

# Enforce Box User Manual



This manual contains instructions on accessing the web interface, system settings and setup guidelines, and usage and maintenance.

# Enforce Box

## USER MANUAL

Document version: 2023.05.30.  
Firmware version: 1.4.0.208

### Table of Contents

1. Overview.....	4
2. AR Device Tool .....	5
2.1. Finding Devices/Cameras .....	5
2.2. Firmware, License, and Engine Upload - Manually .....	6
2.3. Firmware and Engine – Checking for Updates.....	7
3. Overview of the Web Interface.....	8
4. Live .....	9
4.1. Full-Screen Mode .....	9
4.2. Saving image.....	10
4.3. Switching Stream.....	10
4.4. Help.....	10
4.5. Overlay .....	11
4.6. Event Preview.....	12
5. Playback.....	13
5.1. Navigate among the Recordings.....	13
5.2. Filtering the Detectors .....	15
5.3. Exporting the Recordings .....	15
6. Events.....	16
7. Settings .....	19
7.1. System / Status.....	19
7.2. System / Device .....	20



7.3.	SyStem / Network .....	22
7.4.	System / Security .....	23
7.5.	System / Storage.....	25
7.6.	I/O .....	30
7.7.	System / Service .....	32
7.8.	System / Notifications.....	33
7.9.	External.....	34
7.10.	Media / Video.....	35
7.11.	Analytics / Settings .....	38
7.12.	Analytics / Detectors.....	38
8.	How to use the enforce Box .....	58
8.1.	Device Installation .....	58
8.2.	Red Stop detector and network settings .....	59
9.	API Documentation .....	61
9.1	Introduction.....	61
9.2	Authentication.....	62
9.3	Executing commands .....	64
9.4	Data types .....	66
9.5	Command options.....	68
9.6	Features .....	69
10.	Detectors & Engines.....	70
10.1.	Types.....	70
10.2.	Geometry.....	72
11.	Events.....	73
11.1.	Modes .....	73
11.2.	Live event query .....	74
11.3.	Live event stream.....	75
11.4.	Stored event query .....	79
11.5.	Stored event upload .....	80
12.	Miscellaneous.....	83
12.1.	GPIO state stream.....	83
13.	Reference.....	85
13.1	Analytics.....	85
13.2	Storage.....	96
13.3	System.....	97
13.4	Structs.....	110
	CONTACT INFORMATION .....	246



## 1. OVERVIEW

The Enforce Box device has its own web interface through which you can access the settings, the **LIVE**, the **PLAYBACK**, the **EVENTS** and the **SETTINGS** interfaces.

### Note

It is recommended to use an up-to-date web browser to access the web interface.

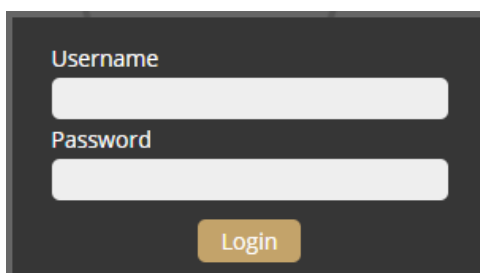
Accessing the web interface:

1. Start a browser and enter the device IP address into the address bar of the browser.
2. Type the username and the password on the displayed login interface and click on **[Login]**.

The default user account is the following:

**Username:** admin

**Password:** admin

A screenshot of a web login interface. It features a dark background with two white input fields. The first field is labeled 'Username' and the second is labeled 'Password'. Below these fields is a yellow button with the text 'Login' in black.

If you cannot connect to the camera's web interface, please refer to section 3.1 of the Installation Guide.



## 2. AR DEVICE TOOL

With the AR DeviceTool, you can discover Einar or Visus cameras, Carmen Box or Carmen Nano, or Enforce Box devices on the local network. You can upload Firmware, License and Engine files to these cameras/devices. Download the program here: [AR DeviceTool](#).

Upload firmware	Upload license	Upload engine	Check for updates				
Name	Family	Version	Firmware	IP Address	MAC address	Upload information	
IDEV-TRAFFIC	CBOX	Photon	1.0.0.2040	192.168.6.143	00:04:4b:e9:dc:20		
RIO - CarmenBox	CBOX	Photon	1.1.0.213	192.168.6.89	48:b0:2d:3d:f2:04		
ICAM-D7D6	EINAR	Einar-5	2.1.1.3	192.168.6.197	00:19:b4:02:d7:d6		
Einar	EINAR	Einar-5	2.1.1.3	192.168.6.82	00:19:b4:00:d2:5b		
Einar-PZS-T	EINAR	Einar-ST	2.1.1.3	192.168.6.240	00:19:b4:00:d2:23		

Devices: 5      Offline: 0

### 2.1. FINDING DEVICES/CAMERAS

Once started, the program lists the AR devices/cameras detected on the local network if the devices/cameras are in the same network segment as the computer. The device/camera name, product family name, type, firmware version, IP address, MAC address and brief information about the current upload process will be displayed.

The currently available devices/cameras are marked with green color in the first column.

The red color indicates a previously discovered device/camera that has not been available since then. If newer firmware or engine are available for any of the listed cameras/devices, a star sign is added into the green indicator.






Double-click on the selected device/camera to open its web interface in the default browser.

Upload firmware	Upload license	Upload engine	Check for updates				
Name	Family	Version	Firmware	IP Address	MAC address	Upload information	
ICAM-D209	EINAR	Einar-5	1.9.0.49	192.168.100.18	00:19:b4:02:d2:09		
Cam74	VISUS	ILD-420E-BL-IR	v4.6.0 (build 12)	192.168.100.74	00:19:b4:01:84:67		
Cam75	VISUS	ILD-420E-BL-IR	v4.6.1 (build 10)	192.168.100.75	00:19:b4:01:7f:cf		

Devices: 3      Offline: 1

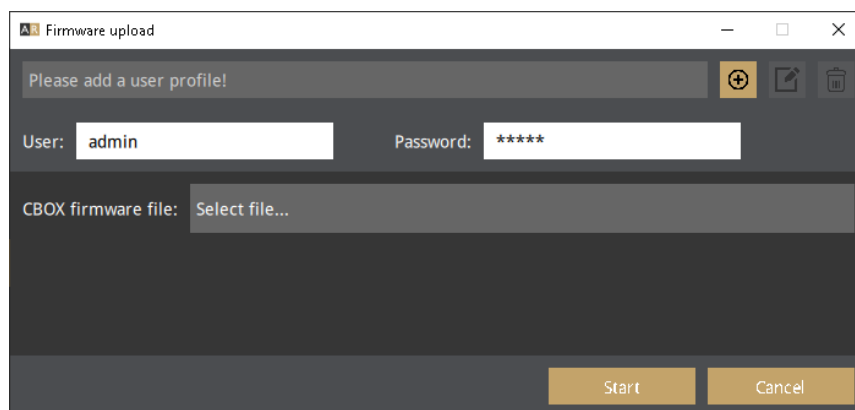
## 2.2. FIRMWARE, LICENSE, AND ENGINE UPLOAD - MANUALLY

In addition to find devices/cameras, you can also use the AR DeviceTool to upload Firmware, License or even Engine for the selected single camera/device or a group of cameras/devices using Ctrl/Shift. The License file is unique for each device/camera, therefore it cannot be uploaded in groups.

Upload firmware	Upload license	Upload engine	Check for updates			
Name	Family	Version	Firmware	IP Address	MAC address	Upload information
 IDEV-TRAFFIC	CBOX	Photon	1.0.0.2040	192.168.6.143	00:04:4b:e9:dc:20	
 ICAM-D7D6	EINAR	Einar-5	2.1.1.3	192.168.6.197	00:19:b4:02:d7:d6	
 Einar-PZS-T	EINAR	Einar-5T	2.1.1.3	192.168.6.240	00:19:b4:00:d2:23	
 Einar	EINAR	Einar-5	2.1.1.3	192.168.6.82	00:19:b4:00:d2:5b	
 RIO - CarmenBox	CBOX	Photon	1.1.0.213	192.168.6.89	48:b0:2d:3d:f2:04	
Devices: 5      Offline: 0						

Select the device(s)/camera(s) you want to update and press the **[Upload firmware]**, **[Upload License]** or **[Upload engine]** buttons that become active.

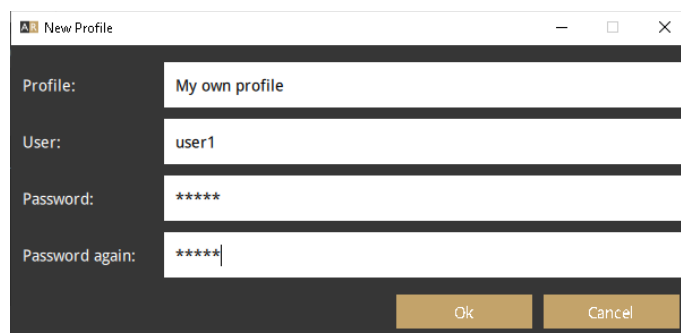
In the window that appears, enter the username and password to access the camera, select the file you want to upload and click **[Start]** to start the upload.



If you want to save the username and password to access the device(s)/camera(s), you can create user profiles. This way, you don't have to enter credentials before each upload.

Click on **[Please add a user profile!]** or the **[+]** button and enter the required information. For further uploads, you will only need to select the user profile.

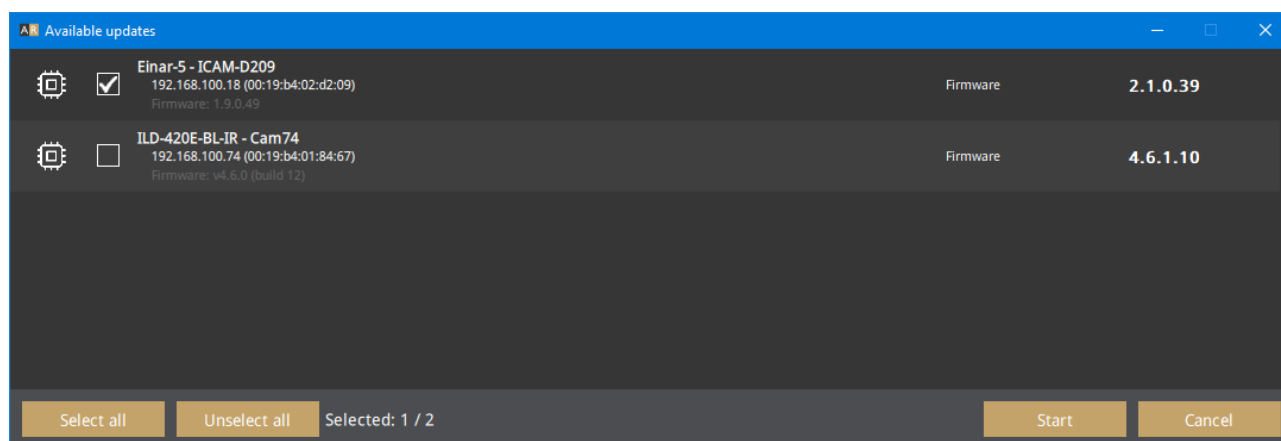
Previously created user profiles can be edited or deleted using the buttons next to the **[+]** button.



A dialog box titled "New Profile" with a dark background. It contains four input fields: "Profile:" with the text "My own profile", "User:" with the text "user1", "Password:" with masked text "\*\*\*\*\*", and "Password again:" with masked text "\*\*\*\*\*". At the bottom right are two buttons: "Ok" and "Cancel".

## 2.3. FIRMWARE AND ENGINE – CHECKING FOR UPDATES

If your PC is connected to the internet, you can check if newer firmware or engine are available for any of your cameras/devices. Press **[Check for updates]** button, select device(s)/camera(s) you want to update and press **[Start]**. Confirmation and credentials must be provided.



A dialog box titled "Available updates" with a blue header bar. It displays a list of two devices with their respective firmware versions. The first device, "Einar-5 - ICAM-D209", has its update checkbox checked. The second device, "ILD-420E-BL-IR - Cam74", has its update checkbox unchecked. At the bottom, there are buttons for "Select all", "Unselect all", and "Start", along with a status indicator "Selected: 1 / 2".

Device	IP Address	MAC Address	Firmware	Update Status
Einar-5 - ICAM-D209	192.168.100.18	(00:19:b4:02:d2:09)	2.1.0.39	<input checked="" type="checkbox"/>
ILD-420E-BL-IR - Cam74	192.168.100.74	(00:19:b4:01:84:67)	4.6.1.10	<input type="checkbox"/>

AR DeviceTool downloads the appropriate firmware and/or engine from a central server, and uploads it to the selected device(s)/camera(s). A new folder will be created in your Download folder: ArDeviceToolDownloads. Please delete it if you no longer need the firmware(s)/engine(s).

### 3. OVERVIEW OF THE WEB INTERFACE

The following menu items are available on the web interface:



#### 1. LIVE

Shows a live view of the connected camera streams.

#### 2. PLAYBACK

Browse recordings on the configured storage device.

#### 3. EVENTS

Browse the recorded events on the configured storage device.

#### 4. SETTINGS

Under this menu, you can access the following options:

#### SYSTEM

- Status
- Device
- Network
- Security
- Storage
- I/O
- Service
- Notifications
- External

#### MEDIA

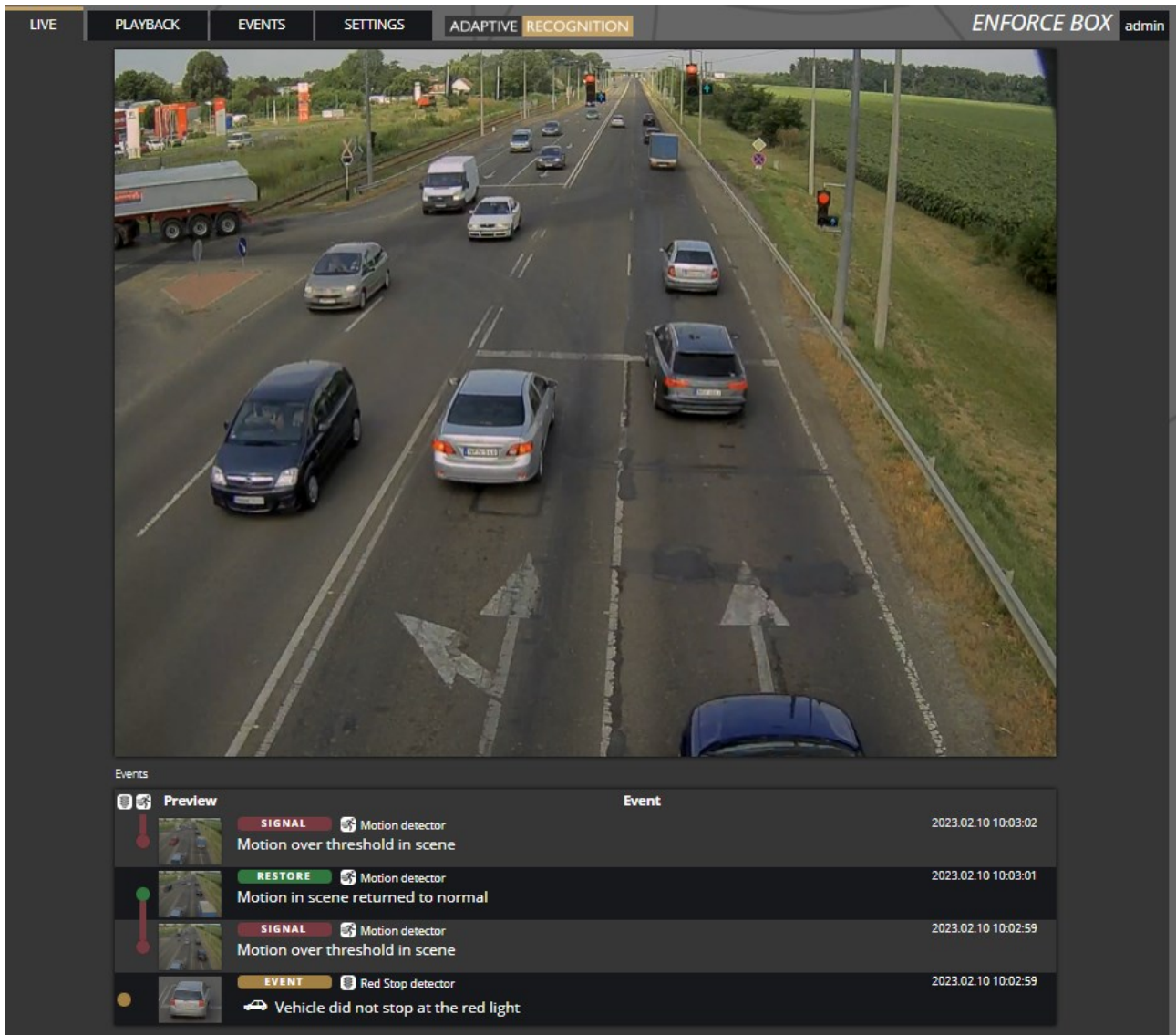
- Video

#### ANALYTICS

- Settings
- Detectors

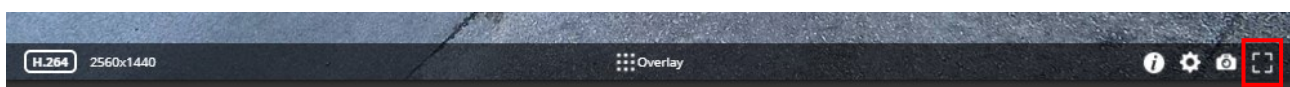
## 4. LIVE

After login, the interface navigates to the LIVE tab that shows a live feed of the connected camera stream.



### 4.1. FULL-SCREEN MODE

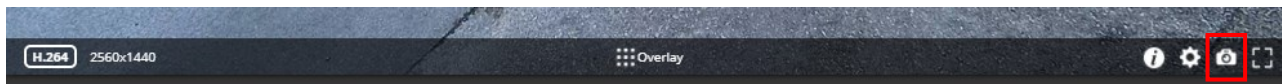
The camera's live stream can be displayed on full screen by clicking on the icon located in the bottom-right corner of the image.



To exit from the full-screen mode, press the **ESC** keyboard key or click on the icon mentioned above.

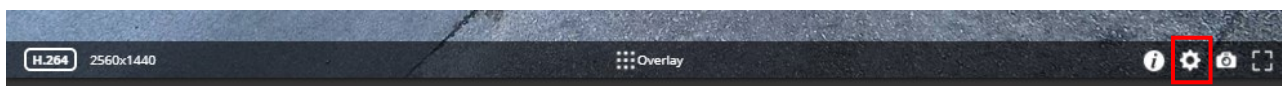
## 4.2. SAVING IMAGE

Next to the [Full-screen] icon is the [Save image] icon. By clicking on it, you can save an image of the current live stream with previously selected OSD information. The **CTRL + S** keyboard shortcut can be used as well.



## 4.3. SWITCHING STREAM

The [Streams] button is located next to the [Save image] icon. By clicking on it, you can select which stream will be displayed as LIVE.



## 4.4. HELP

Next to the [Streams] icon is the [Help] button. It brings up keyboard shortcuts on how to use and navigate the video feed. To exit from the Help OSD, press the [Help] button or click in the grey area.

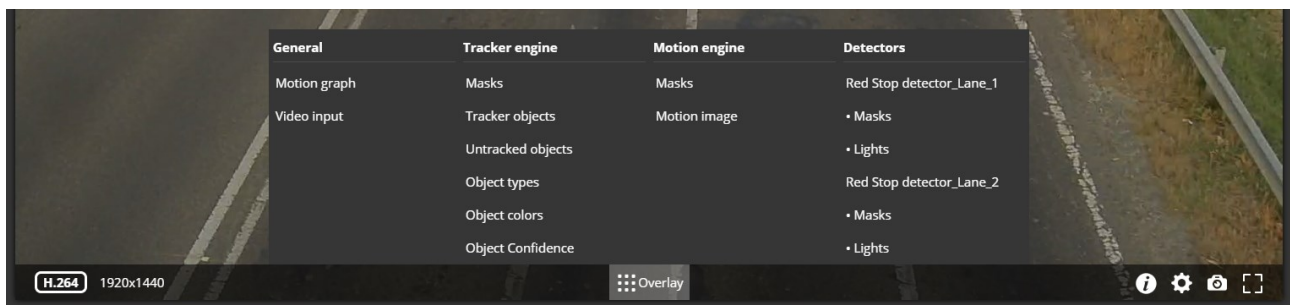


## 4.5. OVERLAY

In the middle, at the bottom of the window, is the **[Overlay]** button. With it, you can turn on/off the OSD, and you can view the masks of the applied detectors, image information, motion data, etc.

The overlay can be displayed in LIVE and PLAYBACK mode, as well as in any submenu of the SETTINGS menu where the video stream is visible.

