that generated the event

21.2.9 `{{#Channel}}{{id}}{{/Channel}}`

The id of the channel that the event occurred on

21.2.10 `{{#Channel}}{{name}}{{/Channel}}`

The name of the channel that the event occurred on

21.2.11 `{{#Zone}}`

An array of the zones associated with the event. It has the following sub-properties:

- `id`: The id of the zone
- `name`: The name of the zone
- `channel`: The id of the channel the zone is attached to
- `colour`: The RGBA colour of the zone
- `detection`: 0 if the zone is non-detection zone, 1 otherwise
- `type`: 0 for a closed polygon, 1 for a line
- `outline`: The outline of the object (see the outline token for more details)

Example:

```
{{#Zone}}
id: {{id}}
name: {{name}}
channel:{{channel}}
colour: ({{colour.r}}, {{colour.g}}, {{colour.b}}, {{colour.a}})
{{/Zone}}
```

21.2.12 `{{#Object}}`

An array of the objects that triggered the event. It has the following sub-properties:

- `id`: The id of the object
- `outline`: The outline of the object (see the outline token for more details)
- `width`: The width of the bounding box based on the outline
- `height`: The height of the bounding box based on the outline

Example:

```
{{#Object}}
id: {{id}}
width: {{width}}
height: {{height}}
Top left corner: ({{outline.rect.top_left.x}}, {{outline.rect.top_left.y}})
{{/Object}}
```

21.2.13 `{{outline}}`

The bounding box outline of an object or zone. It has the following sub-properties:

- `outline.rect.top_left.x`: x-coordinate of the top left corner
- `outline.rect.top_left.y`: y-coordinate of the top left corner
- `outline.rect.bottom_right.x`: x-coordinate of the bottom right corner
- `outline.rect.bottom_right.y`: y-coordinate of the bottom right corner

Using a combination of these four coordinates, any corner of an object's bounding box can be obtained.

21.2.14 `{{#CountingLine}}`

An array of line counter counts. It has the following sub-properties:

- `rule_id`: The id of the line counter rule
- `width`: The calibration width of the line counter
- `position`: The position at which the object crossed the line
- `count`: The number of objects that crossed the line
- `direction`: The direction in which the object(s) crossed the line. 0 for A, 1 for B

Example:

```
{{#CountingLine}}
rule_id: {{rule_id}}
calibration width: {{width}}
position: {{position}}
count: {{count}}
direction: {{direction}}
{{/CountingLine}}
```

21.2.15 `{{#Counter}}`

An array of counter counts. It has the following sub-properties:

- `id`: The id of the counter
- `name`: The name of the counter
- `value`: The number of counts

Example:

```
{{#Counter}}
id: {{id}}
name: {{name}}
count: {{value}}
{{/Counter}}
```

21.2.16 `{{#Area}}`

The estimated area of the object. This token is a property of the object token. It is only produced if calibration is enabled. It has the following sub-properties:

- `value`: The estimated area of the object

Example:

```
{{#Object}}{{#Area}}{{value}}{{/Area}}{{/Object}}
```

21.2.17 `{{#Height}}`

The estimated height of the object. This token is a property of the object token. It is only produced if calibration is enabled. It has the following sub-properties:

- `value`: The estimated area of the object

Example:

```
{{#Object}}{{#Height}}{{value}}{{/Height}}{{/Object}}
```

21.2.18 `{{#GroundPoint}}`

The estimated position of the object. This token is a property of the object token. It is only produced if calibration is enabled. It has the following sub-properties:

- `value.x`: The estimated normalised x-axis position of the object
- `value.y`: The estimated normalised y-axis position of the object

Example:

```
{{#Object}}{{#GroundPoint}}Position: ({{value.x}}, {{value.y}}){{/GroundPoint}}{{/Object}}
```

21.2.19 `{{#Speed}}`

The estimated speed of the object. This token is a property of the object token. It is only produced if calibration is enabled. It has the following sub-properties:

- `value`: The estimated speed of the object

Example:

```
{{#Object}}{{#Speed}}{{value}}{{/Speed}}{{/Object}}
```

21.2.20 `{{#Classification}}`

The classification of the object. This token is a property of the object token. It is only produced if calibration is enabled. It has the following sub-properties:

- `value`: The classification of the object

Example:

```
{{#Object}}{{#Classification}}{{value}}{{/Classification}}{{/Object}}
```