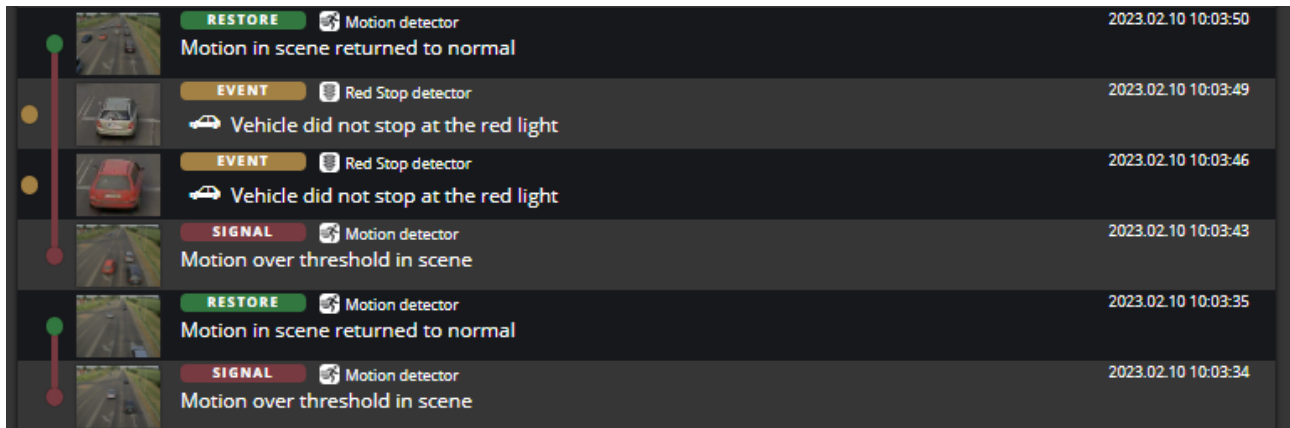
The OSD layers come in handy for observing the internal workflow of the device, setting up the device or troubleshooting.















## 4.6. EVENT PREVIEW

You can find the event preview section under the live stream image, displaying the notifications about the latest received events.

Basic events are shown with a tan color and the "EVENT" text. A dark red colored "SIGNAL" text indicates a start of a longer event that lasts for multiple frames. A long end of event is marked with the green "RESTORE" text.



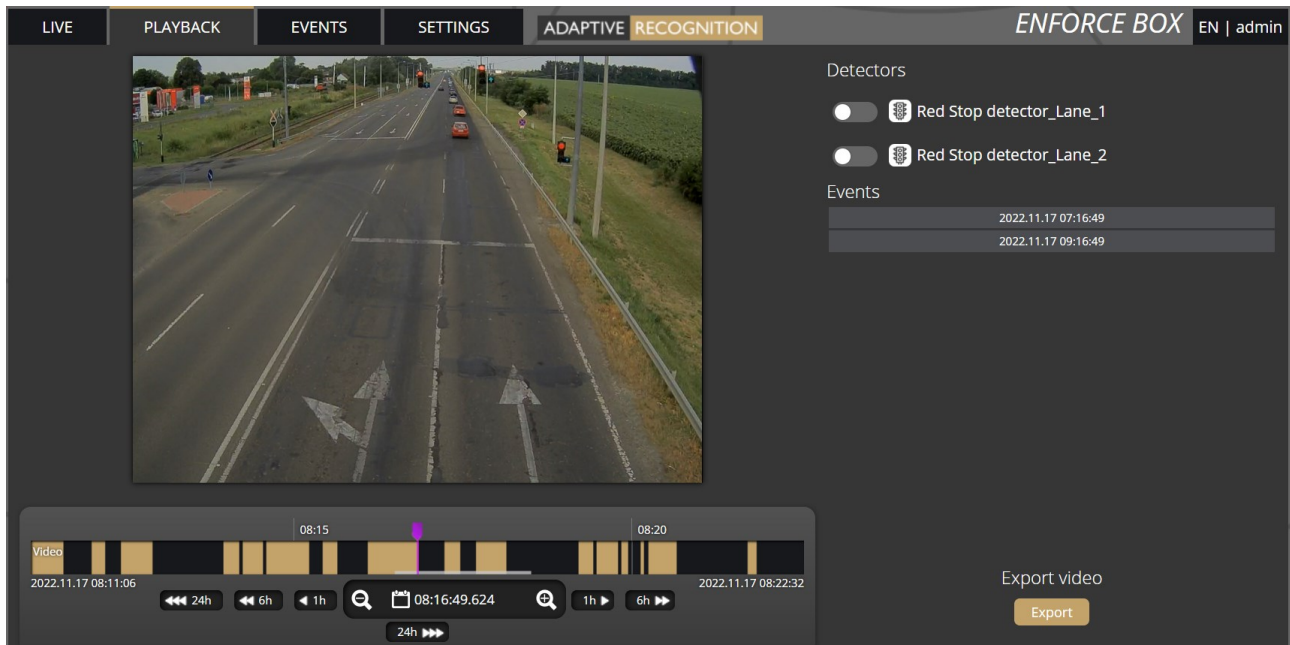
		<b>RESTORE</b> Motion detector Motion in scene returned to normal	2023.02.10 10:03:50
		<b>EVENT</b> Red Stop detector Vehicle did not stop at the red light	2023.02.10 10:03:49
		<b>EVENT</b> Red Stop detector Vehicle did not stop at the red light	2023.02.10 10:03:46
		<b>SIGNAL</b> Motion detector Motion over threshold in scene	2023.02.10 10:03:43
		<b>RESTORE</b> Motion detector Motion in scene returned to normal	2023.02.10 10:03:35
		<b>SIGNAL</b> Motion detector Motion over threshold in scene	2023.02.10 10:03:34

The list also contains the exact date and time an event was emitted. Clicking on the row of event brings up a more detailed view of that event. Clicking on the image shows the event image in full view. One more click takes you back to the event window.



## 5. PLAYBACK

You can access the **PLAYBACK** interface if the storage is turned on. By clicking on this tab, the recordings stored on the storage device will be listed. You can then navigate them by clicking on the timeline below the video feed.



### Note

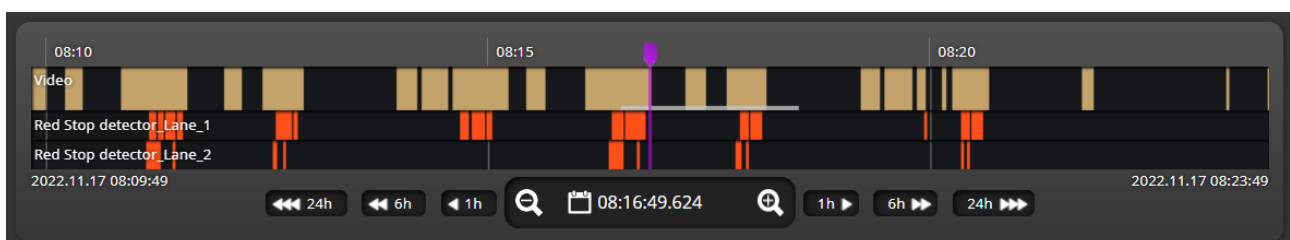
In case the storage is turned off but the storage device is available, the previously recorded elements can be viewed and played if the storage function is switched on.

### 5.1. NAVIGATE AMONG THE RECORDINGS

You can navigate among the recordings by using the timeline and calendar.

The **timeline** is the black bar under the camera image. The **gold bands** indicate those time intervals where recordings exist. Under this section, the currently selected detectors are located.

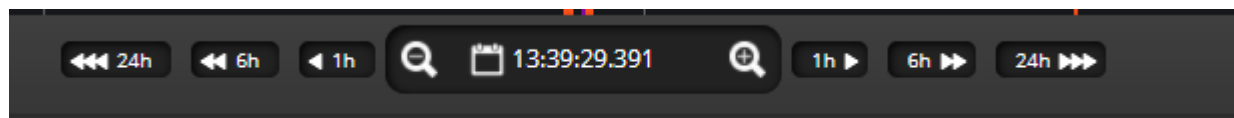
The **red markers** point where events have taken place.



Change the displayed timeline by clicking and holding the left mouse button and moving it to the left (backward in time) and/or to the right (forward in time). By clicking on the desired date, the timeline will skip to that point.

The displayed **white stripe** at the bottom of the gold timeline indicates the video parts ready to be played.

In the middle of the timeline (see image above), there is a **purple marker** that shows where you are in the playback. Under this section, you can also see the current time of the playback.



The **magnifying glasses** located under the timeline are to increase (magnifying glass with + sign) or decrease (magnifying glass with – sign) the time interval found on the timeline.

In the middle of this panel, there is a calendar with which you can seek an exact date and time to playback.

The current time of the computer can be set with the **[Now]** button. After clicking on the **[Done]** button, the playback skips to the selected date.

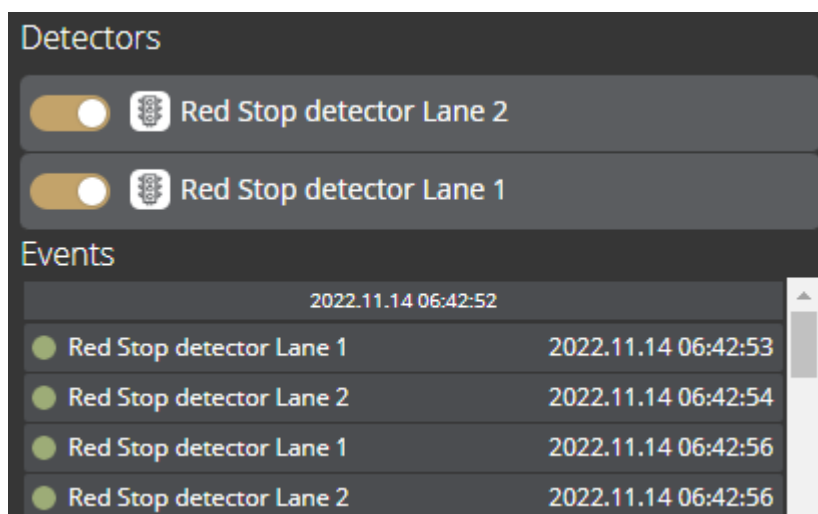
The small image that appears when the mouse cursor is positioned over the timeline shows a preview image of the video near that location.

By moving the cursor over the video, an **OSD menu** appears, the functionality of which is identical to the menu located on the live stream.

To modify the playback speed, click the **cogwheel** on the video menu and select a speed value. This is where you have the help and the image saving options.

## 5.2. FILTERING THE DETECTORS

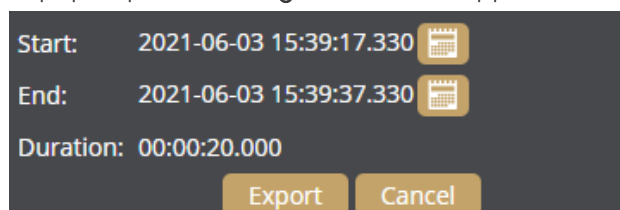
You can find a list of the configured detectors and events related to them on the right side of the **PLAYBACK** interface.



The events and timeline of each detector can be turned on/off by clicking on the appropriate detector button. Clicking on an event in the list navigates the playback to the date and time of the event. If you hover the cursor over an event located in the list, the detector related to the event is highlighted above the list. It works vice versa: by hovering the cursor over the detector, the events related to the detector will be highlighted in the list below.

## 5.3. EXPORTING THE RECORDINGS

Video clips can be saved as mp4 files and can be viewed in most modern video player applications. The **[Export]** button is located in the bottom-right corner of the **PLAYBACK** interface. By clicking on this button, a dialog box pops up, and **two gold arrows** appear on the timeline.



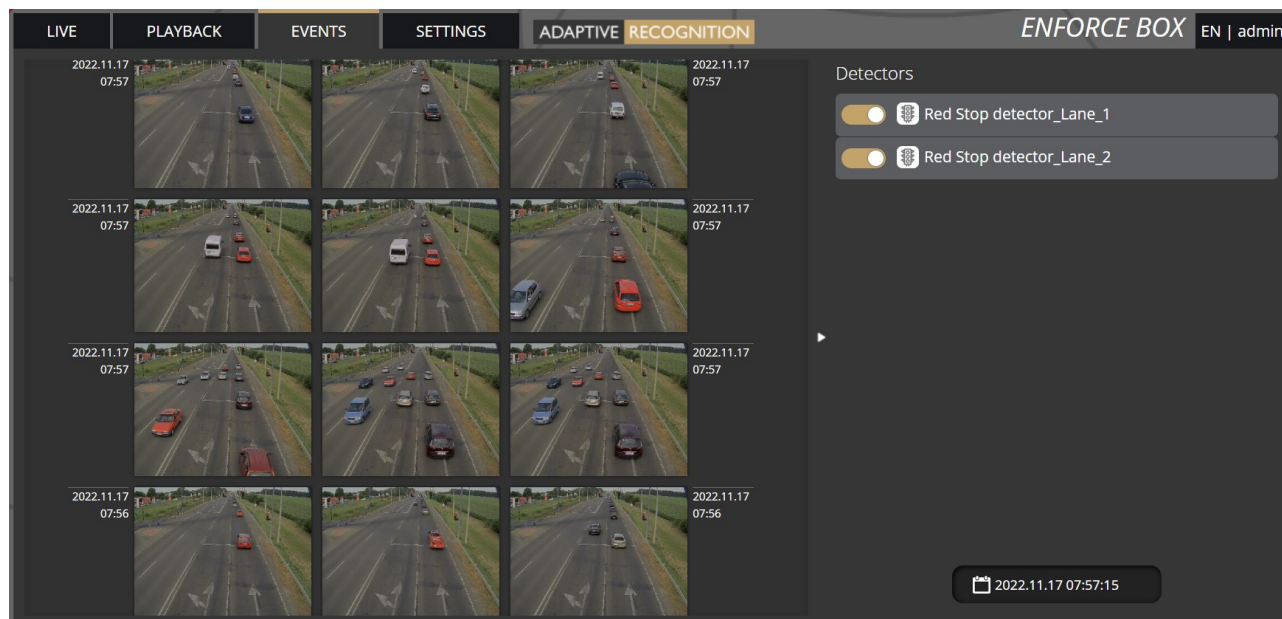
Drag the arrows with the mouse, and click the **[Calendar buttons]** next to "Start" and "End" to modify the exported time range. The duration of the video to be exported is displayed in the bottom line ("Duration").

### Note

You can adjust the exact time by clicking on the calendar icon.

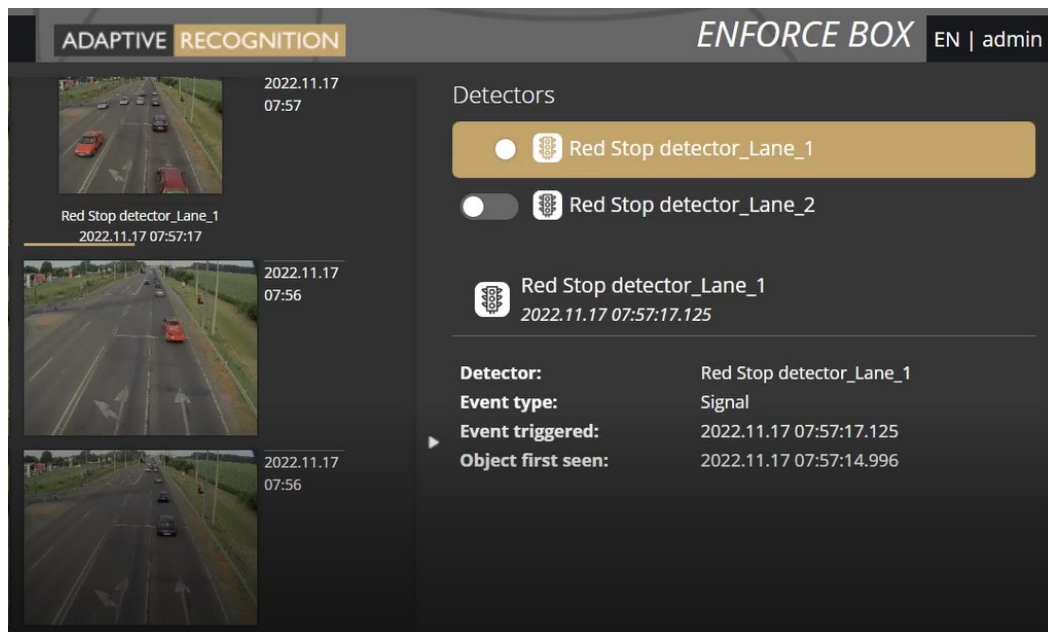
## 6. EVENTS

You can access the **EVENTS** interface provided that the storage is turned on. By clicking on this tab, all events recorded by the device will be listed.

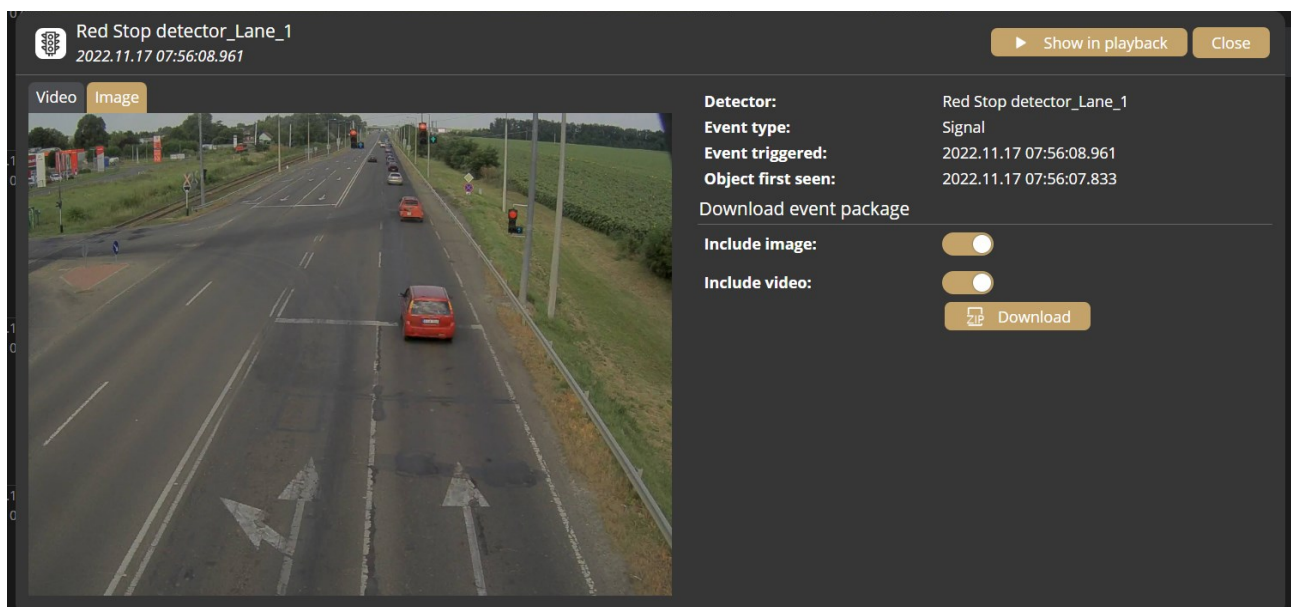


You can navigate between the recorded events in the event browser by scrolling through them with your **scroll wheel**. The events appear as small images. The latest events are at the top.

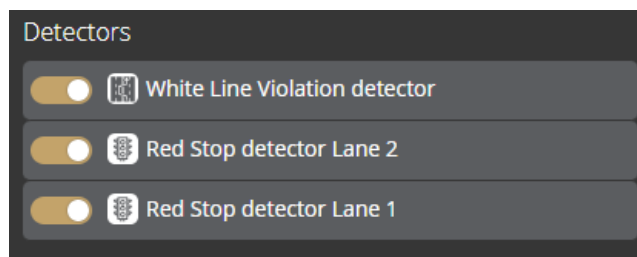
Hovering the cursor over an event, the detector related to the event is highlighted in the list on the right. Simultaneously, a video clip of the event will be loaded and played automatically.



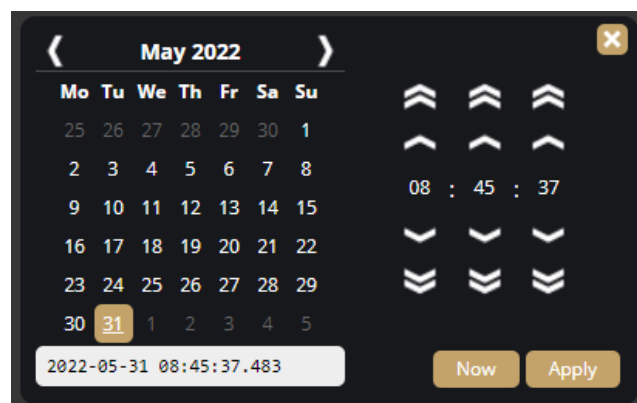
Clicking on an event brings up a detailed view of that event, including a video clip and any related image. The interface can be redirected to the **PLAYBACK** menu item by clicking on the **[Show in playback]** button. The data belonging to the event can be saved as a ZIP file by clicking on the **[Download]** button.



The configured detectors are displayed on the right. By moving the cursor over the detector, the events related to the detector will be highlighted in the event browser. By clicking on the detector, the display of its events can be turned on/off.



A calendar appears by clicking on the time located in the bottom-right corner. After setting the appropriate time and clicking on the **[Apply]** button in the calendar, the browser skips to the specified time.



## 7. SETTINGS

The SETTINGS page contains all customizable parameters of the device.

### 7.1. SYSTEM / STATUS

On this interface, you can find a summary of the important data of the device, the installed detectors, the operating time, the Traffic licenses, etc. API documentation can also be found here for integrating.

Device

Name:	Enforce Box	Type:	ENFORCE_BOX
Description:	Enforce Box	Serial:	1206260
Date & time:	2023.02.10 10:05:41	Firmware:	1.3.0.84
Storage:	Enabled	Location:	1.1, 1.1
License:	Traffic license		Uptime:
License key:	USB key - 1225906		37 minutes

For integrating this device check out the [API documentation](#)


Network


Wired connection:	10.0.7.70, 169.254.83.16 (MAC: 48b02d3e5310)	DNS:	1.1.1.1, 8.8.8.8
-------------------	---	------	------------------

Video

Video input	1920x1440 @ 3.53 Mbit/s
-------------	-------------------------

Detectors

 Red Stop detector

 Motion detector

## 7.2. SYSTEM / DEVICE

On the **Device** interface, you can do the following:

- Modify the name, description and location of the device
- Reboot the device remotely
- Perform a factory reset (after clicking on the button, the original manufacturer settings are restored except for the network settings)
- Set the date and time
- Upload firmware and license.

General

Device name:

Enforce Box

Location:

1

1

Latitude

Longitude

Device description:

Enforce Box

Save

Date & time

Device time:

2022. 11. 14. 7:53:51

Set local time

Use NTP:

NTP servers:

time1.google.com

X

pool.ntp.org

X

2.europe.pool.ntp.org

X

Add

NTP status: ?

8m

194.58.206.20

12m

216.239.35.0

9m

78.41.116.149

Save

Maintenance

Reboot:

Perform reboot

Factory reset:

Perform reset

Firmware:

Browse files...

Upload

Current firmware version: 1.2.0.208

License:

Browse files...

Upload

### Date & time settings

The device's current time is displayed at the **Device time** using your web browser's locale. The device time can be set manually by clicking on the **[Calendar icon]**. You can synchronize the device to the computer time with the **[Set local time]** button next to the calendar icon.



To automatically synchronize the time using an NTP server, turn on the **[Use NTP]** option and add an NTP server to the field of the **NTP servers**. Use at least a local NTP server if you manage more than one camera and/or use integration via API/HTTP/FTP/etc.

**! Important!**

In the case of the device being registered to the Intellio server, **do not** use NTP servers.

**NTP status** shows the current status of each configured NTP server. The color indicates the state of the server and the value is the delay until synchronization is performed again.

Color states are the following:

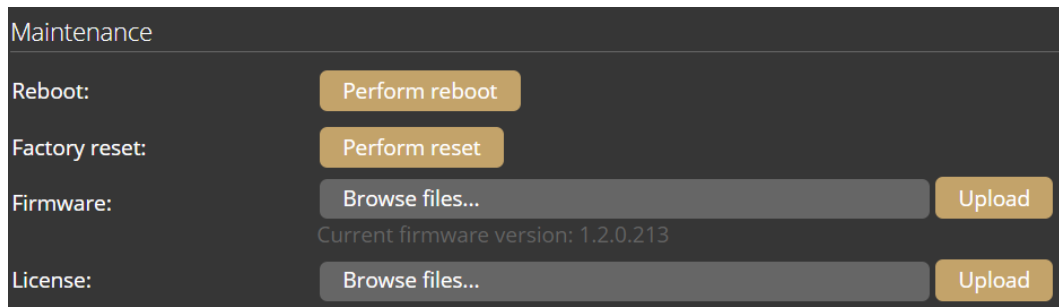
**Red:** Server is not suitable or unreachable.

**Green:** Server is working and used for synchronization.

**Gray:** Server is not used because there is a better alternative.

### Firmware, License update

Click **[Browse files...]** on the field to be modified, then select the Firmware (.ifw) or License (.ukeys) files to be uploaded. Finally, click on the corresponding upload button.



The update process can be interrupted by clicking on the **[Cancel]** button located on the panel showing the upload status.

When the upload is finished (in the case of uploading license before the update process), the device asks a security question whether you are sure about the modification. Choosing **[No]** interrupts the update process, and the device operates with the previous settings. If you opt for **[Yes]**, the update continues. Updating and rebooting the device may take a few minutes.

**! Important!**

During the update process **do not** unplug the device.

## 7.3. SYSTEM / NETWORK

The **Network** menu item hides the network settings. The IP address assigned to the device can be static or dynamic.

Default DNS: 8.8.8.8

The screenshot shows the 'Settings' tab of the Enforce Box interface. Under 'DNS servers', there are two input fields: '1.1.1.1' and '8.8.8.8', each with a delete icon (X). An 'Add' button is below them. The 'DNS search domain' has an 'Add' button. The 'Default interface' is set to 'Wired connection' with a dropdown arrow. A 'Save' button is at the bottom left of this section. Below this is the 'Interfaces' section, with 'Wired connection' selected. It shows the 'MAC address' as '48:B0:2D:3E:8F:C6'. The 'Mode' is set to 'Static address' (highlighted in orange) with a 'DHCP' button next to it. A 'Fallback to static' toggle switch is currently turned on. To the right, a table shows the 'Current address' with two rows: '192.168.6.146' with subnet '255.255.254.0' and mode 'dhcp', and '169.254.143.198' with subnet '255.255.0.0' and mode 'static'. Below the table, the 'Static address' section has input fields for 'Address' (192.168.2.171), 'Netmask' (255.255.255.0), and 'Gateway' (192.168.2.10), with a 'Save' button at the bottom.

Current address		
192.168.6.146	Subnet: 255.255.254.0	dhcp
169.254.143.198	Subnet: 255.255.0.0	static

### Fallback to static

If the device is set to DHCP, the "**Fallback to static**" option will be accessible. The device will use the configured fallback address when obtaining a new address from a DHCP server fails.

Put the Enforce Box device and your camera in the same network segment.




## Monitoring

The Monitoring tab shows statistics of active media connections (e.g., live feeds, event stream) and lists all in- and outgoing traffic by network adapter.

Settings

Monitoring

Media streams

	Client	Type	Send	Waiting	Dropped	Uptime
	192.168.9.158	RTSP	2.41 Mbit/s	0 B	0 B	7 minutes 4 seconds
	192.168.3.47	IVS	0 bit/s	0 B	0 B	3 minutes 24 seconds
	192.168.3.47	IVS	14.35 Mbit/s	0 B	0 B	3 minutes 24 seconds

Interfaces

Interface	Send	Receive
Wired connection	16.88 Mbit/s	142.32 Kbit/s

## 7.4. SYSTEM / SECURITY

In the **Users** database, you can perform the maintenance of the user data, like:

- Adding new users
- Deleting users
- Modifying the already existing user profiles

The default user name and password is "**admin**".

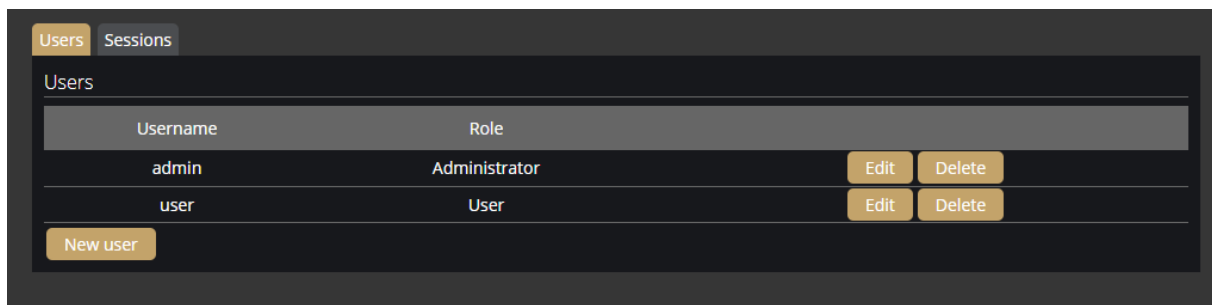
### ! Important!

To increase the security of using the device on the network, please **change the default password** of your account.

When adding a new user, you can set three levels of permissions:

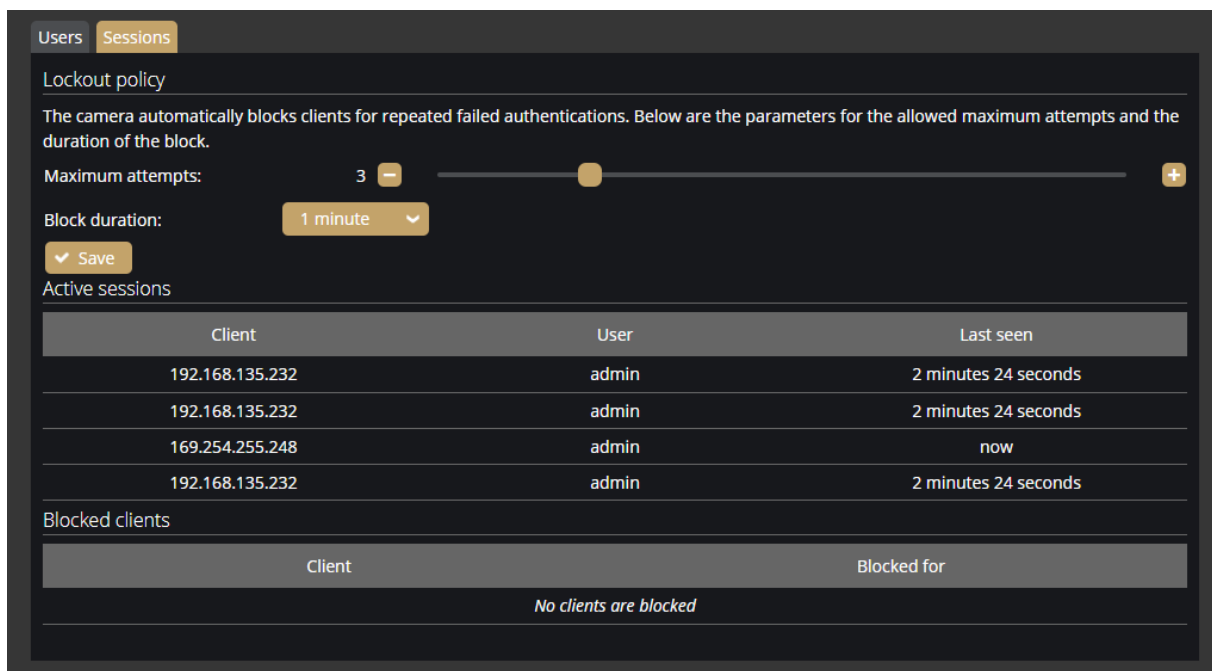
1. **Administrator**: The administrator can access and edit all parameters of the device.
2. **User**: The user can view but not edit the parameters of the device. Some pages containing sensitive information may be hidden.

3. **Operator:** The operator has the same privileges as a user.



## Sessions

At **Lockout policy**, the maximum number of failed login attempts can be adjusted. After reaching the specified number, the device blocks that session. By default, after three failed login attempts, the device blocks the IP address of the client for a minute. Note that the number of **Maximum attempts** may vary between one and ten. The duration of the block can be set between 30 seconds and seven days. The Active sessions and Blocked clients can also be seen on this tab.



## 7.5. SYSTEM / STORAGE

The settings related to the storage can be performed at **Storage**. After enabling the storage function, select a device under **Storage device** where the images, video streams and events are saved.

### Operation mode

Under **Operation mode**, the **storage trigger** can be selected. The image sequences will be saved based on this selection.

#### Important!

These settings only have an impact on the storage device. They do not affect the storage in the IVS.

The following can be selected as a **storage trigger**:

- **Event**: Only those image sequences will be stored which have taken place during the signaling of one of the selected detector(s).
- **Motion**: When the device detects motion, the storage process starts and finishes when the motion is over.
- **Event+Motion**: Storage is performed in cases of both an **Event** or **Motion**.
- **Continuous**: The storage function saves every frame regardless of event and motion.

### Recordings before and after activation (seconds)

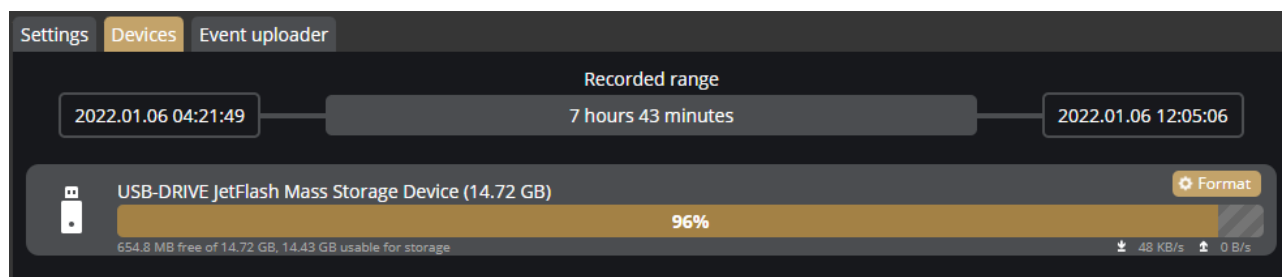
The recording time (in seconds) before and after the events can be regulated with the help of the sliders.

## Selected detectors

It may not be necessary to record at every detector signalling. Thus, the user can select which detector signal(s) should trigger the recording.

## Devices

Under the **Devices** tab, information about the data of the storage device, the length of the recordings, the available storage, and the writing speed can be found.



## Formatting the storage device

With the **[Format]** button, you can format the storage unit immediately. After clicking on the **[Format]** button, a window pops up. Click on the **[Yes]** button to start the operation. The capacity bar indicates the remaining time of the formatting process.

### Important!

The formatting deletes every data from the storage device.

## Setting the GDS upload

Enable the upload to the GDS (Globessey Data Server) at **Upload mode**, then enter the required data to set the GDS server. A storage device is required for GDS upload to work. The uploader sends data only from previously stored content.

The following fields should be defined:

- **Server:** Address (IPv4) of the GDS server
- **Port:** Access port of the GDS server
- **Path:** Access within the server
- **Table name:** The name of the GDS table where the upload will be done
- **Username:** Username required for the identification
- **Auto obtain user:** The username can be queried automatically. The device queries the user token, which will be the user. However, it has to be authorized manually from the GDS site by a second party.
- **Reset Uploader:** Resets the uploader progress to the current date. Event that are older and not yet uploaded will be ignored.

In the **Uploader status** section, you can view the status and the data of the uploader.

Settings Devices **Event uploader**

Upload mode: **GDS**

GDS settings

Server: 192.168.6.80 Port: 8888 Path: gate Username: primula

Table name: multi\_event Auto obtain user: ☒

Reset Uploader: **Reset uploader**

☒ Save

Uploader status

Target: gds://primula@192.168.6.80:8888/gate#multi\_event

Position: 2022.10.28 08:49:42

Status: Success

0%

## Setting the HTTP(S) upload

Enable the upload to the HTTP/HTTPS POST at **Upload mode**, then enter the full URL of the web service to set the HTTP event receiver. A storage device is required for HTTP(S) upload to work. The uploader sends data only from previously stored content.

In the **Uploader status** section, you can view the status and the data of the uploader.

The screenshot shows a web interface for configuring the uploader. The top section, titled "HTTP/HTTPS POST settings", contains several controls: a "Server:" field with the value "http:192.168.2.111:8083/ar\_http\_upload.php", three toggle switches for "Upload images:", "Upload cropped images:", and "Upload videos:" (all are turned on), a "Reset Uploader:" button labeled "Reset uploader", and a "Save" button. To the right of the "Upload images:" toggle is a "Media content sent with:" section with two buttons: "Name" (selected) and "Name and filename". The bottom section, titled "Uploader status", displays the following information: "Target: HttpPost: http:192.168.2.111:8083/ar\_http\_upload.php", "Position: n/a", and "Status: Success". At the very bottom, a dark bar with diagonal stripes contains the text "Nothing to upload".

You can set which data should be uploaded in addition to the event data:

- Event image
- A cropped image of the license plate
- Video of the event

**Reset Uploader.** Resets the uploader progress to the current date. Event that are older and not yet uploaded will be ignored.

In the **Uploader status** section, you can view the status and the data of the uploader.



## Setting the FTP upload

Enable the upload to the FTP at **Upload mode**, then enter the required data to set the FTP upload. A storage device is required for FTP upload to work. The uploader sends data only from previously stored content.

The following fields should be defined:

- **Protocol:** the services that are supported by the uploader (FTP(ES), FTPS, SFTP) can be selected
- **Server:** IP address (IPv4) or hostname of the FTP server
- **Port:** the service's port where it listens to requests
- **Username/password:** Username and password required for the identification
- **Reset Uploader:** Resets the uploader progress to the current date. Event that are older and not yet uploaded will be ignored.

With the **[Start test]** button you can test the connection between the camera and the FTP server.

In the **Uploader status** section, you can view the status and the data of the uploader.

The screenshot shows a web interface for configuring FTP settings. The 'FTP settings' section includes fields for Protocol (ftp(es)), Server (192.168.2.111), Port (21), Path (events/%MAC/%DATE/), Username (intellio), and Password (masked). There is a toggle for 'Upload videos' and buttons for 'Start test', 'Reset uploader', and 'Save'. The 'Uploader status' section shows the Target (ftp://192.168.2.111:21/events/3df204/2022-10-28/), Position (n/a), and Status (Success). A banner at the bottom indicates 'Nothing to upload'.

FTP settings	
Protocol: ?	ftp(es)://
Server: ?	192.168.2.111
Port: ?	21
Path: ?	events/%MAC/%DATE/
Username:	intellio
Password:	.....
Upload videos: ?	<input type="checkbox"/>
Test settings	Start test
Reset Uploader: ?	Reset uploader
	Save

Uploader status	
Target:	ftp://192.168.2.111:21/events/3df204/2022-10-28/
Position:	n/a
Status:	Success

Nothing to upload

## 7.6. I/O

In the I/O menu you can modify the input and output settings of the added ONVIF devices, the trigger configuration and you can monitor the state changes of the input/output ports at IO log section.

Inputs		Outputs	
Port		Idle state	Active state
ONVIF_DigitalInput_DS-2CD2683G2-IZS_AlarmIn_1		High	Low
ONVIF_DigitalInput_Einar-5T_IN_0		High	Low

IO log		
08:05:12	ONVIF_DS-2CD2683G2-IZS_AlarmOut_0	Port has been deactivated
08:05:17	ONVIF_DigitalInput_DS-2CD2683G2-IZS_AlarmIn_1	Port has been deactivated
14:09:36	ONVIF_Einar-5T_OUT_0	Port has been deactivated
14:09:36	ONVIF_DigitalInput_Einar-5T_IN_0	Port has been activated

On the input side, you can change the Auto-restore timeout value. If the Input stays in 'Active' state more than the given timeout in milliseconds then it will be restored to 'Deactive' state.

**Edit port ONVIF\_DigitalInput\_DS-2CD2683G2-IZS\_AlarmIn\_1**

Port: ONVIF\_DigitalInput\_DS-2CD2683G2-IZS\_AlarmIn\_1

Auto-restore timeout (ms): ?  − +

✓ Save Cancel

You can also modify the Output ports at the Outputs tab.

Inputs		Outputs			
Port		Idle state	Active state	Mode	Active
ONVIF_Einar_tesztzoba_OUT_0		Open	Closed	Impulse: 500 ms	No

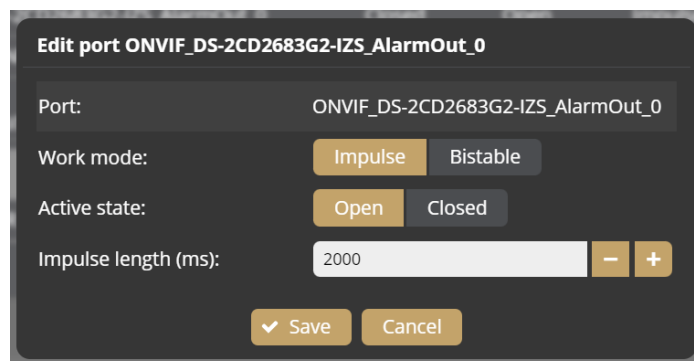
Trigger configuration		
Detector		Ports
White Line Violation detector		none
Red Stop detector Lane 2		none
Red Stop detector Lane 1		none

✓ Save

The following parameters can be adjusted after clicking on the Edit button:

- **Work mode:** Impulse or Bistable
- **Active state:** The active state of the port. If it is "Open", the port is open when an event occurs. If it is "Closed", the port closes when an event occurs.

- **Impulse length (ms):** In the case of activating the output port, the length of the active state can be adjusted.



**Edit port ONVIF\_DS-2CD2683G2-IZS\_AlarmOut\_0**

Port: ONVIF\_DS-2CD2683G2-IZS\_AlarmOut\_0

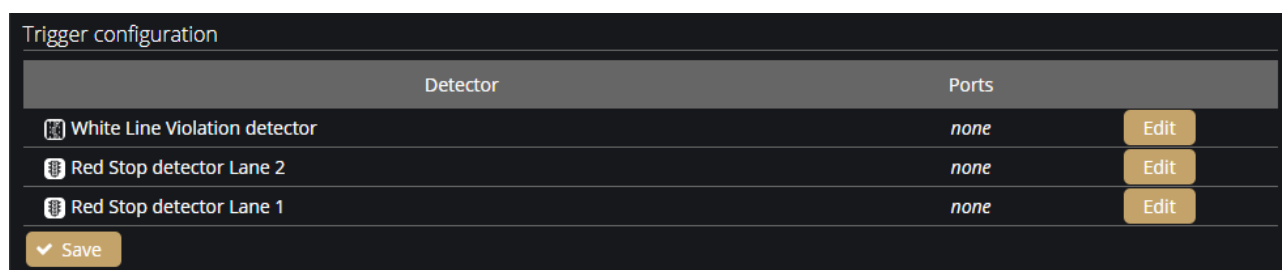
Work mode: **Impulse** Bistable

Active state: **Open** Closed

Impulse length (ms): 2000 - +

✓ Save Cancel

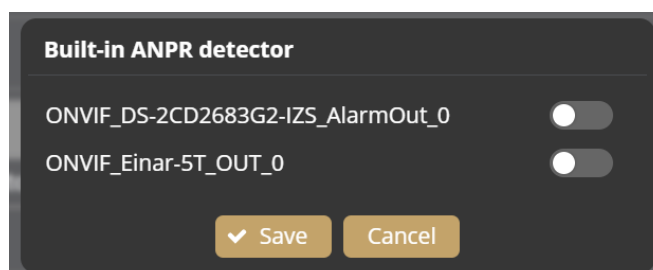
The detectors configured in the Analytics/Detectors menu can be associated to the Ports of the added ONVIF device in the Trigger configuration menu. (You can add an ONVIF device in the **External** menu.) For some cameras, the IO port must also be enabled separately on the camera.