21.2.21 `{{#DLClassification}}`

The classification generated by a deep learning model (e.g. Deep Learning Filter or Deep Learning Object Tracker). This token is a property of the object token. The algorithm must be enabled in order to produce this token, but calibration is not required. It has the following sub-properties:

- **class:** What the object has been classified as (person, vehicle)
- **confidence:** A value between 0.0 and 1.0 representing the confidence of the classification (0.0 least confident, 1.0 most confident)

Example:

```
{{#Object}}{{#DLClassification}}
Class: {{class}}
Confidence: {{confidence}}
{{/DLClassification}}{{/Object}}
```

21.2.22 {{#ColourSignature}}

The colour signature of the object. This token is a property of the object token. It is only produced if the colour signature algorithm is enabled. It has the following sub-properties:

- **colour_name:** Descriptive name of colour e.g. 'red'
- **proportion:** The percentage of an object that is made up of the specified colour

Example:

```
{{#Object}}{{#ColourSignature}}
{{#colours}}
Colour: {{colour_name}}, Proportion: {{proportion}}
{{/colours}}
{{/ColourSignature}}{{/Object}}
```

21.3 Examples

The following is an example of a template using most of the available tokens:

```
Event #{{id}}: {{name}}
Event type: {{type}}
Start time (ISO 8601 format): {{start.iso8601}}
End time:
day: {{end.day}}
time: {{end.hour}}:{{end.minutes}}:{{end.seconds}}.{{end.microseconds}}
Device: {{host}}
Channel: {{#Channel}}{{id}}{{/Channel}}
{{#type.presence}}
{{#Object}}
Object ID: {{id}}
{{#Classification}}Object Classification: {{value}}{{/Classification}}
{{#Height}}Object Height: {{value}}m{{/Height}}
Object bounding box: [
  ({{outline.rect.top_left.x}}, {{outline.rect.top_left.y}}),
  ({{outline.rect.bottom_right.x}}, {{outline.rect.top_left.y}}),
  ({{outline.rect.bottom_right.x}}, {{outline.rect.bottom_right.y}}),
  ({{outline.rect.top_left.x}}, {{outline.rect.bottom_right.y}})
```

```

]
{{/Object}}
{{/type.presence}}

{{#Counter}}
Counter triggered.
id: {{id}}
name: {{name}}
count: {{count}}
{{/Counter}}

{{#LineCounter}}
rule_id: {{rule_id}}
calibration width: {{width}}
position: {{position}}
count: {{count}}
direction: {{direction}}
{{/LineCounter}}

```

In this example, the object information is only printed for events of type “presence”.

This template might result in the following message:

```

Event #350: My Bad Event
Event type: presence
Start time (ISO 8601 format): 2017-04-21T10:09:42+00:00
End time:
day: 21
time: 10:09:42.123456
Device: mysecretdevice
Channel: 0

```

```

Object ID: 1
Object Classification: Person
Object Height: 1.8m
Object bounding box: [
  (16000, 30000),
  (32000, 30000),
  (32000, 0),
  (16000, 0)
]