Detector	Ports	
White Line Violation detector	none	Edit
Red Stop detector Lane 2	none	Edit
Red Stop detector Lane 1	none	Edit

✓ Save

Select which ONVIF device the Detector should be associated with.



**Built-in ANPR detector**

ONVIF\_DS-2CD2683G2-IZS\_AlarmOut\_0 ☐

ONVIF\_Einar-5T\_OUT\_0 ☐

✓ Save Cancel

## 7.7. SYSTEM / SERVICE

### Webserver

- **Service port / Secure service port:** The service ports of the Webserver can be specified by filling in the field.

### RTSP

- **Service port:** The service port of the RTSP can be specified by filling in the field.
- **Authentication required:** By selecting **Enabled**, authentication is required when connecting to the RTSP stream.

### UPnP

- **Allow discovery:** Enable or disable the device discovery provided by the UPnP protocol.

### IVS

- **Service port:** The service port of the IVS can be specified by filling in the field.

The screenshot displays a configuration window with a dark background and light text. It is organized into sections for different services. The 'Webserver' section has two input fields: 'Service port' with the value '80' and 'Secure service port' with the value '443'. The 'RTSP' section has a 'Service port' field with '554' and an 'Authentication required' section with two buttons, 'Disabled' and 'Enabled', where 'Enabled' is selected. The 'UPnP' section has an 'Allow discovery' section with 'Disabled' and 'Enabled' buttons, where 'Enabled' is selected. The 'IVS?' section has a 'Service port' field with '53539'. At the bottom left, there is a 'Save' button with a checkmark icon.

Section	Field	Value
Webserver	Service port	80
	Secure service port	443
RTSP	Service port	554
	Authentication required	Enabled
UPnP	Allow discovery	Enabled
IVS?	Service port	53539

## 7.8. SYSTEM / NOTIFICATIONS

In the **Messages** tab of this configuration interface, you can find system messages of the device.

Messages		Email	
#	Date	Type	Description
1	2022.05.06 00:55:46	Storage	USB-DRIVE JetFlash Mass Storage Device plugged in
0	2022.05.06 00:55:36	System	System started at 2022.05.06 00:55:36
Refresh			
* Only the last 1000 entry is shown!			

In the **Email** tab, you can specify the email settings for sending messages. The following parameters can be adjusted after clicking on the **[Enabled]** button:

- **Delay between messages:** After sending an email, the device will wait at least the selected duration before it can send another email.
- **Exclude:** Notification types selected here are excluded from the email messages.
- **SMTP settings:** enter the required data to set the access of the SMTP service.
- **E-mail settings:** set the display name and the email address that the device uses when sending email messages. The "Send to" field is used to set the recipients

Messages
Email

E-mail notifications:
Disabled
Enabled

Delay between messages: ?
1 minute

Exclude: ?
Storage
NTP
Security
System
License

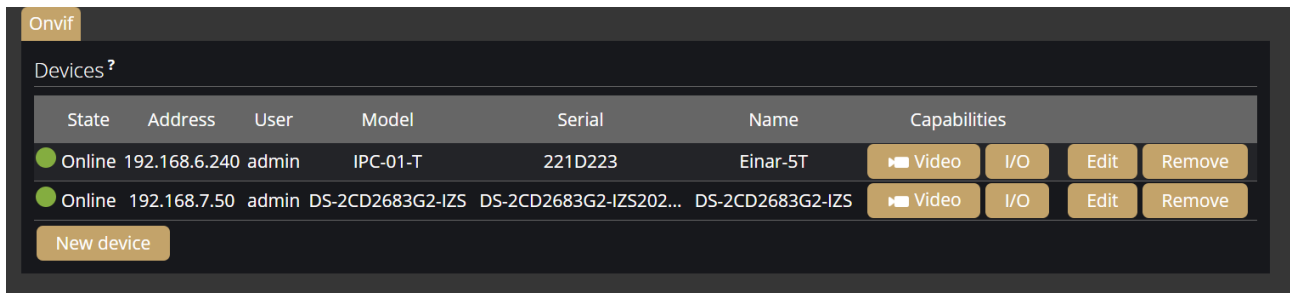
SMTP settings
Host:
smtp.gmail.com
Port:
465
Encryption:
SSL/TLS
Username:
te@gmail.com
Password:

E-mail settings
Sender name: ?
Tester
(e.g.: CAM-Floor3)
Sender address: ?
te@gmail.com
Send to:
lo@gmail.com

\* One email address per line
Save
Test settings ?
Test settings

## 7.9. EXTERNAL

You can manage the associated Onvif devices in the External menu. You can add a new device, edit the data of existing devices and delete a device.



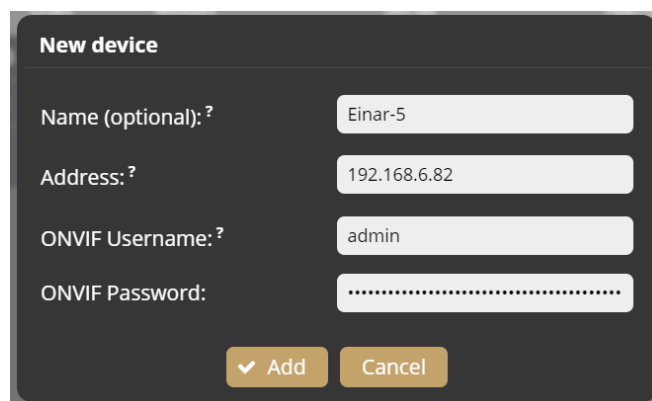
State	Address	User	Model	Serial	Name	Capabilities
Online	192.168.6.240	admin	IPC-01-T	221D223	Einar-5T	Video  I/O  Edit  Remove
Online	192.168.7.50	admin	DS-2CD2683G2-IZS	DS-2CD2683G2-IZS202...	DS-2CD2683G2-IZS	Video  I/O  Edit  Remove

New device

For the added ONVIF device the Video button will only appear if the device does support a video channel. The IO button only appears if the added ONVIF device supports IO. Clicking on the Video or IO button will switch to that menu item.

The following should be set when adding a new device:

- **Name: (optional):** The name of the device can be entered.
- **Address:** IP address where the device is accessible.
- **ONVIF Username:** The device's ONVIF username.
- **ONVIF Password:** The device's ONVIF password.



**New device**

Name (optional):

Address:

ONVIF Username:

ONVIF Password:

Add Cancel

### Note

For many cameras, the ONVIF Username and Password do not match the username and password used in the browser. ONVIF may also need to be enabled on the camera.

## 7.10. MEDIA / VIDEO

When clicking on the **Video** menu item, the video stream of the connected camera can be specified by filling in the field. Above these, the live stream of the connected camera remains visible.

### Video input

The following parameters can be set:

- **RTSP source:** Select the streams of the added ONVIF devices from the list or select Manual option. If select the Manual option, enter the video stream url of the connected camera. Enforce Box can receive H.264 stream only. Put the Enforce Box device and your camera in the same network segment.
- **RTP over RTSP:** Determines the channel which is used to send video. Enable this option to use the reliable TCP connection.
- **User authentication:** If Enforce Box needs authentication to receive video stream from the connected camera, select „Enabled” and fill in the „User” and „Password” fields.

Some examples of RTSP source field values in case of different manufacturers:

#### AR Vidar, MicroCam:

rtsp://Camera\_IP/stream/h264

#### Intellio Visus:

rtsp://Camera\_IP:554/primary/h264

rtsp://Camera\_IP:554/secondary/h264

#### Intellio Initio:

rtsp://Camera\_IP:554

**AXIS:**

rtsp://Camera\_IP/axis-media/media.amp

rtsp://Camera\_IP:554/axis-media/media.amp?videocodec=h264&camera=1&fps=15&resolution=1920x1080

rtsp://Camera\_IP:554/onvif-media/media.amp?profile=profile\_1\_h264&sessiontimeout=60&stream-type=unicast

rtsp://Camera\_IP:554/onvif-media/media.amp?profile=profile2&sessiontimeout=60&stream-type=unicast

(tested with AXIS P1447-LE, Firmware version: 9.10.1)

**Bosch:**

rtsp://Camera\_IP:554/rtsp\_tunnel?p=0&h26x=4

rtsp://Camera\_IP:554/rtsp\_tunnel?p=1&inst=2&h26x=4

(tested with NBE-5503-AL, Firmware version: 6.60.0065)

**Dahua:**

Main stream:

rtsp://Camera\_IP:554/cam/realmonitor?channel=1&subtype=0&unicast=true

rtsp://Camera\_IP:554/live

Sub streams:

rtsp://Camera\_IP:554/cam/realmonitor?channel=1&subtype=1&unicast=true (if Sub Stream 1 is enabled in the camera)

rtsp://Camera\_IP:554/cam/realmonitor?channel=1&subtype=2&unicast=true (if Sub Stream 2 is enabled in the camera)

(tested with IPC-HDBW4431E-ASE, FW: 2.460.0000.14.R, Build Date: 2017-07-20)

**Hanwha:**

rtsp://Camera\_IP/profile2/media.smp

(tested with PNO-A6081R, FW: 2.11.02\_20210630\_R206)

**Hikvision:**

Main stream:

rtsp://Camera\_IP:554/Streaming/Channels/101

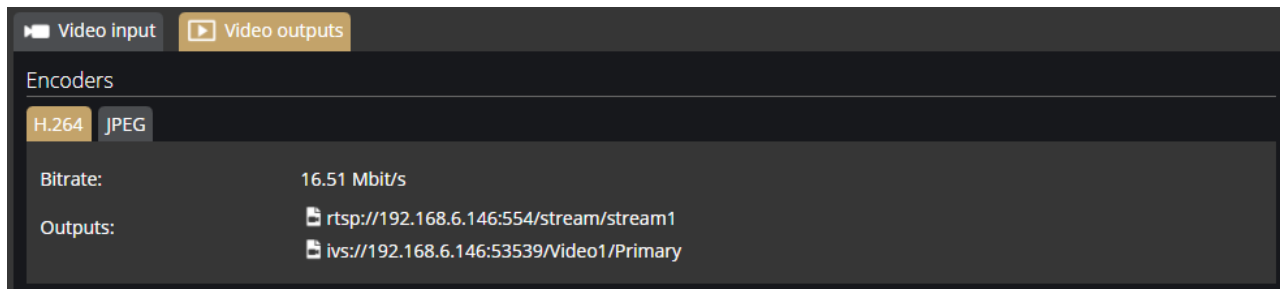
Sub stream:

rtsp://Camera\_IP:554/Streaming/Channels/102

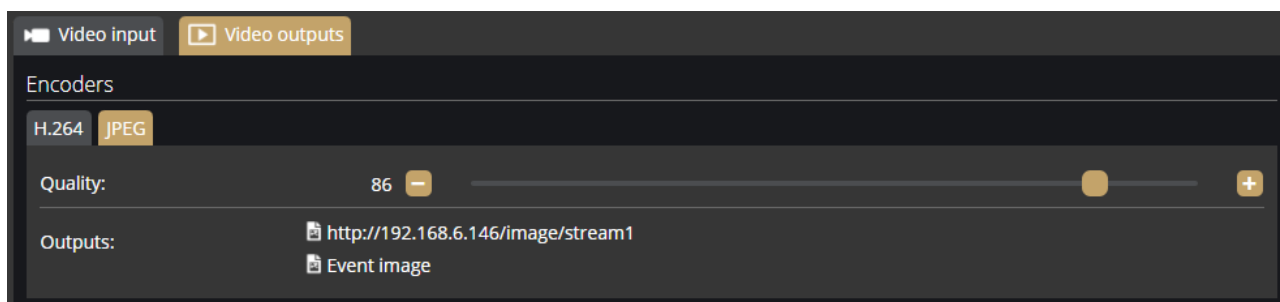


## Video outputs

Enforce Box can transfer the incoming video stream with a slight delay. These stream urls can be seen here.



JPEG still images can be accessed with the following url, and event image quality can be set here.



Videostream / image URL links can be found on this interface, such as:




H.264 stream: `rtsp:// ENFORCE _BOX_IP:554/stream/stream1`

JPEG image: `http:// ENFORCE _BOX_IP/image/stream1`

## 7.11. ANALYTICS / SETTINGS

### Status

The registered detectors' name, type, ID and status are displayed on the page alongside the list of detectors supported by the device and their current/total quantity.

Settings <span>Status</span>			
Detectors			
Detector	Type	ID	State
 White Line Violation detector	White line violation detector	{68A8613A-61FF-4FD2-B39D-50467714AED4}	✓
 Red Stop detector Lane 2	Red stop detector	{ED966779-2D02-48AC-4562-598625815E48}	✓
 Red Stop detector Lane 1	Red stop detector	{316C8AFB-6B80-4872-2C97-DB3849520211}	✓
Supported detectors			
Type	Currently active	Maximum supported	
Emergency lane detector	0	16	
Forbidden zone detector	0	16	
IO detector	0	16	
Lane detector	0	16	
Motion detector	0	16	
Red stop detector	2	16	

## 7.12. ANALYTICS / DETECTORS

You can add, modify or delete the device's detectors in this window.

### 7.12.1. Motion engine and general use of masks

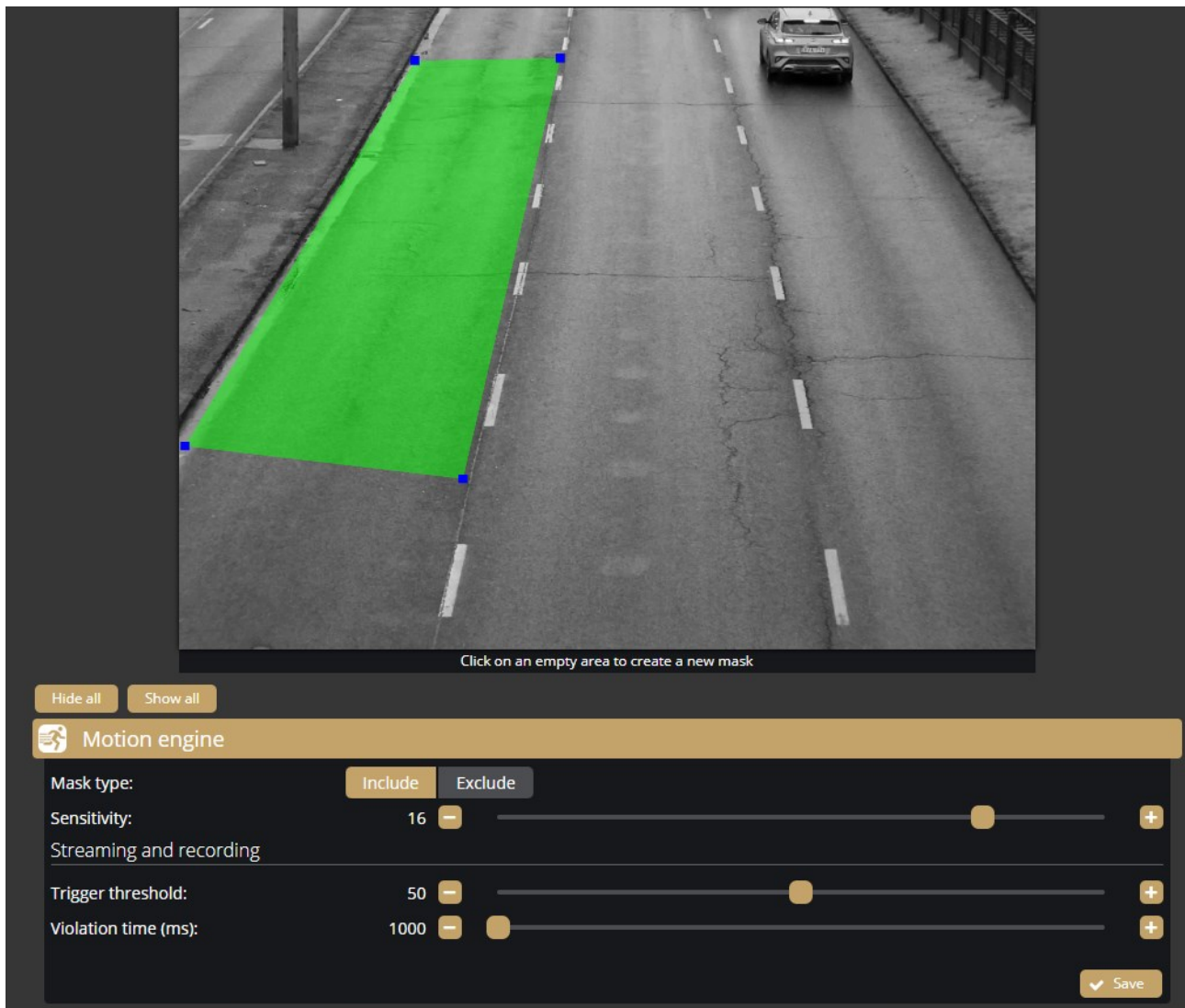
The Motion engine is a fundamental engine that regulates motion-based storage. It cannot be deleted.



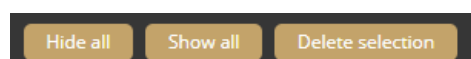
The Motion engine does not induce events; it is responsible for the setting of the motion-based recordings.

If you click on the engine, a mask can be applied to the live stream. This can be set to exclusive or inclusive with the "**Masks Type**" option. If the mask is set to "Include", the engine will only trigger when motion happens inside the selected area. When it is set to "Exclude", it will not trigger inside the area.

The mask can be modified by clicking on the green area.

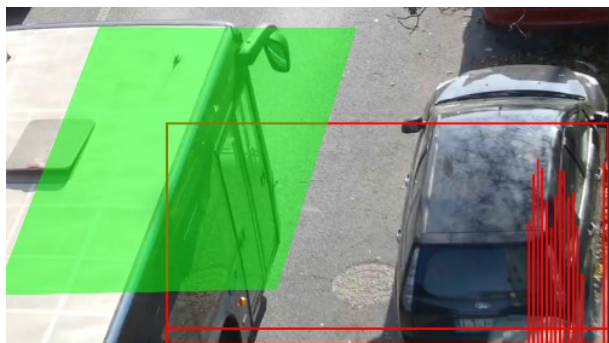


The mask can be deleted by clicking on the **[Delete selection]** button located under the live stream:



**Sensitivity:** It sets the sensitivity of the motion engine.

**Trigger threshold:** You can use it to define the sufficient level of motion in the image to trigger the motion engine. Further filtering can be done with the previously set sensitivity conditions to determine the degree of action intensity triggering recording. The "motion graph" is the OSD belonging to the setting, which can provide visual assistance. See also **{Overlay}**.



### 7.12.2. Motion detector

The Motion detector can be used to create events based on Motion engine. The following can be adjusted on the Motion detector interface:

- **Name:** The name of the detector can be entered.
- **Description:** To add a brief description to the detector.
- **Trigger threshold:** You can use it to define the sufficient level of motion in the image to trigger the motion detector.
- **Violation time (ms):** The time between sensing movement and the alarm event. If the movement stops during the masked area during this time, the alarm will not sound.

**Motion detector** Motion

Name: Motion detector

Description: Signals on any movement in the selected area

Trigger threshold: 50 — — — — — +

Violation time (ms): 1000 — — — — — +

Save Disable Delete

## iTracking Engine and iTracking Detectors

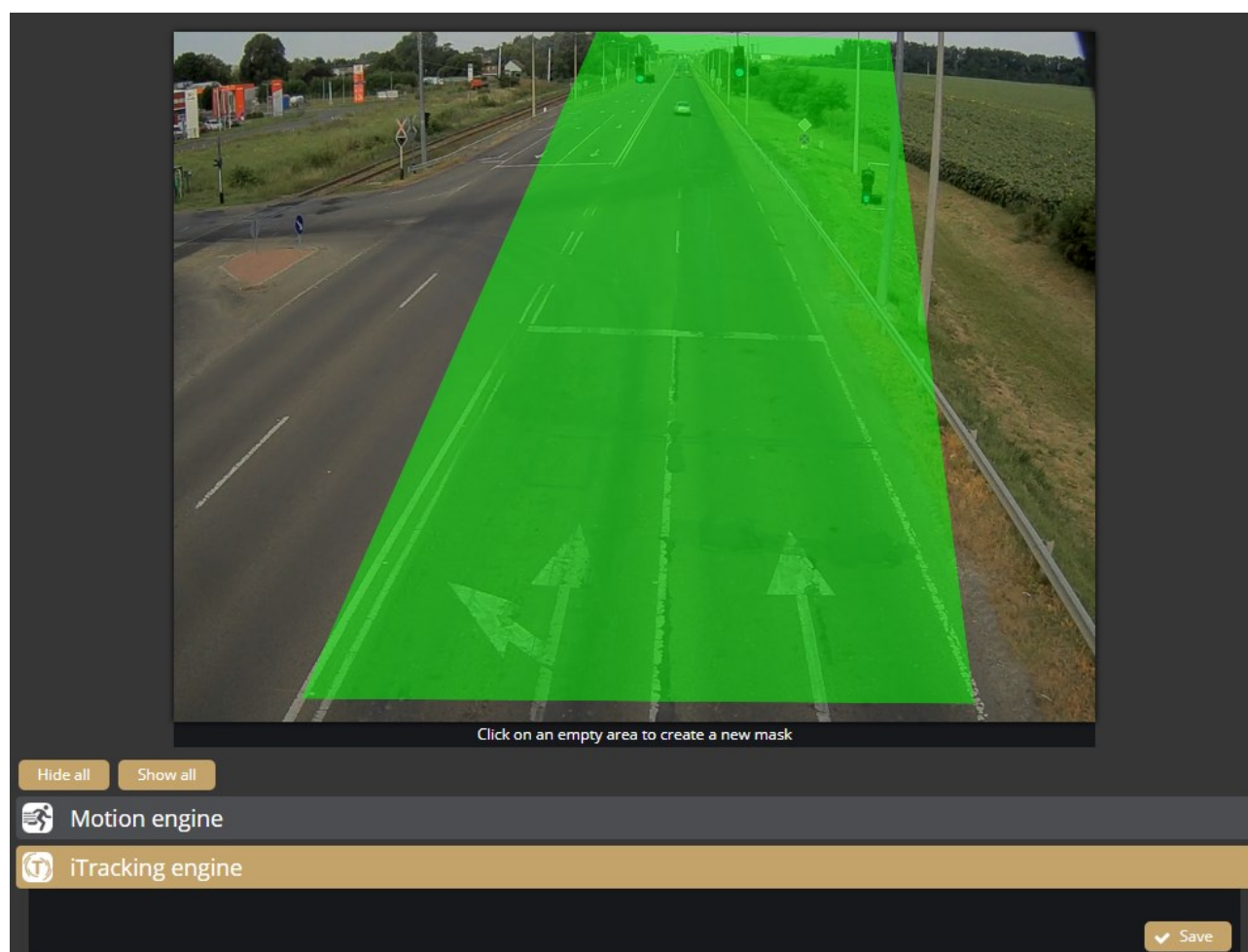
The iTracking Engine and the iTracking Detectors identify the vehicle that has committed the set of fence.

### ! Important!

Both the iTracking Engine and an iTracking Detector must be present and enabled on the device to operate the system. The mask of the iTracking Engine and the mask(s) of the iTracking Detector(s) must have a common area where the detected license plate number will trigger an event.

### 7.12.3. iTracking engine

You can create a mask on the image, after which offending vehicles will be detected only in the selected area.

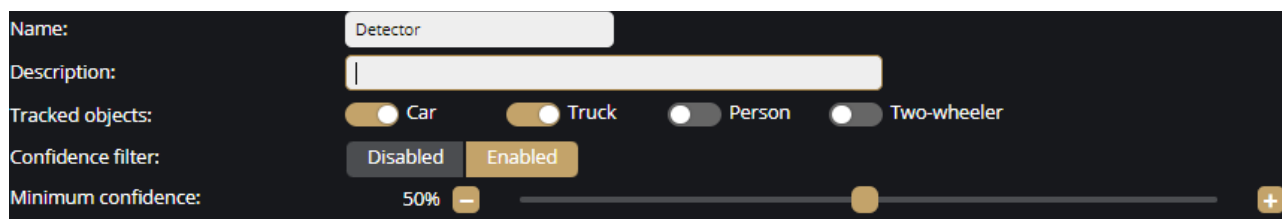


#### 7.12.4. iTracking Detectors

iTracking detector(s) is responsible for creating events from the results of the iTracking Engine.

The following iTacking detectors can be used:

	Emergency Lane detector
	Forbidden Zone detector
	Lane detector
	Network Anpr detector
	Red Stop detector
	Stop Violation detector
	Stopped Object detector
	Traffic Line detector
	U Turn detector
	White Line Violation detector
	Wrong Turn detector
	Wrong Way detector



The screenshot shows a configuration form for an iTracking detector. It includes fields for 'Name' (containing 'Detector'), 'Description' (empty), and 'Tracked objects' (with toggle switches for Car, Truck, Person, and Two-wheeler). There is a 'Confidence filter' section with 'Disabled' and 'Enabled' buttons, and a 'Minimum confidence' slider set to 50%.

The following settings are available for all iTracking detectors:

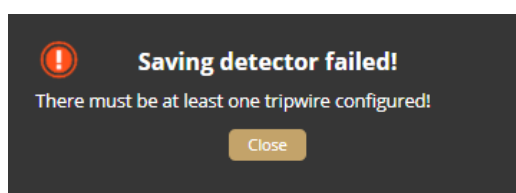
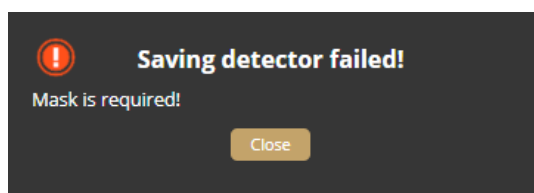
- **Name:** the name of the detector can be entered
- **Description:** To add a brief description to the detector
- **Tracked objects:** You can select which object types are detected by the iTracking detector
- **Confidence filter:** Enable to specify a minimum confidence
- **Minimum confidence:** Adjustable between 1% and 100%

Objects with a confidence below the set value will be ignored during detection. If the confidence filter is disabled, an event can be generated for all objects with a confidence value.

### ! Important!

**For each iTracking Detector a wire or mask should create in the image.**

If the wire or mask does not draw, one of the following error messages will be displayed after saving:





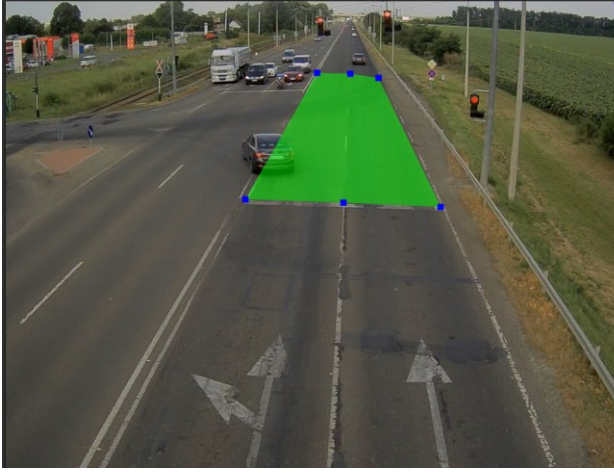
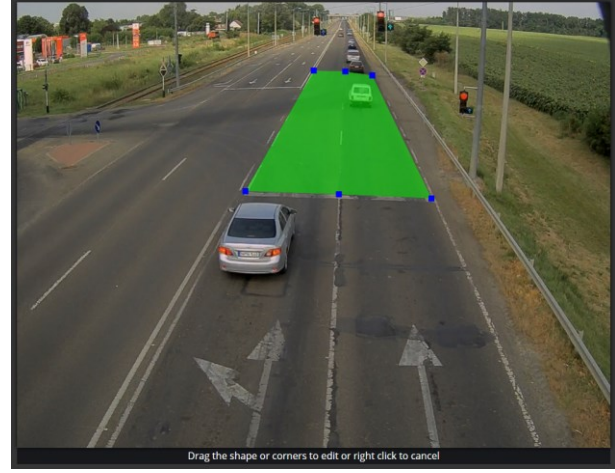


## How to create a wire.

<p>1. Step</p> 	<p>2. Step</p> 
<p>In the Settings/Detectors section, click on New Detector button and select the detector you need. After selecting the detector, the detector settings options are displayed and a new wire can be create on the image.</p>	<p>Left-click on the image where the wire will start-point. A blue point will then appear on the image.</p>
<p>3. Step</p> 	<p>4. Step</p> 
<p>Left-click on the image where the wire will end-point. A blue dot will then appear in the picture and the green wire will appear. Right-click on the image to finish.</p>	<p>To make further edits, left-click on the wire. Drag the shape or corners to edit or right click to cancel. When the wire is final, click Save button in the Detector section.</p>



## How to create a mask.

1. Step	2. Step
 <p>Click on an empty area to create a new mask</p>	 <p>Click on an empty area to create a new mask</p>
<p>In the Settings/Detectors section, click on New Detector button and select the detector you need. After selecting the detector, the detector settings options are displayed and a new wire can be create on the image.</p>	<p>Left-click on the image to create the first point of the mask. A blue point will then appear on the image.</p>
3. Step	4. Step
	 <p>Drag the shape or corners to edit or right click to cancel</p>
<p>Left-click on the image to add the other points of the mask. After the third point is added, a green mask appears, which is always modified when a new point is added. When you have added all the points, right-click on the image.</p>	<p>To make further edits, left-click on the mask. Drag the shape or corners to edit or right click to cancel. When the wire is final, click Save button in the Detector section.</p>

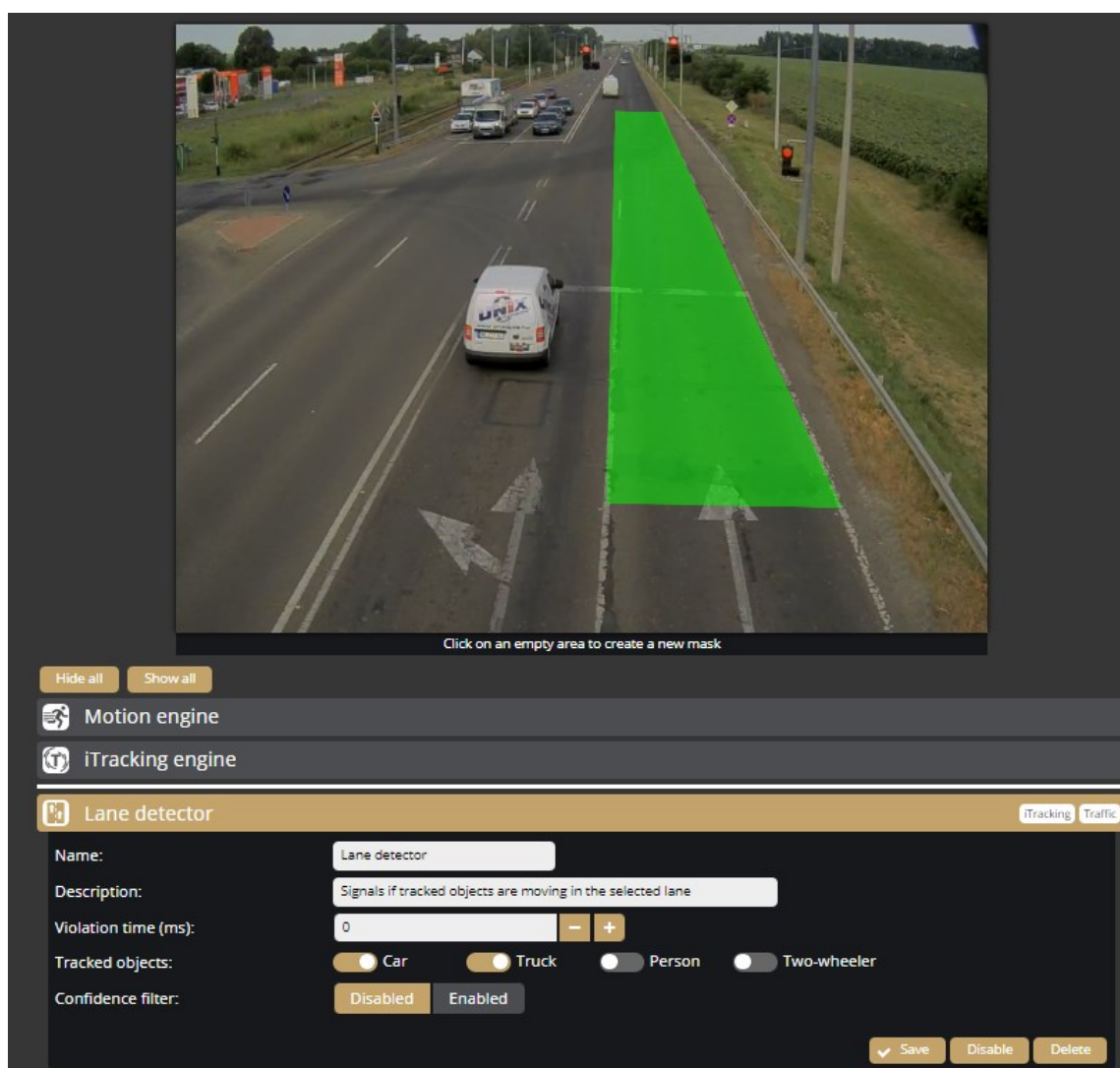
#### 7.12.4.1 Emergency Lane Detector

The Emergency Lane Detector detects vehicles those pass in the emergency lane. You must draw a mask that covers the emergency lane in the image.

#### 7.12.4.2 Forbidden Zone detector

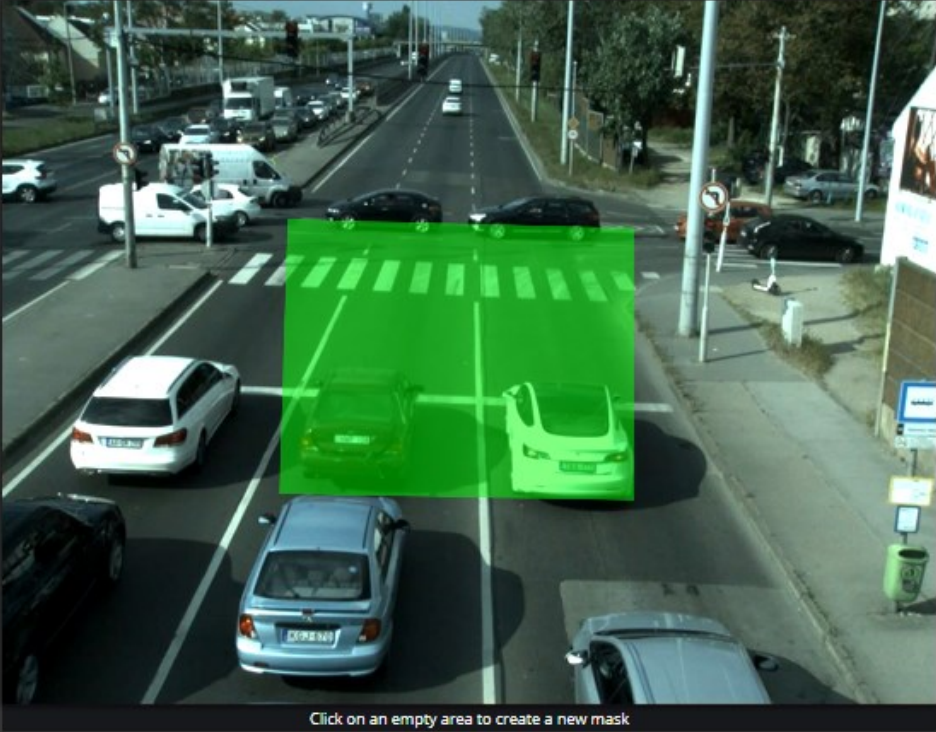
The Forbidden Zone Detector detects vehicles those pass in the emergency lane forbidden zone. You must draw a mask that covers the area in the image this will be the forbidden zone.

#### 7.12.4.3 Lane detector



The Lane Detector detects vehicles that pass in the lane. You must draw a mask that covers the lane to be scanned in the image.

## 7.12.4.4 Network ANPR detector



Click on an empty area to create a new mask

Hide all Show all

Motion engine

iTracking engine

**Network Anpr detector** ANPR Async Traffic

Name: Network Anpr detector

Description: Signals when the configured ANPR camera detects a license plate

AR device family: Vidar

AR device URL: 192.168.1.201

Minimum confidence: 75% — +

Authentication: Disabled Enabled

Save Disable Delete

With the Network ANPR detector you can associate Vidar camera event data (ANPR, MMR, Image) with the Enforce detector(s) used.

Steps to setup:

1. Enter the IP address of the Vidar camera
2. On the image, draw the area that the Vidar camera's ANPR sensor sees. This will be the area covered by the green mask.
3. Save the detector

You can use 4 Network ANPR detectors at the same time, i.e. you can retrieve ANPR data packets from 4 Vidar cameras and associate them to the event created by EnforceBox.



## 7.12.4.5 Red Stop detector

The screenshot shows the Enforce Box software interface. On the left is a sidebar with a menu containing: SYSTEM (Status, Device, Network, Security, Storage, I/O, Service, Notifications, External), MEDIA (Video), ANALYTICS (Settings, Detectors). The 'Detectors' option is highlighted. The main area shows a live video feed of a road. A white car is in the center. A green line is drawn across the road, and a red line is drawn along the side of the road. Red arrows point to the green line and the red line. Below the video feed is a configuration panel for the 'Red Stop detector'. The panel includes fields for Name, Description, Light type, Tolerance time (ms), Tracked objects, and Confidence filter. The 'Light type' is set to 'Traffic'. The 'Tracked objects' are set to 'Car', 'Truck', 'Person', and 'Two-wheeler'. The 'Confidence filter' is set to 'Enabled'. There are buttons for 'Save', 'Disable', 'Delete', and 'New detector'.

The Red Stop detector detects vehicles those drive through the red light. Red Stop detector will generate an event if the detected object crosses the green wire when the signal light is already red, and then crossing the red wire will close the event.

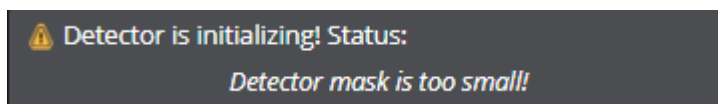
Steps to add a Red Stop detector:

1. Click on the new detector button and select Red Stop detector.
2. A green mask appeared on the live image. Drag this mask onto the traffic light and align it as accurately as possible. Then right-click on it.
3. The stop bar should then be added to the live image. Left-click on the image to pick up the first point. This will appear as a blue dot. Left-click again to pick up the next point and the green line will be drawn. You can draw the line to indicate the line-up by picking up more points, when you are done right-clicking.

4. You must then draw a line on the live image indicating the exit from the traffic junction. Again, left-click on the image. This will appear as a blue dot. To add the next point, left-click again and the red line will be drawn. You can draw the line by picking up more points, when you are done right-clicking.
5. If you have drawn all the lines listed so far, you can then modify them. Just left click on the line and you can move the points you have added.
6. Select the type of traffic light.
7. Set the tolerance time, which refers to the time that should elapse after the light turns red before the first violation event is recorded.
8. In the Tracked object section, you can specify which object types are to trigger events.
9. You can set a configuration value, objects with a configuration below the set value will not generate an event.
10. . When all settings are complete, click Save button.

After the Red Stop detector is set up and saved, the detector is initializing. The detector then searches for the light in the selected area. During the initialisation, a blue "i" will appear on the detector section. When the initialisation is complete, the blue "i" disappears. Then the detector is ready.

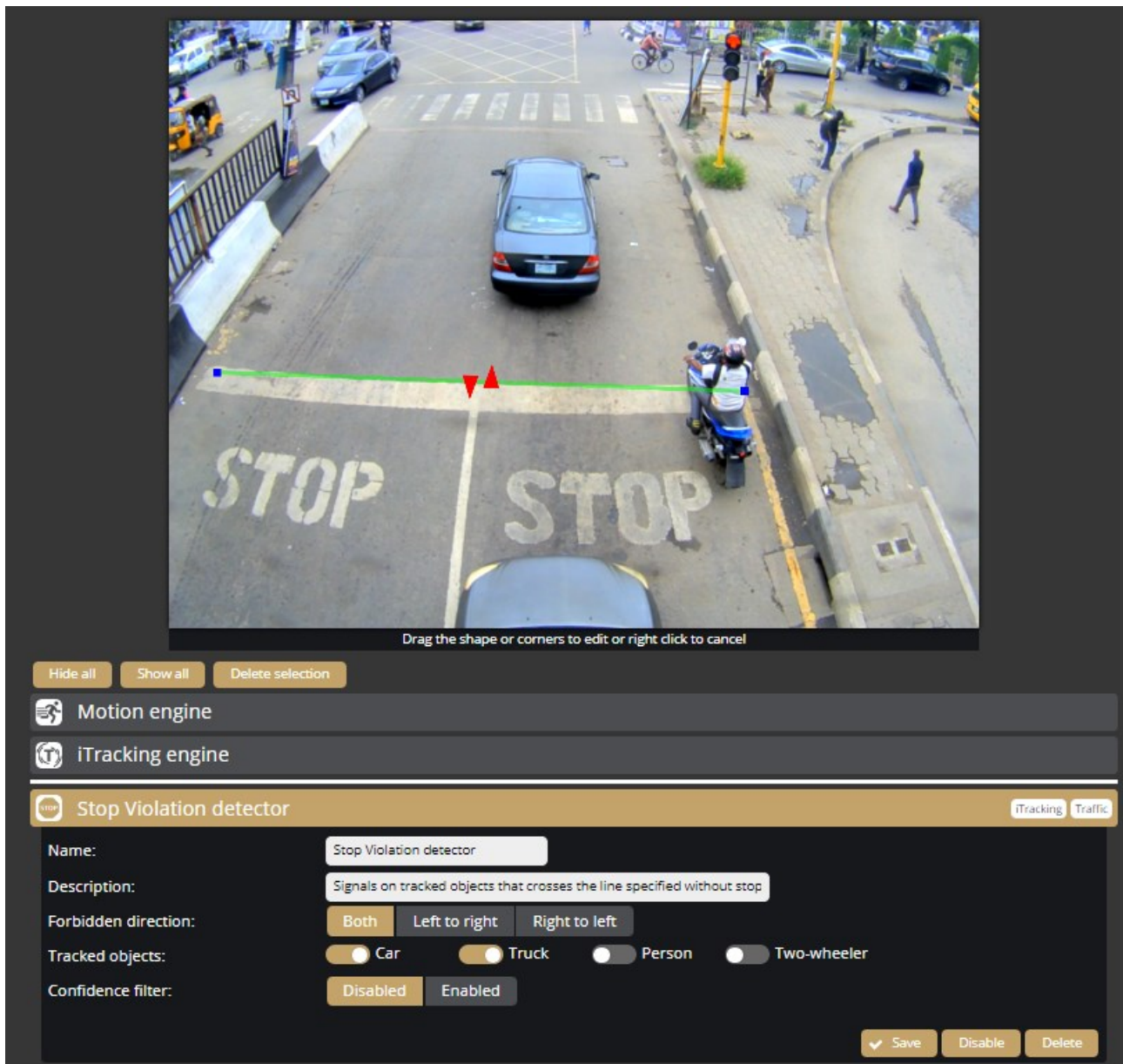
If the mask drawn on the lamp is too small, the following warning appears after saving:



#### Note

If monitoring multiple lanes, create one Red Stop detector per lane. You can distinguish the detectors by changing the Name and Description field.