```

```

Counter triggered.
id: 10
name: My Counter
count: 1

```

```

rule_id: 350
calibration width: 1
position: 1

```

count: 1
direction: 0

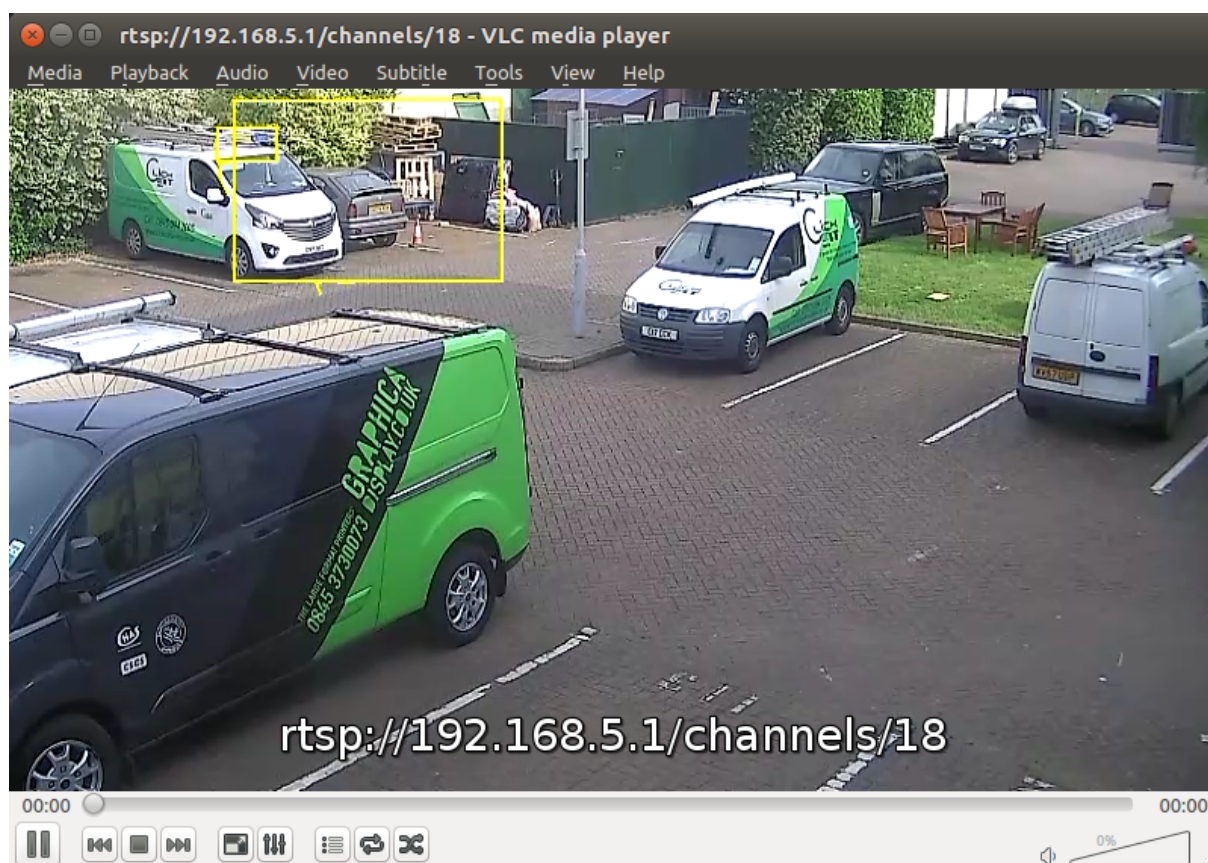
Chapter 22

RTSP Server

VCAserver supports an RTSP server that streams annotated video in RTSP format.

The RTSP URL for channels on a VCA device is as follows:

```
rtsp://\<device ip>:8554/channels/\<channel id>
```



Chapter 23

Sureview Immix



VCAserver supports the notification of events with annotated snapshots and streaming of real-time annotated video to Sureview Immix.

23.1 Prerequisites

The following ports need to be accessible on the VCA device (i.e. a VCAbridge or an instance of VCAserver) from the Immix server:

- 8080: TCP web requests
- 8554: TCP/UDP annotated RTSP video stream

23.2 Limitations

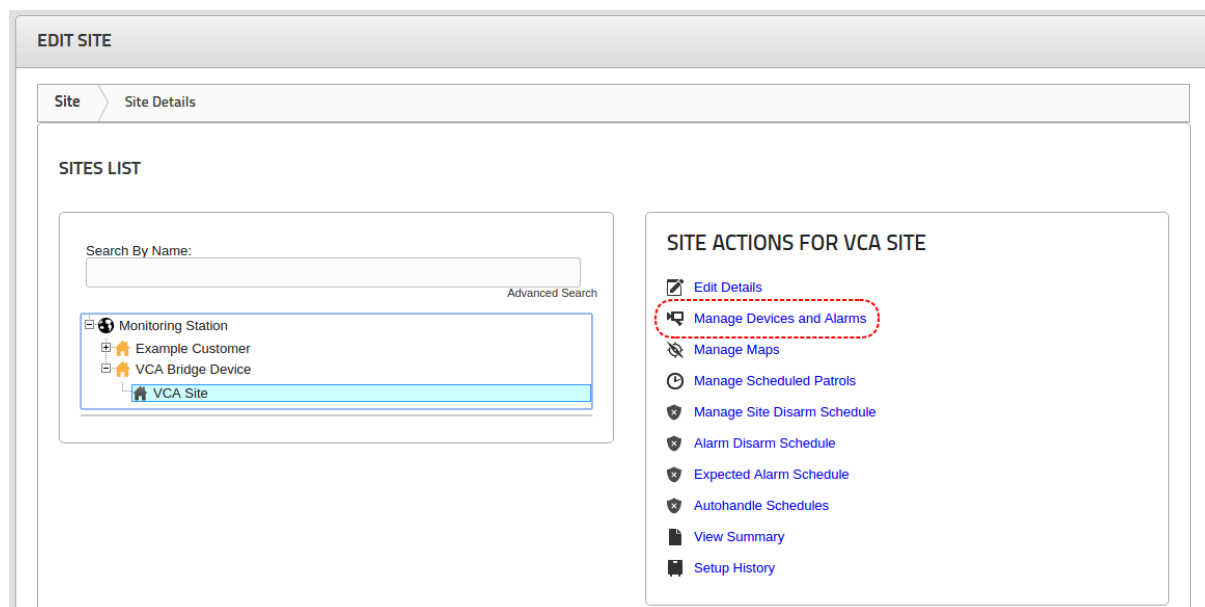
- Only one camera is supported per VCA device. This means that a device has to be configured in Immix for each channel in an instance of VCAserver. E.g. if a VCAserver has 8 channels, the Immix configuration must consist of 8 devices, each with one channel.

23.3 Immix Configuration

23.3.1 Add VCA Device

The first step is to add the VCA device.

In the Immix site configuration tab, click **Manage Devices and Alarms**, then **Add Device**:



On the Add Device page, set the following options:

DEVICES **CAMERAS** **MULTIVIEWS** **SPLITS** **TOURS** **AUDIOS** **RELAYS** **ALARMS** **SUMMARY**

DEVICE DETAILS

Device Type Filter

☒ Video Devices
☐ Alarm Panel
☐ Access Control
☐ Show All

Device Type *

VCA Bridge

Title *

VCA Bridge

CONNECTION DETAILS

Used to connect to the device for monitoring. Obtain these details from the person who installed the device.

IP/Host

www.vcabox.com

Port

9555

Ports must only contain numeric values.

Username

Password

- **Device Type Filter:** Select **Video Devices**
- **Device Type:** Select **VCA Bridge**
- **Title:** Give the device a suitable name
- **IP/Host:** Enter the IP address or hostname of the VCA device
- **Port:** Enter the RTSP port of the VCA device. In this case the device is behind a firewall and port 9555 on www.vcabox.com is forwarded to the standard RTSP port (554) on the VCA device. The RTSP server on the VCA device runs on the standard RTSP port (554).

23.3.2 Add Camera

Once the device has been added, channels from the VCA device can be added.



Note: Immix currently supports only one VCA channel per device. To support more channels, simply add more devices.

Click the **Cameras** tab and **Add a Camera** to add a new channel:

Devices
Cameras
Multiviews
Splits
Tours
Audios
Relays
Alarms
Summary

CURRENT CAMERAS

PREVIOUSNEXT

REFERENCE IMAGE	CAMERA NAME	DEVICE NAME	INPUT	QUALITY	
	VCA Bridge camera (1)	VCA Bridge	1	Default	<div> <div></div> <div></div> <div></div> </div> <div>  Add a Camera </div>

On the **Camera Details** page set the following options:

Devices
Cameras
Multiviews
Splits
Tours
Audios
Relays
Alarms
Summary

CAMERA DETAILS

Select Device
VCA Bridge (VCA Bridge)


Input *
1

Camera Name *
VCA Bridge camera (1)

Camera Quality
Default

Streaming Resolution
Default (640x480)

REFERENCE SHOT

Reference Shot 1


ADD REFERENCE SHOT

- **Input:** Enter the VCA channel Id + 1 (see below for more details)
- **Camera Name:** Enter a suitable name for the channel
- Leave the other settings with default values

23.3.3 Setting the Input in Immix

In order to set the Input value correctly in Immix, the following steps should be followed:

- Find the channel Id in VCA. The channel Id is displayed at the bottom of the **Channel Settings Menu** on the **Channel Page**:

CHANNEL-ID

- Set the value of the **Input** field in Immix to the channel Id in VCA + 1, i.e. as illustrated in the table below:

Channel Id in VCA	Input in Immix
0	1
1	2

Channel Id in VCA	Input in Immix
2	3
5	6
100	101

The reason that the Immix Input is 1 higher than the VCA channel Id is that Immix uses one-based inputs but VCA uses zero-based channel Ids.

23.3.4 Retrieve a Summary

Generating a summary provides a single document with all of the details necessary to configure the VCA device. Click the **Summary** tab and a PDF report is created:

Setup Report for VCA Site

VCA Bridge Channel 1 (VCA Bridge)

Identifier: 22

SMTP Server Address: S22@ImmixAlarms.com

IP Address: www.vcabox.com

Port: 9555

Cameras:

Input: 1, Name: VCA Bridge Channel 1 camera (1)

VCA Bridge Channel 2 (VCA Bridge)

Identifier: 23

SMTP Server Address: S23@ImmixAlarms.com

IP Address: www.vcabox.com

Port: 9555

Cameras:

Input: 1, Name: VCA Bridge Channel 2 camera (1)

Make a note of the email addresses highlighted in red. These email addresses need to be entered in the VCA device configuration (see next section).