## 7.12.4.6 Stop Violation detector



Drag the shape or corners to edit or right click to cancel

Hide all Show all Delete selection

Motion engine

iTracking engine

Stop Violation detector

Name: Stop Violation detector

Description: Signals on tracked objects that crosses the line specified without stop

Forbidden direction: Both Left to right Right to left

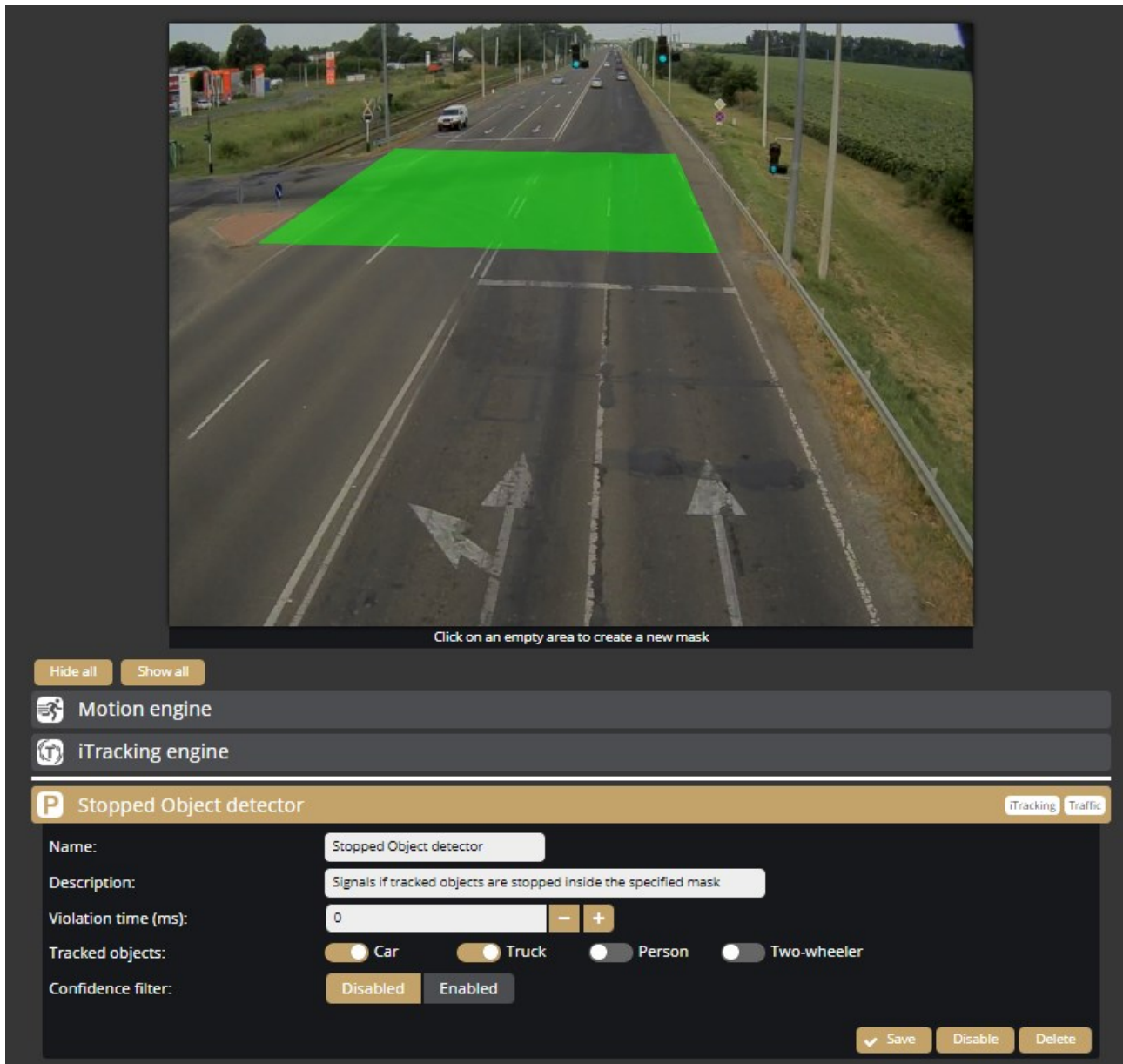
Tracked objects: Car Truck Person Two-wheeler

Confidence filter: Disabled Enabled

Save Disable Delete

The STOP Violation detector detects vehicles that cross the marked wire without stopping. In this case a wire must be drawn on the image. The detector can be set to record only one direction or the both directions.

## 7.12.4.7 Stopped Object detector



Click on an empty area to create a new mask

Hide all Show all

Motion engine

iTracking engine

**P Stopped Object detector** iTracking Traffic

Name: Stopped Object detector

Description: Signals if tracked objects are stopped inside the specified mask

Violation time (ms): 0 - +

Tracked objects: ☒ Car ☒ Truck ☐ Person ☐ Two-wheeler

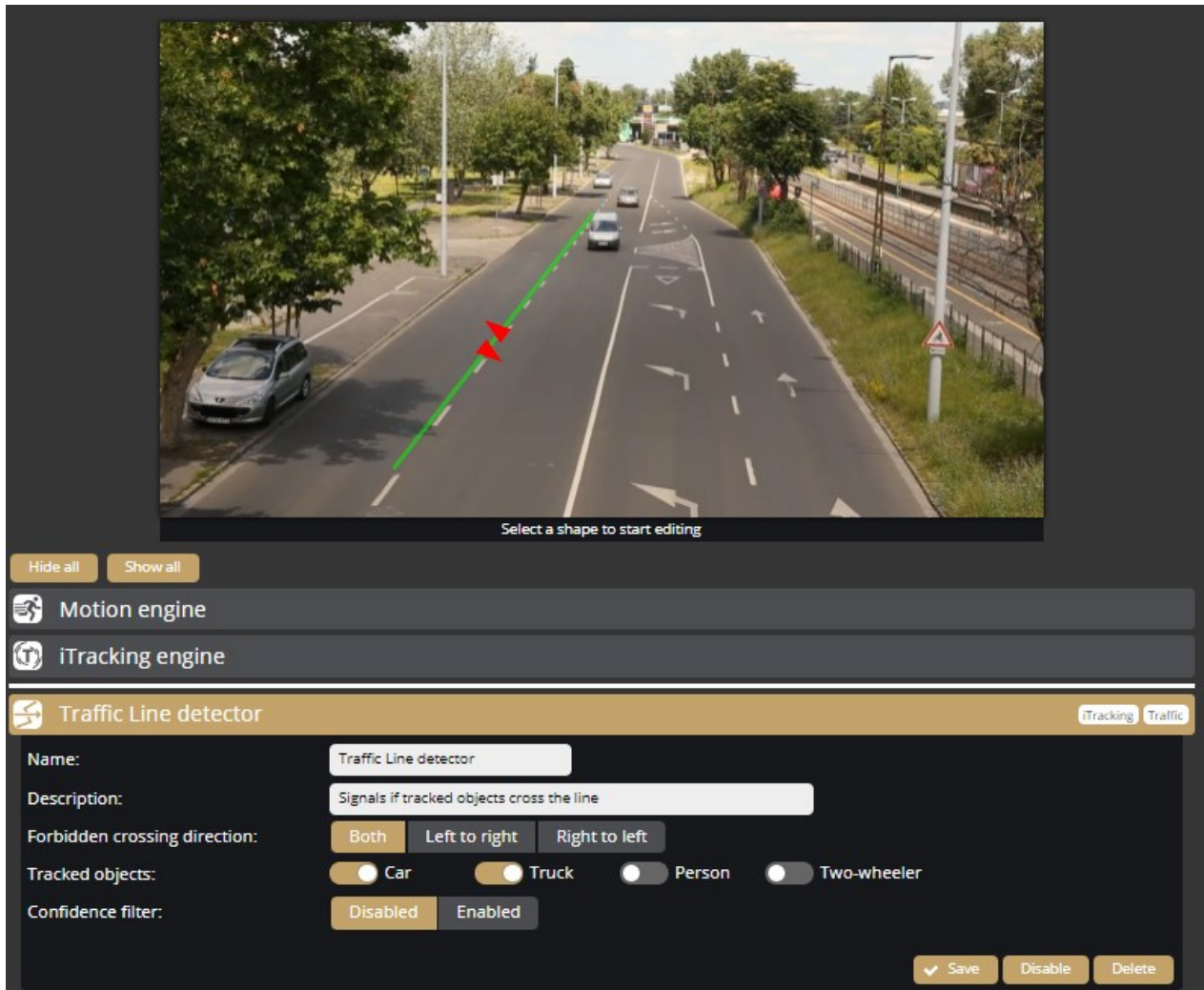
Confidence filter: Disabled Enabled

Save Disable Delete

The Stopped Object Detector detects vehicles that are stopped in the selected area. You must draw a mask that covers the area to be scanned in the image. By specifying Violation time, only vehicles that stay in the selected area for longer than the specified time will be detected. A detector can cover several separate areas in the image, to do this it must draw several masks on the image.



## 7.12.4.8 Traffic Line detector



Select a shape to start editing

Hide all Show all

Motion engine

iTracking engine

Traffic Line detector iTracking Traffic

Name: Traffic Line detector

Description: Signals if tracked objects cross the line

Forbidden crossing direction: Both Left to right Right to left

Tracked objects: ☒ Car ☒ Truck ☐ Person ☐ Two-wheeler

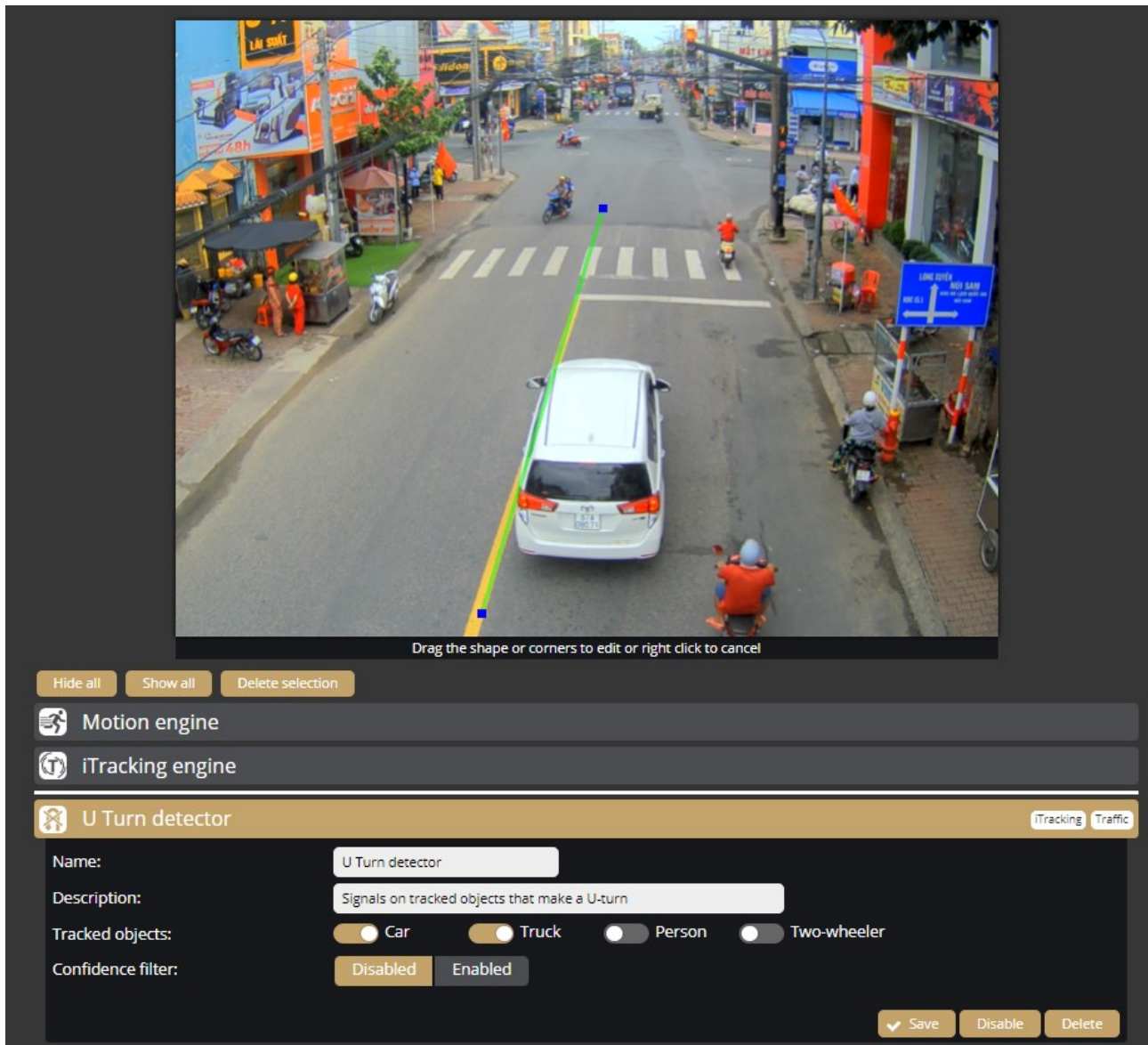
Confidence filter: Disabled Enabled

Save Disable Delete

The Traffic Line detector detects vehicles that cross the designated line. In this case, a wire must be drawn on the image. The detector can be configured to record only one or the both directions.

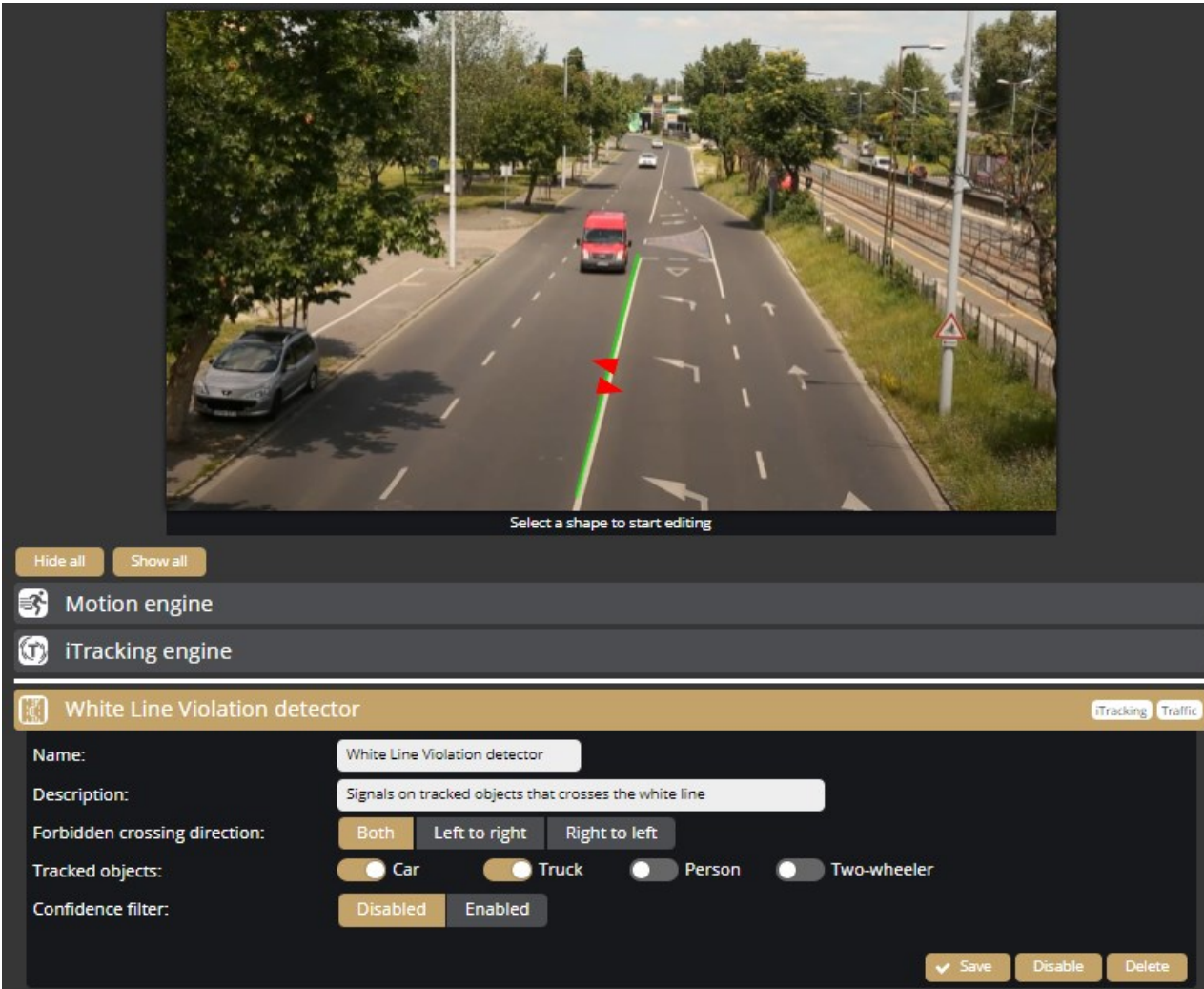


## 7.12.4.9 U Turn detector



The U Turn detector detects the vehicle that passes parallel to the drawn wire in both directions.

## 7.12.4.10 White Line Violation detector



Select a shape to start editing

Hide all Show all

Motion engine

iTracking engine

White Line Violation detector iTracking Traffic

Name: White Line Violation detector

Description: Signals on tracked objects that crosses the white line

Forbidden crossing direction: Both Left to right Right to left

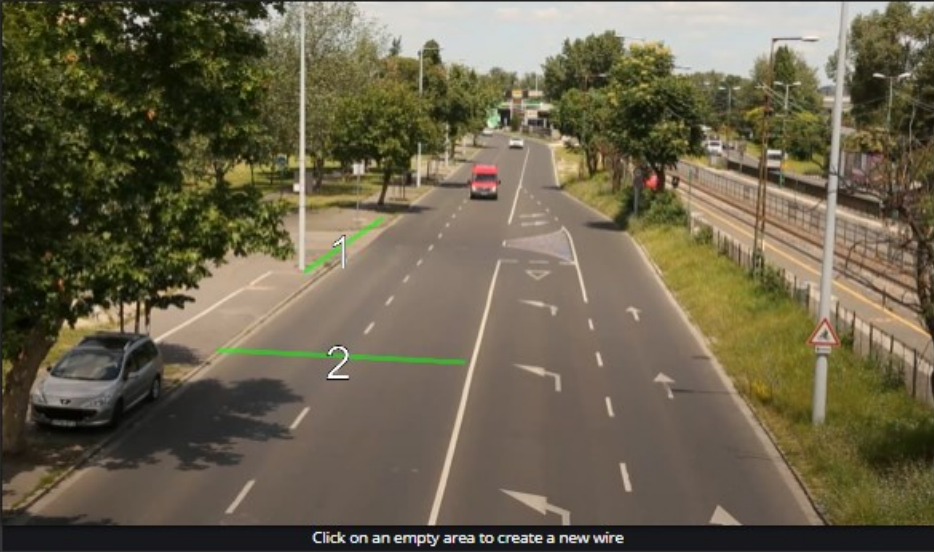
Tracked objects: ☒ Car ☒ Truck ☐ Person ☐ Two-wheeler

Confidence filter: Disabled Enabled

Save Disable Delete

The White Line Violation detector detects vehicles that cross the marked white line. In this case a wire must be drawn on the image. The detector can be configured to record only one or both directions.

## 7.12.4.11 Wrong Turn detector



Click on an empty area to create a new wire

Hide all Show all

Motion engine

iTracking engine

**Wrong Turn detector**

Name: Wrong Turn detector

Description: Signals on tracked objects that turn the wrong way

Tracked objects: ☒ Car ☒ Truck ☐ Person ☐ Two-wheeler

Confidence filter: ☒ Disabled ☐ Enabled

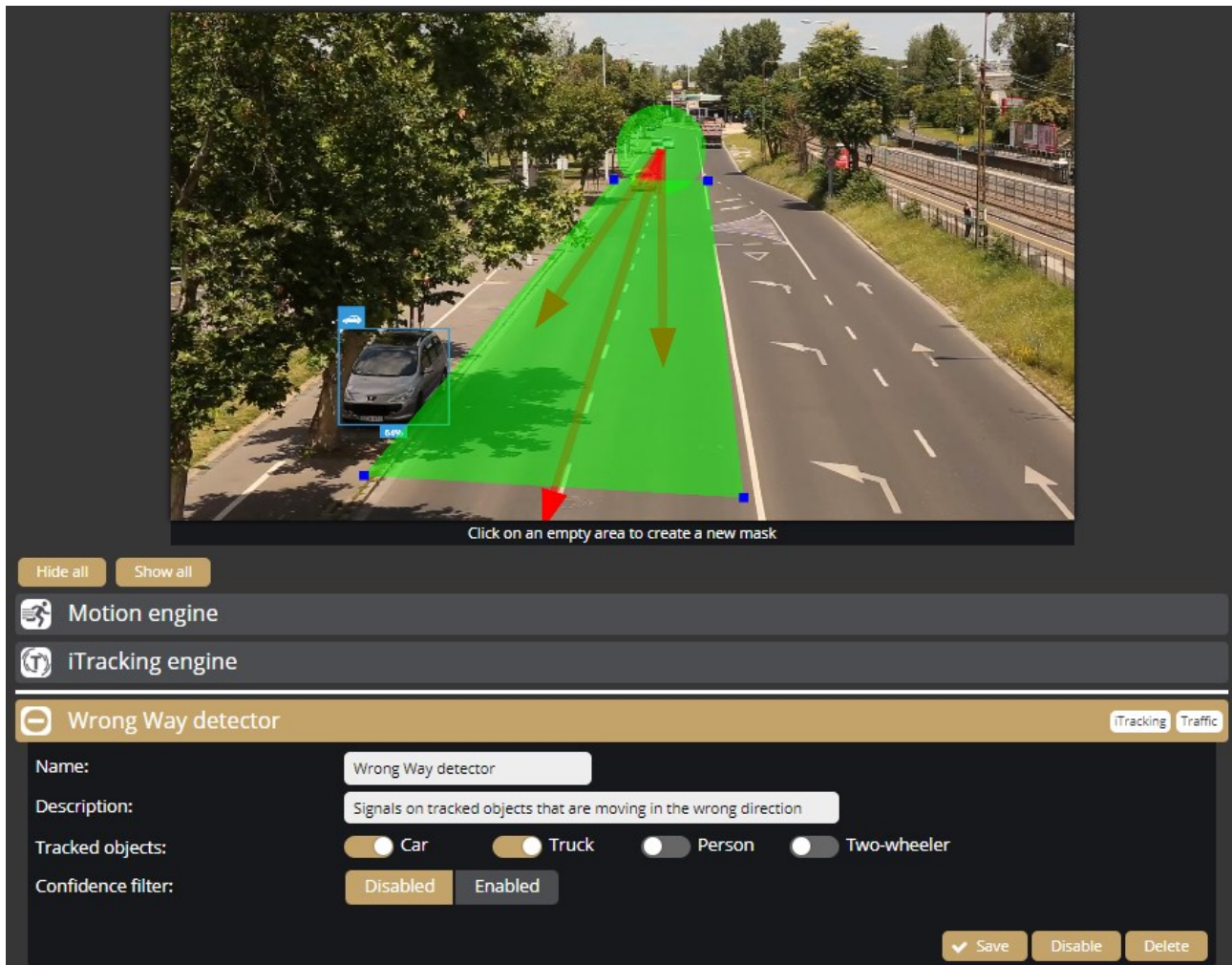
⚠ Detector is not configured!

Save Disable Delete

The wrong turn detector detects vehicles that are turning in the wrong direction. In this case 2 wires should be drawn on the image. The first wire (1) should cross the lane from which the vehicles under test are coming. The second wire (2) shall cross the lane where the vehicles are not allowed to go. The vehicle is detected if it crosses both wires.



## 7.12.4.12 Wrong Way detector



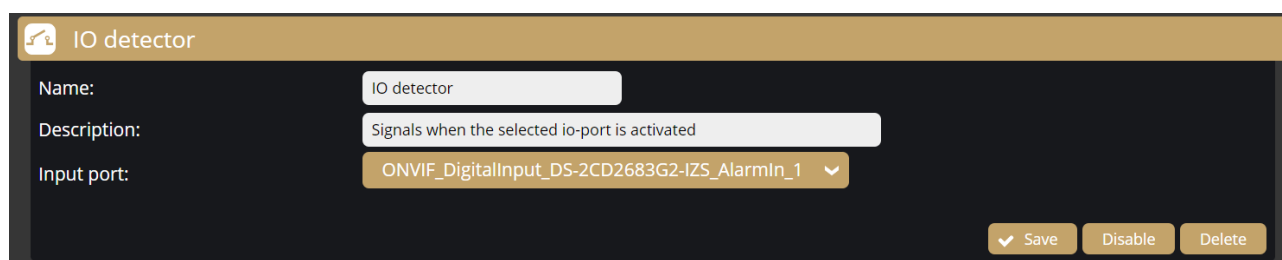
The wrong way detector detects vehicles moving in the wrong direction on the lane. When the detector is picked up, arrows appear with a common starting point. The arrows can be rotated around the starting point. The arrows are used to mark the good direction on the lane. You also need to designate an area where you want to detect vehicles moving in the wrong direction. The vehicle is detected when it is moving in the opposite direction as the arrows and through the marked area.

### 7.12.4.13 IO detector

The IO detector can be used to create events based on the input signals from the added ONVIF device. You can add a new ONVIF device in the System/ External menu.

The following can be adjusted on the **IO detector** interface:

- **Name:** The name of the detector can be entered.
- **Description:** A brief description can be added to the detector.
- **Input port:** The selectable the added ONVIF device



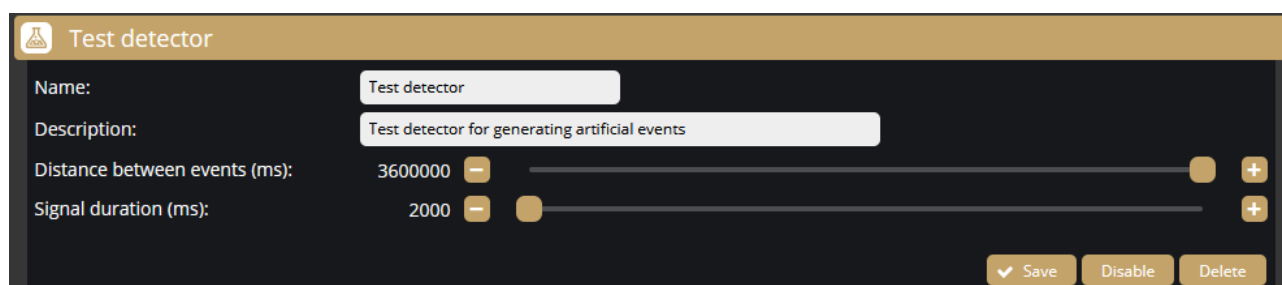
The screenshot shows the 'IO detector' configuration window. It has a title bar with a folder icon and the text 'IO detector'. Below the title bar, there are three fields: 'Name:' with the value 'IO detector', 'Description:' with the value 'Signals when the selected io-port is activated', and 'Input port:' with a dropdown menu showing 'ONVIF\_DigitalInput\_DS-2CD2683G2-IZS\_AlarmIn\_1'. At the bottom right, there are three buttons: 'Save', 'Disable', and 'Delete'.

### 7.12.4.14 Test detector

The Test detector can be used to test the device's upload and storage capabilities. We recommend to use it for testing purposes only, not to put unnecessary load to the SD card and the device.

The following can be adjusted on the **Test detector** interface:

- **Name:** The name of the detector can be entered.
- **Description:** A brief description can be added to the detector.
- **Distance between events:** The device produces a test event at the set intervals.
- **Signal duration:** To adjust the signal length.



The screenshot shows the 'Test detector' configuration window. It has a title bar with a flask icon and the text 'Test detector'. Below the title bar, there are three fields: 'Name:' with the value 'Test detector', 'Description:' with the value 'Test detector for generating artificial events', and two sliders: 'Distance between events (ms):' with a value of 3600000 and 'Signal duration (ms):' with a value of 2000. At the bottom right, there are three buttons: 'Save', 'Disable', and 'Delete'.

## 8. HOW TO USE THE ENFORCE BOX

This chapter gives you a quick overview of using Enforce Box and what to look out for when installing and operating the device.

### 8.1. DEVICE INSTALLATION

1. Mount the device into a weatherproof cabinet near the connected camera or place it in your server room. Consider the temperature tolerance and IP protection of the device (-30°C to +60°C; IP40).
  - a. Provide uninterruptible power to the device's **power supply**. The Enforce Box device cannot work with PoE power supply. However it can provide PoE power to the connected camera.
  - b. Use the **LAN port** on the device to connect the device to your network.
  - c. Use the **PoE port** on the device to power and communicate with the connected camera. If your camera is far from the device on the network, and it is not connected to the device directly, use either port.
  - d. To **store events** in the device, plug an USB drive into the device's USB3 port or use microSD card inserted inside the device. The device includes an 8 GB microSD card.
  - e. Enforce Neural Network Controller is installed in the device (miniPCIe format). Do not plug more Enforce NNC (eg. USB dongle) into the device.
2. **Find the device on the network**, then access the device's web interface
  - a. The default IP address is shown on the sticker at the bottom of the device. However, if a DHCP server is available on the network, the device will also get an IP address from the DHCP server. You can access the device from both IP addresses.
  - b. Optionally, you can use the **AR Device Tool** to locate the device on your local network *{AR Device Tool}*.
  - c. You can access the web interface of the device with the **admin/admin** username/password pair. It is strongly recommended to change the default password.
3. Some simple but important basic settings:
  - a. Upload the **License Key** to your device *{System / Device}* according to the email received when you purchased the device. You will also need to update the License Key for updates after a year. You need to see in the *{System / Status}* menu that your License key is present and your licence corresponds to the desired region (push the „Traffic License“ button and the popup window shows the validity time of the license).

- b. Check that the **device has the latest firmware**. You can download the latest FW from [Adaptive Recognition website](#) or use AR Device Tool. Upload it in **{System / Device}**.
  - c. Check/set the **device time {System / Device}**.
  - d. Set up **storage** and **event upload**.
4. Video stream setting and detector setting:
  - a. In the **Video** section, enter the RTSP source manually or select it from the drop-down list. The ONVIF devices that you added in the External menu are listed in the drop-down list. After saving, the stream of the selected device will start on the live view.
  - b. In the **Detectors** section, click on the New detector button to select and set the type of detector you want. For a description of how to set up the detectors, see Chapter 7.

## 8.2. RED STOP DETECTOR AND NETWORK SETTINGS

1. Perform the steps described in chapter 8.1.
2. In the Video menu, add the stream of the overview camera or, in the case of a dual-sensor camera, the overview sensor's stream and save the settings. Vidar camera RTSP source format: `rtsp://Camera_IP/stream/h264`
3. In the System/Device menu, set the time and turn on the NTP server.
4. Also set the correct time in the Basic Setup/Date and Time menu in the Vidar GUI interface of the Vidar camera. Enter the IP address of the EnforceBox in the NTP server hostname/IP field. Save the data and the camera will restart. This setting is necessary so that the time synchronization of the two devices will not slip. The additional setup descriptions only apply to the EnforceBox. For further information on setting up the Vidar camera, please refer to the Vidar User Manual.
5. Under System/ Detectors, add the Red Stop detector as shown below:
  - a. Click on the new detector button and select Red Stop detector.
  - b. A green mask appeared on the live image. Drag this mask onto the traffic light and align it as accurately as possible. Then right-click on it.
  - c. The stop bar should then be added to the live image. Left-click on the image to pick up the first point. This will appear as a blue dot. Left-click again to pick up the next point and the green line will be drawn. You can draw the line to indicate the line-up by picking up more points, when you are done right-clicking.
  - d. You must then draw a line on the live image indicating the exit from the traffic junction. Again, left-click on the image. This will appear as a blue dot. To add the next point, left-

- click again and the red line will be drawn. You can draw the line by picking up more points, when you are done right-clicking.
- e. If you have drawn all the lines listed so far, you can then modify them. Just left click on the line and you can move the points you have added.
  - f. Select the type of traffic light.
  - g. Set the tolerance time, which refers to the time that should elapse after the light turns red before the first violation event is recorded.
  - h. In the Tracked object section, you can specify which object types are to trigger events.
  - i. You can set a configuration value, objects with a configuration below the set value will not generate an event.
  - j. When all settings are complete, click Save button.
  - k. After saving, wait for a few lights to change in the live image and use Overlays to make sure that the Red Stop detection works correctly.
5. Add the ANPR Network detector setup steps :
- a. Click on the new detector button and select ANPR Network detector.
  - b. Enter the IP address of the Vidar camera
  - c. On the image, draw the area that the Vidar camera's ANPR sensor sees. This will be the area covered by the green mask. This area can be added by left-clicking on the live image, a new point is added after each click and the mask is drawn after 3 points are added. You can finish drawing by right-clicking.
  - d. Save the detector
  - e. After saving, you can see the data received by the ANPR Network detector from the Vidar camera in the Event menu.
6. You have completed the previous steps to make the necessary settings. When a vehicle passes through the red light, the Red Stop detector will generate an event and you will see the associated ANPR data in the Related Event field.



## 9. API DOCUMENTATION

### 9.1 INTRODUCTION

This document is the API specification of the Camen Box devices starting from firmware version 1.3.0.

Multiple types of APIs are available - all accessed through HTTP protocol - but the main focus of this document is the command API and any further reference to APIs without specifying the type refers to the command API only.

API requests may accept input parameters in the HTTP REQUEST BODY as a JSON formatted text and the device replies with data in the HTTP RESPONSE BODY as a JSON formatted text. A command can be executed by sending a HTTP POST request to the appropriate URL.

**Note:** API functions and properties not covered by this document may be changed or removed in the future without notice

#### 9.1.1. Legend

The following is a list of expressions used in this document:

DEVICE_IP	The IP address or network hostname of the device
REQUEST	A HTTP request sent by the user to the device
RESPONSE	A HTTP response sent by the device to a REQUEST
HTTP BODY	Body part of a HTTP message (see <a href="http://en.wikipedia.org/wiki/HTTP_body_data">http://en.wikipedia.org/wiki/HTTP_body_data</a> )
EXCEPTION	A response given by the device when an error occurred

## 9.2 AUTHENTICATION

Accessing resources on the device requires an authenticated session.

### 9.2.1 Login

To acquire a session the client must use the Login command available at

```
http://DEVICE_IP/login
```

and supply the User and Password of the selected user account. Example login request to the device at 192.168.1.101:

```
POST /login HTTP/1.1
Host: 192.168.1.101
Content-Length: 35
Content-Type: application/json
```

```
{"User": "myusername", "Password": "myuserpassword"}
```

On successful login the device will respond with a JSON object with a single field called **sid** that contains the unique session identifier of the authenticated session.

Example login reply **body** of a successful login:

```
HTTP/1.1 200 OK
Cache: no-cache
Content-Type: application/json
Content-Length: 61

{
  "Type": "Response",
  "Data": {
    "sid": "60ab2b6b"
  }
}
```

Using the wrong username or password will result in an **InvalidCredentialException** error.

After successfully acquiring a session ID the rest of the device API can be accessed by sending the session id as a GET or COOKIE variable under the name **sid**.

### 9.2.2 Session lifetime

A session will time out if the user logs out, no new authenticated connections are initiated for a long period of time or the device reboots. Already active and authenticated connections are kept open even when the associated session ends.

### 9.2.3 Logout

Termination of a session is done by invoking the logout command at

```
http://DEVICE_IP/logout
```

with the session id (sid) sent as a COOKIE or a GET variable. This command will always succeed even if the session identifier is invalid.

Example logout request for session with sid 60ab2b6b:

```
POST /logout?sid=60ab2b6b HTTP/1.1
```

```
Host: 10.10.22.234
```

```
Connection: keep-alive
```

```
Content-Length: 2
```

```
Content-Type: application/json
```

```
{}
```

### 9.2.4 Sessionless access

URLs may be accessed without an active session by providing credentials with each request. The username and password values may be sent with the appropriate **user** and **password** GET parameters.

```
http://DEVICE_IP/SOME/PATH/ON/DEVICE?user=USERNAME&password=PASSWORD
```

Credentials may also be sent using HTTP basic access authentication. Below is an example call using the popular cURL command line tool.

```
curl -v "http://USERNAME:PASSWORD@DEVICE_IP/SOME/PATH/ON/DEVICE"
```

The device does respond with authentication headers by default. Setting the **challenge** GET parameter to 1 on any device URL will force the device to issue a challenge with proper headers when an authenticated resource is requested or the authentication fails.

```
http://DEVICE_IP/SOME/PATH/ON/DEVICE?challenge=1
```

**Note:** It is strongly recommended to use the session based authentication method. Sessionless access is provided for easy access while experimenting with APIs

## 9.3 EXECUTING COMMANDS

### 9.3.1 Accessing the API

The core functionality of the device can be accessed through the API URL which is

```
http://DEVICE_IP/api
```

The available methods are grouped into categories. Each category has a set of methods that can perform an action on the device or query the device for information.

To execute a method the client must invoke the full URL representing it which is as follows:

```
http://DEVICE_IP/api/CATEGORY/METHOD_NAME
```

For example the **GetDevice** method of the **System** category is executed by sending a request to the following URL:

```
http://DEVICE_IP/api/System/GetDevice
```

**Note:** The API requires an authenticated user. The request must include a valid session identifier in the COOKIE or GET variable named **sid**

### 9.3.2 Input/output parameters

Every method's specification may include a **request** and/or a **response** object. These define the input and output parameters of the call. A request object is sent the same way as the login data: as a serialized JSON object in the HTTP POST BODY. The response data is encapsulated in an another layer and contains the response to the method call.

**System/RunTest** is a dedicated command for testing the API with example requests and responses below.

**Note:** The response may contain additional undocumented top level keys beside **Type** and **Data** that can be safely ignored

### 9.3.3 Successful request

We send a RunTest request to the device with the text "First test" and **ThrowException** set to false.

```
POST /api/System/RunTest?sid=951a6d59 HTTP/1.1
```

```
Host: 192.168.1.100
```

```
Connection: keep-alive
```

```
Content-Type: application/json
```

```
Content-Length: 49
```

```
{ "Text": "First test", "ThrowException": false }
```

The device will respond with the following HTTP response:

```
HTTP/1.1 200 OK
```

```
Cache: no-cache
```

```
Content-Type: application/json
```

```
Content-Length: 115
```

```
{
  "Type": "Response",
  "Data": {
    "Text": "Input received: First test",
    "Size": 10,
    "User": "admin"
  }
}
```

The **"Type": "Response"** indicates that our request was successful and the device executed the method and replied with data.

The cURL command-line tool may be used to send the above request using the following call:

```
curl \
-X POST \
-H 'Content-Type: application/json' \
-d '{ "Text": "First test", "ThrowException": false }' \
http://192.168.1.100/api/System/RunTest?sid=951a6d59
```

## Failed request with exception

We send a RunTest request to the device with the text "Second test" and ThrowException set to true forcing the device to respond with a TestException.

```
POST /api/System/RunTest?sid=951a6d59 HTTP/1.1
```

```
Host: 10.10.22.234
```

```
Connection: keep-alive
```

```
Content-Length: 44
```

```
{
  "Text": "Second test",
  "ThrowException": true
}
```

The device will respond with the following exception:

```
HTTP/1.1 200 OK
Cache: no-cache
Content-Type: application/json
Content-Length: 150

{
  "Type": "Error",
  "Data": {
    "ExceptionClass": "TestException",
    "ErrorMessage": "This is a test exception for testing error reporting."
  }
}
```

## 9.4 DATA TYPES

The JSON format allows transfer of several data types but is limited compared to high-level programming languages. The reference of structures used in the device API contains a **Type** field that specifies the real data structure behind the items. The device will try to convert any input to the expected type or ignore the value on conversion failure.

### 9.4.1 Boolean

The **bool** type represents a boolean with a true or false value. This type can accept JSON booleans, literal "true" or "false" (case-insensitive) strings and numbers as well.

### 9.4.2 Integers

The **int8**, **int16**, **int32** and **int64** types represent integers with a fixed bit width. If the input value doesn't fit into the specified bit length then it will be discarded.

**Note:** When sending **int64** types keep in mind that some implementations cannot represent large 64 bit numbers. The device parses any string input as number when a numeric type is expected so it is recommended to send large numbers as strings.

### 9.4.3 Timestamps

The timestamp information is usually handled as an **int64** number representing a UTC timestamp in milliseconds. The epoch of the timestamp is

Monday, January 1, 1601 12:00:00 AM

also known as Windows epoch.

```
POSIX_TIME_IN_MS + 11644473600000 = WINDOWS_TIME_IN_MS
WINDOWS_TIME_IN_MS - 11644473600000 = POSIX_TIME_IN_MS
```

#### 9.4.4 Double

The **double** type represents a standard (IEEE 754) 64 bit double-precision number.

#### 9.4.5 GUID

The **guid** type is a string with a fixed format of {XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX} where X is a hexadecimal digit (0 to F).

#### 9.4.6 Arrays of integers

Some methods require a long list of numbers (e.g.: coordinates). For this case there is an **Array** type that holds integers. The JSON array type is equivalent with this except **Array** can only contain numbers.

#### 9.4.7 Unnamed keys

There are cases when the sequence of data that must be sent does not have any identifier (key). For this case the API handles numeric keys as unnamed keys. Any entry with a numeric key is considered unnamed and will be parsed accordingly. The actual number used does not make any difference since the numeric keys are not interpreted but the placement order of the elements are preserved.