23.4 VCAserver Configuration

Once a device and camera are configured in Immix, the email addresses generated as part of the summary need to be added to the VCAserver configuration.

VCAserver notifies Immix of events via email, so each channel configured for Immix needs to have an email action configured. For more details on how to configure **Actions** or **Sources** see the corresponding topics.

23.4.1 Add an Email Action

Add an **Email** action with the following configuration:

- **Body Format:** Set to Sureview Immix
- **Server:** Enter the IP address or hostname of the Immix server
- **Port:** Enter the port of the Immix server (Default 25)
- **To:** Enter the email address generated in the Immix summary report. Here S22@ImmixAlarms.com corresponds to Immix Input 1 (which is VCA channel Id 0).

Once this is done, add the correct source to the email action.

23.4.2 Event Type Mappings

The event types reported in the VCAserver interface are slightly different to the event types reported in the Immix client. The events are mapped as follows:

Event in VCA	Event in Immix
Presence	Object Detected
Enter	Object Entered
Exit	Object Exited
Appear	Object Appeared
Disappear	Object Disappeared
Stopped	Object Stopped
Dwell	Object Dwell
Direction	Object Direction
Speed	Object Speed
Tailgating	Tailgating
Tamper	Tamper Alarm

Chapter 24

ONVIF support

VCAserver has inbuilt support for a subset of ONVIF profile S endpoints. To date, these provide the following functions using the ONVIF interface:

- Discovery (Device information, supported services)
- Events (the last 100 actions the device has sent)
- ONVIF User Management (change ONVIF password)

More detail on each ONVIF function is given below, screenshots are provided using the ONVIF Device Manager implementation varies application to application.

Note: ONVIF Device Manager is a third-party, open source windows application available at [ONVIF Device Manager](#)

24.1 Discovery


ONVIF device discovery retrieves information about the ONVIF enabled device including the following data:

- **Name:** Device name comprised of the VCAserver platform and the IP address.
- **Address:** The IP address of the host system.
- **Location:** Universally set to global.
- **Manufacturer:** VCA Technology
- **Firmware:** Current version of VCAserver.

You logged in as **admin** [Log out](#)

Device list <<
 [Cancel](#)

VCABridge-192.168.1.34



[Identification](#)
[Time settings](#)
[Maintenance](#)
[Network settings](#)
[User management](#)
[Web page](#)
[Events](#)

Identification

Name
Location
Manufacturer
Model
Hardware
Firmware
Device ID
IP address
MAC address
ONVIF version
URI:

[Apply](#) [Cancel](#)

Device list

Name, location or address [Cancel](#)

IPN302HD-5117
Firmware
Address
Location

VCABridge-192.168.1.34
Firmware
Address
Location

General
Firmware
Address
Location

IPN2103HD-6611
Firmware
Address
Location

The above image shows the ONVIF Device Manager's Identification interface with a VCAbridge running on 192.168.1.34

24.2 Events

The ONVIF events service allows a third-party application to pull a list of events from the VCAserver platform. When a pull request is made the last 100 events triggered within VCAserver are returned. An event is defined as any logical rule (with **Can Trigger Actions** enabled) or **Other Source** (such as interval or DI) which triggers. Importantly, neither the logical rule nor the Other Source has to be configured with an action to be included within ONVIF event service cache.

The following fields are currently included for each event.

Property	Description
start_time	The start time of the event
end_time	The end time of the event
id	The id of the event
name	The user-specified name of the event
type	The type of the event
category	The category of the event

- The start_time and end_time properties of the events are specified as nanoseconds since the Unix epoch.

You logged in as **admin** [Log out](#)

Device list <<
 [Cancel](#)

VCABridge-192.168.1.34

[Identification](#)
[Time settings](#)
[Maintenance](#)
[Network settings](#)
[User management](#)
[Web page](#)
[Events](#)

Events

Arrival time	Topic	Operation	Source	Key	Data
14:27:01		Initialized			start_time: 1562250421528926542 end_time: 1562250421528926542 id: 2054 name: Presence 4 type: presence category: analytics
14:26:59		Initialized			start_time: 1562250419608227307 end_time: 1562250421655370768 id: 2053 name: Direction 5 1142 type: direction category: analytics
14:27:01		Initialized			start_time: 1562250421528926542 end_time: 1562250421688040293 id: 2054 name: Presence 4 type: presence category: analytics
14:26:55		Initialized			start_time: 1562250415930012938 end_time: 1562250421688040293 id: 2052 name: Presence 4 type: presence category: analytics

The above image shows the ONVIF Device Managers Events interface with a VCABridge running on 192.168.1.34 where the data component of each event is populated with the above properties.

24.3 ONVIF User Management

Some ONVIF functions, such as the events, are secured using user credentials.

The default credentials for ONVIF access are:

- **username:** admin
- **password:** admin

Please note that these credentials are separate from the VCAserver platform, changing one has no impact on the other. The ONVIF password can only be changed using the ONVIF user management. The process for this will vary depending on the ONVIF implementation.

Only a single ONVIF user, with username `admin` is supported within VCAserver.

You logged in as **admin** [Log out](#)

Device list <<
 [Cancel](#)

VCABridge-192.168.1.34

[Identification](#)
[Time settings](#)
[Maintenance](#)
[Network settings](#)
[User management](#)
[Web page](#)
[Events](#)

User management

Policies: [Backup](#) [Restore](#)

Users:

Name: admin
Password:
Role: administrator

[Create](#) [Modify](#) [Delete](#)

If you require more information on ONVIF profiles please refer to the [ONVIF documentation](#).

Chapter 25

Verifier

VCAserver supports the notification of events with annotated snapshots to Verifier.

25.1 Prerequisites

A Verifier account, please contact your VCA sales representative for details.

25.2 VCAserver Configuration

Once you have a Verifier account setup, an email action using the Verifier template must be configured for VCAserver to send data to Verifier. See below (or [Actions](#)) for more details on how to configure a Verifier email action within VCAserver.

25.2.1 Add an Email Action

Add an **Email** action with the following configuration:

- **Body Format:** Set to Verifier
- **Server:** Enter the IP address or hostname provided by Verifier
- **Port:** Enter the port of the server provided by Verifier (Default 25)
- **To:** Enter the email address generated on the Verifier dashboard

Once this is done, add the correct source to the email action.

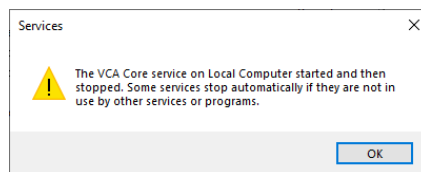
Chapter 26

Troubleshooting

26.1 Error when starting the VCAcore service on Windows 10

26.1.1 Issue

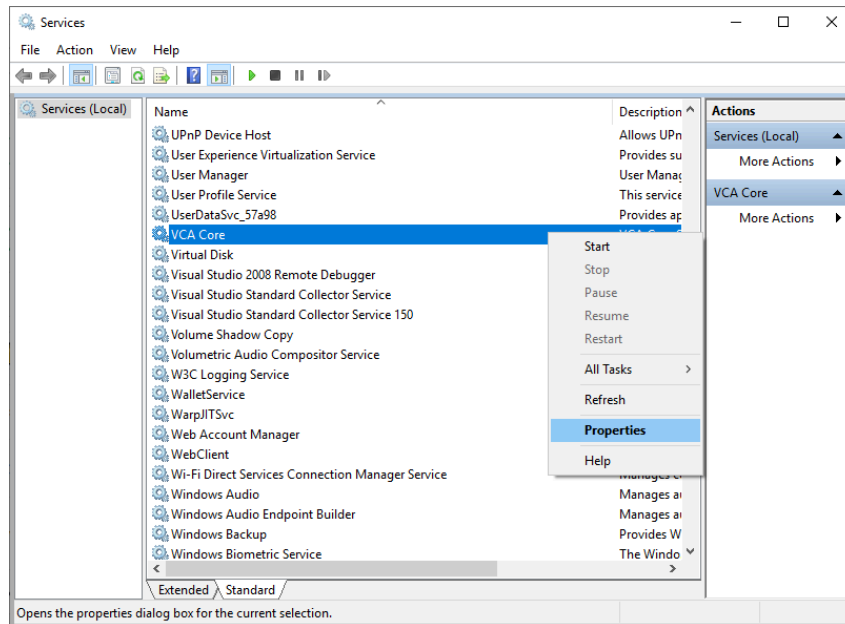
When starting VCAserver as a service you get an error similar to this:



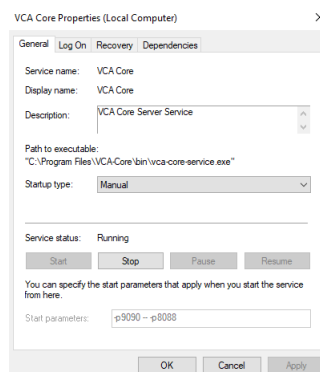
26.1.2 Solution

The most common cause of this error is one of the ports VCAserver is trying to bind to is already in use by another service or application. To fix this you can manually specify different ports:

- Open the services window
- Find the VCA Core service listing, right click and click properties



- In the Start Parameters field insert `-p 9090 -- -p8080` where 9090 is the recovery port and 8080 is the web port you want VCAserver to bind to. Once defined, click the Start button to start the service with these settings.



After doing this once, the service will use these ports going forward.

Another useful Start Parameter to be aware of allows the definition of the port for VCAcore's RTSP server. In this case insert `--rtsp-port 1234` in the the Start Parameters field, where 1234 is the desired port for VCAcore's RTSP server. This can be done at the same time as the web server port.

26.2 Error when starting the VCAcore service on Ubuntu

26.2.1 Issue

When starting VCAserver as a service, the application does not start and the recovery service shows it failed to start.

26.2.2 Solution

The most common cause of this error is one of the ports VCAserver is trying to bind to is already in use by another service or application. To fix this you can manually specify a different port:

1. From a terminal, type `sudo systemctl edit vca-core --full` (other editing tools are available)
2. At the end of the line `ExecStart=/opt/VCA-Core/bin/vca-core-service -p 9090 -- add -p 1234` (where **1234** is the port you want to use for web access) Example:
`ExecStart=/opt/VCA-Core/bin/vca-core-service -p 9090 -- -p8081)`
3. Save and restart the service `sudo systemctl restart vca-core`

26.3 Error when starting vca-cored application on Ubuntu

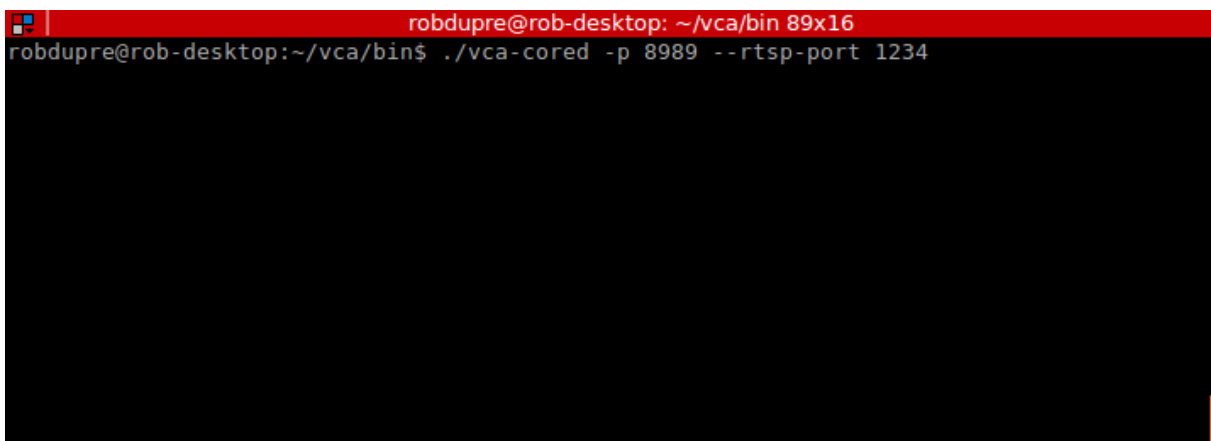
26.3.1 Issue

When starting VCAserver the following message is displayed `Error attaching rtsp server`

26.3.2 Solution

Typically if the VCAserver application is not starting it is due to a port conflict where both VCAserver and another application are trying to bind to the same port. This can be either the web service serving the VCAserver UI or the RTSP Server. To fix this you can specify different ports manually when starting the application. (`-p` for the web server, `--rtsp-port` for the RTSP server). Once VCAserver successfully starts with these parameters, the configuration file will persist these changes, negating the need to specify the parameters each time VCAserver is started.

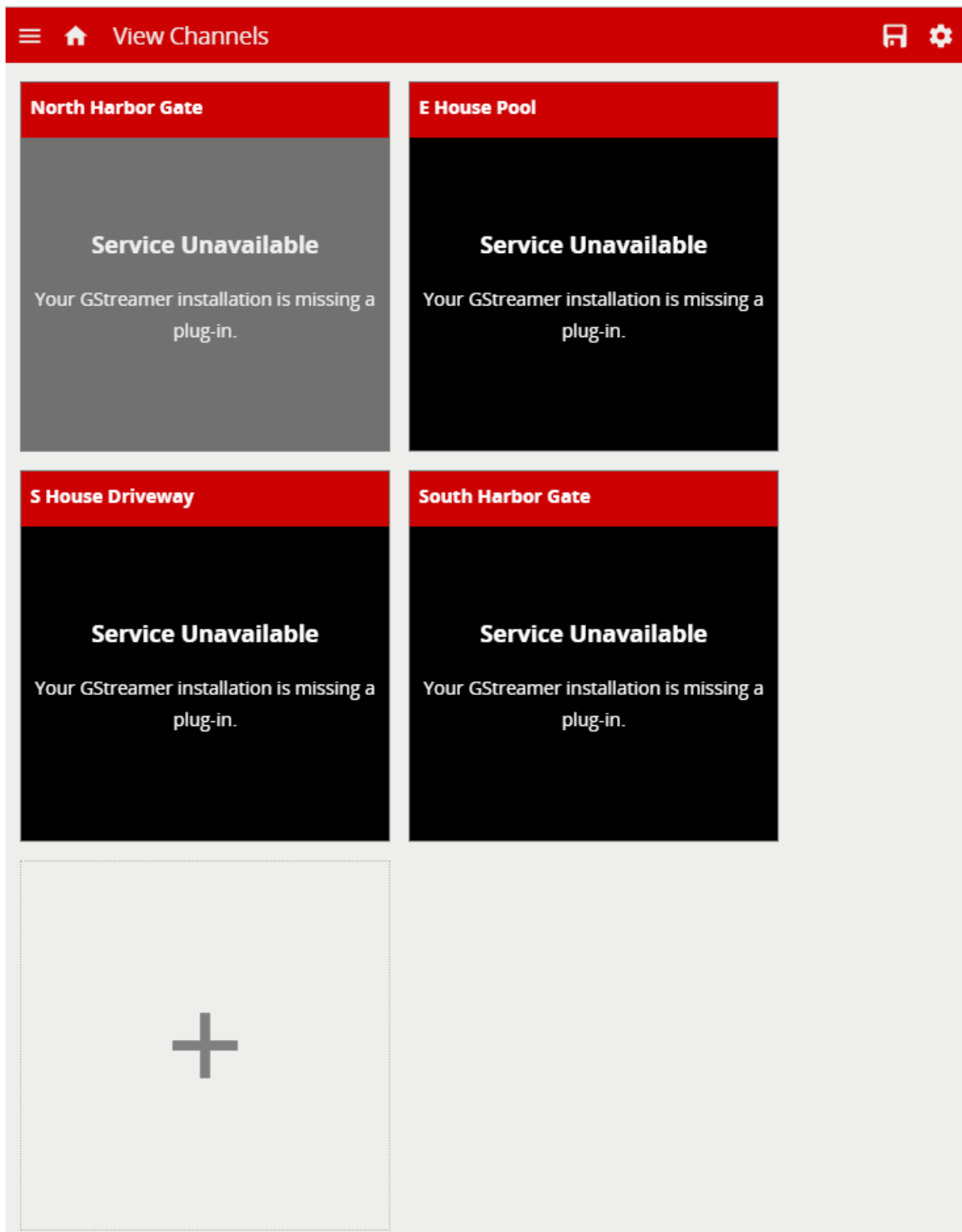
```
./vca-cored -p 8989 --rtsp-port 1234
```

A terminal window with a red title bar. The title bar text is "robdupre@rob-desktop: ~/vca/bin 89x16". The terminal content shows the command "robdupre@rob-desktop:~/vca/bin\$./vca-cored -p 8989 --rtsp-port 1234" being entered. The rest of the terminal area is black, indicating the command is running or the output is not visible.

26.4 Service Unavailable, Your GStreamer installation is missing a plug-in

26.4.1 Issue

When adding a source to VCAserver you are presented with the following message in the View Sources page:



26.4.2 Solution

This error will be presented when the source input into VCAserver is encoded with an unsupported codec. Please ensure that the source being added is encoded with one of VCAserver's [Supported Video](#)

Sources