An example of an object with named (Test1 & Test2) and unnamed (28, 91 & 4) keys:

```
{
  "91" : "Unnamed entry with arbitrary numeric key",
  "Test1" : "Named entry which will be the second in the list",
  "28" : "Another unnamed entry",
  "4" : "Third unnamed entry",
  "Test" : "Another named entry which is the last in the list (5th)"
}
```

#### 9.4.8 Lists

The **List** type contains elements of the same type with unnamed keys.

#### 9.4.9 Map

The **Map** type contains elements of the same type with named keys.

## 9.5 COMMAND OPTIONS

Certain structures' parameters are limited to numeric ranges or a list of possible values. These possible values are called **Options**.

Structures with **Options** are commonly used in get/set method pairs (like **System/GetNtpSettings** and **System/SetNtpSettings**). When a command pair contains options the setter command will only accept data that fit the restrictions specified by the options in the getter command. Values outside of the specified boundaries will be ignored.

If an **Option** item is present inside the **Data** field of the response then its structure will be the exact copy of the **Data** structure where instead of the normal types and structures, there will be **Option-NumericRange** and **OptionValueList** structures describing the allowed values for each entry. The **Option** structure is ready-only and can be omitted when calling the appropriate setter command.

Example:

```
{
  "Type": "Response",
  "Data": {
    "TestItem": 12,
    "TestList": "Item1", "Options": {
      "TestItem": {
        "Default": 50,
        "Minimum": 0,
        "Maximum": 100
      },
      "TestList": {
        "Default": "Item0", "Values": {
          "0": "Item0",
          "1": "Item1",
          "2": "Item2",
          "3": "Item3",
          "4": "Item4",
          "5": "Item5",
          "6": "Item6",
        }
      }
    }
  }
}
```



The above example structure describes a response where two items are present: **TestItem** and **TestList**. The **Options** entry is present so there are restrictions on what can be set for **TestItem** and **TestList**.

- **TestItem** has a default value of 50 and accepts anything from 0 to 100
- **TestList** has a default value of "Item0" and accepts any of the elements listed under "Values"

**Note:** The limits imposed by options are different from device to device based on product type and activesettings

## 9.6 FEATURES

Devices have different features available to the user based on product type and hardware configuration. These features can be queried using the **System/GetDevice** command. The response contains a map of modules under the **Modules** name with descriptors for each modules' capabilities. A descriptor may also contain a tree of strings defining available features. Feature lists are fixed and will not change unless the device is restarted.

### 9.6.1 Common modules

Module	Functionality	Module descriptor
Analytics	Detectors and events	<b>ModuleAnalytics</b>
IO	External I/O ports	<b>ModuleIO</b>
Media	Audio and video streams	<b>ModuleMedia</b>

## 10. DETECTORS & ENGINES

### 10.1. TYPES

The analytics module is divided into **engines** and **detectors**.

**Engines** are core modules running highly specialized algorithms and provide processed data sets for detectors to analyse. Engines do not emit events and don't provide user-queryable output. Depending on the device configuration the following engines may be available:

Engine	Description
ANPR engine	Performs license plate recognition (see <a href="#">Analytics/GetAnprEngine</a> )
iTracking engine	Marks and tracks moving objects (see <a href="#">Analytics/GetTracker</a> )
Motion engine	Performs motion detection on the whole image

**Detectors** are algorithms that analyze one or more data sets, media streams or peripherals and emit events when algorithm-specific criterias are met. For the events' properties see the **Event** structure. Depending on the device configuration the following detectors may be available:

Detector	Reference
AlarmDetectorIO <i>Input port monitor</i>	DetectorConfigurationIO EventIO
AlarmDetectorTest <i>Detector for API testing</i>	DetectorConfigurationTest EventTest
<b>For ANPR devices only:</b>	
AlarmDetectorANPR <i>License plate detection</i>	DetectorConfigurationANPR EventANPR
<b>For Enforcement devices only:</b>	
AlarmDetectorEmergencyLane <i>Emergency lane violation</i>	DetectorConfigurationEmergencyLane EventEmergencyLane
AlarmDetectorForbiddenZone <i>Forbidden zone violation</i>	DetectorConfigurationForbiddenZone EventForbiddenZone
AlarmDetectorLane <i>Lane movement</i>	DetectorConfigurationLane Event- Lane
AlarmDetectorRedStop <i>Traffic light violation</i>	DetectorConfigurationRedStop Event- tRedStop
AlarmDetectorStoppedObject <i>Prohibited stop detection</i>	DetectorConfigurationStoppedObject EventStoppedObject
AlarmDetectorStopViolation <i>Stop sign violation</i>	DetectorConfigurationStopViolation EventStopViolation
AlarmDetectorTrafficLine <i>General line crossing</i>	DetectorConfigurationTrafficLine EventTrafficLine
AlarmDetectorUTurn <i>Illegal U-turn detection</i>	DetectorConfigurationUTurn Event- tUTurn
AlarmDetectorWhiteLineViolation <i>White line violation</i>	DetectorConfigurationWhiteLineViolation Event- WhiteLineViolation
AlarmDetectorWrongTurn <i>Illegal turn violation</i>	DetectorConfigurationWrong- Turn EventWrongTurn
AlarmDetectorWrongWay <i>Wrong-way driving detection</i>	DetectorConfigurationWrong- Way EventWrongWay

## 10.2. GEOMETRY

Some detectors and engines require some form of 2D configuration where polygons and lines define how the images are processed.

### 10.2.1. Coordinate system

The device uses the graphical coordinate system where X values increment to the right and Y values increment downwards. All coordinates are defined in a virtual coordinate system where values are calculated by the following formulas:

```
virtual_x = ( image_x / 16384 + image_width ) / image_width
virtual_y = ( image_y / 16384 + image_width ) / image_width

image_x = ( virtual_x * image_width + 16384 / 2 ) / 16384
image_y = ( virtual_y * image_width + 16384 / 2 ) / 16384
```

### 10.2.2. Geometry objects

The following is a list of common shapes for configuring detectors:

Name	Data type	Description
Straight line	<b>GeometryLineSegment</b>	Straight line with two points defining the start and end of the line
Segmented line	<b>GeometryLine</b>	Segmented line with at least one segment, each consisting of a start and end point
Ordered segmented lines	<b>GeometryLineGroups</b>	Groups of segmented lines where an order of groups is formed using indices
Rectangle	<b>GeometryRectangle</b>	Rectangle where each side is parallel to the x or y axis of the image
Polygons	<b>GeometryPolygons</b>	List of polygons. A polygon has at least 3 points and an arbitrary shape.

## 11. EVENTS

### 11.1. MODES

Devices support multiple modes for acquiring emitted events.

**Live event query** is a polling based event download where the user has to periodically check if new events are available.

Pros	Cons
Moderate latency	Event loss on slow connection
Device buffers events	No image or video content

**Live event stream** is a continuous multipart HTTP stream where new events are automatically streamed to the client with accompanying images.

Pros	Cons
Low latency	Event loss on connection error
Event image available	Event loss on slow connection
	No video content

**Stored event query** is a similar mode to the live event query but uses requires a storage device. Supports filtering by detector and metadata.

Pros	Cons
Event image and video available	Requires storage device
Advanced filtering	Significant latency
	Client implementation may be complex

**Stored event upload** supports GDS and HTTP/HTTPS uploading of stored events to a remote server. The HTTP variant uses multipart POST requests to stream events with accompanying media data.

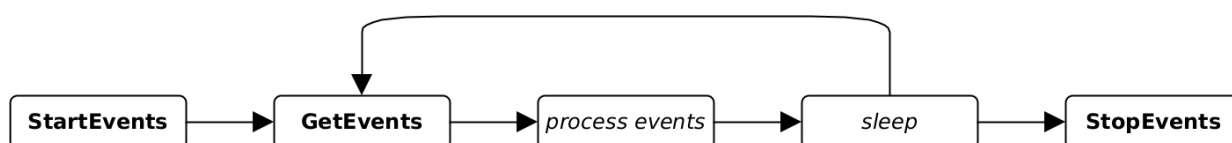
Pros	Cons
Event image and video available	Requires storage device
Event region of interest image available	Significant latency
Compatible with most HTTP server implementation	

## 11.2. LIVE EVENT QUERY

The easiest method of querying events is to poll the events using the **Analytics/GetEvents** call. To start polling initiate a buffer on the device using the **Analytics/StartEvents** call. This tells the device to allocate a buffer for the session and start queueing emitted events.

After the buffer is initiated the **Analytics/GetEvents** call can be used to periodically download collected events and flush the buffer. It is recommended to wait at least a second between two calls to prevent resource exhaustion or activation of the device's DoS protection.

When events are no longer needed the polling can be aborted using the **Analytics/StopEvents** call.



## 11.3. LIVE EVENT STREAM

Live events can be continuously downloaded by sending an authenticated GET request to the device on

```
http://DEVICE_IP/live/events
```

The device will respond with a **multipart/mixed** type connection and start sending events and associated images as they are emitted.

**Events** are sent with the multipart Content-Type of **application/json**. Additional headers include:

<b>X-Event-Index</b>	Index incrementing by one for each event. A gap in the indices means the device was unable to send a packet probably due to slow connection and buffer limitations and dropped the event
<b>X-Timestamp</b>	Posix UTC timestamp of the event in milliseconds

**Images** are sent with the multipart Content-Type of **image/jpeg**. Additional headers include:

<b>X-Image-Index</b>	Index incrementing by one for each image. A gap in the indices means the device was unable to send a packet probably due to slow connection and buffer limitations and dropped the image
<b>X-Frame-Id</b>	ID of sensor frame from which this JPEG was encoded
<b>X-Frame-Timestamp</b>	Monotonic timestamp of the image in milliseconds that is independent of the wall clock and is not affected by clock changes
<b>X-Frame-Width</b>	Image width
<b>X-Frame-Height</b>	Image height
<b>X-Timestamp</b>	Posix UTC timestamp of the image in milliseconds
<b>X-Keep-Alive</b>	Keepalive duration in seconds (see Keepalive below)

The **X-Event-Index** and **X-Image-Index** counters increment by one for each event or image queued respectively. An increment larger than one indicates that the device buffer filled up and data was dropped.

### 11.3.1. Stream format

The stream is in chronological order (except when device time changes) so events with the same timestamp will always be sent together. Images belonging to the events are always sent before the related event and have matching timestamps. If more than one event exists with the same timestamp the image will only be sent once.

The following example demonstrates the order of data when multiple events exist with the same timestamp:

Part #	Type	Source	Timestamp
1	image/jpeg		2021-01-11 19:32:03.978
2	application/json	Detector1	2021-01-11 19:32:03.978
3	application/json	Detector2	2021-01-11 19:32:03.978
4	image/jpeg		2021-01-11 19:39:56.004
5	application/json	Detector2	2021-01-11 19:39:56.004

### 11.3.2. Image attachment

Images can be disabled by setting the GET parameter **image** to zero:

```
http://DEVICE_IP/live/events?image=0
```

### 11.3.3. Resume stream

Network issues may close the connection prematurely and events may be lost while the client is reconnecting. To recover from such scenario the **timestamp** GET parameter can be used to provide the device with a starting point. The device will look up events in its internal buffer and send out any that matches or newer than the timestamp. The unit of timestamp is Windows milliseconds (same as the EventTime property of events).

Using the URL below the client will receive available events starting from 2021-05-14 12:09:41 UTC.

```
http://DEVICE_IP/live/events?timestamp=13265460581098
```

### 11.3.4. Filtering

The stream contains all events from all detectors by default. The events can be filtered by providing a comma separated list of detector ids with the **filter** GET parameters.

Using the URL below the client will only receive events from two detectors with ids {6309907F-5708-47D1-B410-50F02C8882FB} and {B4C797C3-3AF3-4277-194D-9EF952A202A2}.

```
http://DEVICE_IP/live/events?filter={6309907F-5708-47D1-B410-50F02C8882FB},
{B4C797C3-3AF3-4277-194D-9EF952A202A2}
```

### 11.3.5. Keepalive

During quiet periods the device may not transmit any data for a significant amount of time. Many network equipment may detect such connection as stale and close it prematurely.

Set the **keepalive** GET parameter to a duration in seconds to activate the keepalive messages. The device will automatically send an update message with Content-Type of **application/x-keepalive** when no data transfer was detected for the specified duration.

**Note:** The device may override the keepalive parameter if set too low. The actual keepalive duration is always sent back in the X-Keep-Alive HTTP header. A zero value means keepalive is turned off.

Using the URL below the client will receive a keepalive message after a minute without any data transfer:



```
http://DEVICE_IP/live/events?keepalive=60
```

Below is an example update message:

```
--IPCamEventStreamBoundary  
Content-Type: application/x-keepalive  
Content-Length: 0  
  
--IPCamEventStreamBoundary
```

### 11.3.6. Example stream

Example event stream request to the device at 192.168.1.101:

```
GET /live/events HTTP/1.1  
Host: 192.168.1.101  
Connection: keep-alive  
Cookie: sid=60ab2b6b
```

Beginning of the response to the above request that contains one signal event and an image:

```
HTTP/1.1 200 OK
Pragma: no-cache
Expires: Thu, 01 Dec 2003 16:00:00 GMT
Connection: close
Content-Type: multipart/mixed; boundary=IPCamEventStreamBoundary
Cache: no-cache
Accept-Ranges: none
X-KeepAlive: 0
X-Timestamp: 1620986981098
X-Windows-Timestamp: 13265460581098
```

```
-IPCamEventStreamBoundary
Content-Type: image/jpeg
Content-Length: 498749
X-Timestamp: 1620986982002
X-Image-Index: 1
X-Frame-Id: 521699
X-Frame-Timestamp: 757030579
X-Frame-Width: 2560
X-Frame-Height: 1920
```

*binary data*

```
-IPCamEventStreamBoundary
Content-Type: application/json
Content-Length: 308
X-Event-Index: 1
X-Timestamp: 1620986982042
```

```
{
    "DetectorVersion" : 131072,
    "DetectorID" : "{6309907F-5708-47D1-B410-50F02C8882FB}",
    "DetectorClassID" : -835316578,
    "EventTime" : "13265460582042",
    "State" : "dsSignal",
    "EventCode" : 100,
    "EventInfo" : {},
    "EventID" : "{4F34A399-9E02-1846-ADA7-98A2798B46B9}",
    "DetectorEventType" : "detSignal"
}
```

## 11.4. STORED EVENT QUERY

Devices with storage enabled can be queried for stored events using the **Storage/GetEvents** function.

It is recommended to first check the available time range on the storage device using the **Storage/GetStatistics** call then download in moderate segments. Specifying too large durations will result in slow or partial responses (see **Status** in **StorageEvents**).

### 11.4.1. Image

Images related to stored events can be downloaded with the following url:

```
http://DEVICE_IP/playback/image? detector=DETECTOR_ID&event=EVENT_ID&timestamp=EVENT_TIMESTAMP
```

The GET parameters of **DETECTOR\_ID**, **EVENT\_ID** and **EVENT\_TIMESTAMP** correspond to the values of **DetectorID**, **EventID** and **EventTime** from **StorageEvents** respectively.

**Note:** A HTTP status code may be returned when the image is not available.

### 11.4.2. Video

Videos for the stored events may be requested with the following url:

```
http://DEVICE_IP/playback/video?start=START_TIMESTAMP&end=END_TIMESTAMP
```

The GET parameters specify the time range of the video using the same format as **EventTime**.

**Note:** A HTTP status code may be returned when no video content is available in the specified time range.

## 11.5. STORED EVENT UPLOAD

Devices with storage enabled can automatically upload events to an Adaptive Recognition Globessey DataServer (GDS) or a compatible HTTP/HTTPS server. This chapter describes the HTTP/HTTPS mode only.

### 11.5.1. Process

Upon activation the event uploader begins searching for events on the storage device in chronological order. Once an event is found a single standard POST request of **multipart/form-data** type is initiated to the configured URL and all data are transmitted.

### 11.5.2. Error handling

The server must respond with a HTTP status code of 200 for a successful transfer. Other responses are handled as follows:

- When a connection error occurs the uploader will retry indefinitely until the event is no longer available.
- Server may respond with a HTTP status code of 503 or 504 to signal that it is unable to accept requests. The uploader will retry indefinitely until the event is no longer available.
- When any other errors are encountered the uploader will retry a limited number of times then discard the event.

### 11.5.3. Request format

Event data and related media is uploaded in multipart fields identified by their **name**. The name and order of the fields are as follows:

Field name	MIME type	Count	Description
event_timestamp	text/plain	1	Field contains the posix UTC timestamp of the event in milliseconds
event_video_NUM	video/mp4	0≤	Related video content
event_image_NUM	image/jpeg	0≤	Related image content
event_cropped_image_NUM	image/jpeg	0≤	Region of interest cropped out from the original image
event_descriptor	application/json	1	Event descriptor in JSON format (see <b>Event</b> )

The value of *NUM* is a zero-based index (e.g.: event\_image\_0, event\_image\_1, ...).

By default data is sent as standard form-data fields with only a **name** property but - using the web interface - a

**filename** property can be added to media fields (image and video).

**Note:** When using PHP POST fields are accessed through the `$_POST` variable but fields with filenames are available in the `$_FILES` variable

Field header when only names are sent:

```
Content-Disposition: form-data; name="FIELD_NAME"
```

```
Content-Type: MIME_TYPE
```

Field header of media data when filenames are configured as well:

```
Content-Disposition: form-data; name="FIELD_NAME"; filename="FIELD_NAME.EXTENSION"
Content-Type: MIME_TYPE
```

Below is an example event upload transfer between a device and the server at 192.168.1.102 where the server's responses are marked red:

```
POST /http_upload_server_php/ar_http_upload.php HTTP/1.1
```

```
Host: 192.168.1.102
```

```
User-Agent: IntellioHttpPostUploader/1.0Accept: */*
```

```
Cache-Control: no-cache
```

```
Content-Type: multipart/form-data; boundary=IntellioHttpPostUploaderBoundary
```

```
Content-Length: 4330662
```

```
Expect: 100-continue
```

```
HTTP/1.1 100 Continue
```

```
--IntellioHttpPostUploaderBoundary
```

```
Content-Disposition: form-data; name="event_timestamp"Content-Type: text/plain
```

```
1631732906436
```

```
--IntellioHttpPostUploaderBoundary
```

```
Content-Disposition: form-data; name="event_video_0"
```

```
Content-Type: video/mp4
```

```
binary data
```

```
--IntellioHttpPostUploaderBoundary
```

```
Content-Disposition: form-data; name="event_image_0"
```

```
Content-Type: image/jpeg
```

```
binary data
```

```
--IntellioHttpPostUploaderBoundary
```

```
Content-Disposition: form-data; name="event_cropped_image_0"
```

```
Content-Type: image/jpeg
```

```
binary data
```

```
--IntellioHttpPostUploaderBoundary
```

```
Content-Disposition: form-data; name="event_descriptor"
```

```
Content-Type: application/json
```

```
{
  "DetectorVersion" : 131072,
  "DetectorID" : "{7D0829EA-E8FD-7546-92C7-3528E6216CBB}",
  "DetectorClassID" : 1968398405,
  "DetectorClass" : "AlarmDetectorANPR",
  "EventTime" : "13276206506436",
```

```
"State" : "dsNormal",
"EventCode" : 114,
"EventInfo" : {
  "Text" : "ABC123",
  "Confidence" : 0.819999999284744262695,
  "Country" : "BIH",
  "CountryCode" : 113004,
  "Coords" : [
    7808,
    5606,
    8992,
    5632,
    8992,
    5843,
    7808,
    5818
  ],
  "BackgroundColor" : "",
  "DedicatedAreaColor" : "",
  "TextColor" : ""
},
"EventID" : "{93B5A26B-3069-E346-8E89-383ABA7A275C}",
"DetectorEventType" : "detSimpleEvent"
}
--IntellioHttpPostUploaderBoundary--
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: text/html; charset=UTF-8
```



## 12. MISCELLANEOUS

### 12.1. GPIO STATE STREAM

Live I/O state can be continuously downloaded by sending an authenticated GET request to the device on

```
http://DEVICE_IP/live/io
```

The device will respond with a **multipart/mixed** type connection and start sending updates about I/O port statechanges.

I/O state changes are sent with the multipart Content-Type of **application/json**. Additional headers include:

X-Timestamp	Posix UTC timestamp of the state change in milliseconds
X-Keep-Alive	Keepalive duration in seconds (see Keepalive below)

#### 12.1.1. Stream format

The stream always starts with the last known states of the available ports. State changes are sent as **GpioPortStateChange** data structures. The stream is in chronological order (except when device time changes).

#### 12.1.2. Filtering

The stream contains all state changes from all ports by default. The state changes can be filtered by providing a comma separated list of port names with the **filter** GET parameters. Using the URL below the client will only receive state changes of two ports named **IN\_0** and **IN\_1**.

```
http://DEVICE_IP/live/io?filter=IN_0,IN_1
```

#### 12.1.3. Keepalive

During quiet periods the device may not transmit any data for a significant amount of time. Many network equipment may detect such connection as stale and close it prematurely.

Set the **keepalive** GET parameter to a duration in seconds to activate the keepalive messages. The device will automatically send an update message with Content-Type of **application/x-keepalive** when no data transfer was detected for the specified duration.

**Note:** The device may override the keepalive parameter if set too low. The actual keepalive duration is always sent back in the X-Keep-Alive HTTP header. A zero value means keepalive is turned off.

Using the URL below the client will receive a keepalive message after a minute without any data transfer:

```
http://DEVICE_IP/live/io?keepalive=60
```

Below is an example update message:

```
-IPCamIOStreamBoundary
Content-Type: application/x-keepalive
Content-Length: 0

-IPCamIOStreamBoundary
```

#### 12.1.4. Example stream

Example I/O stream request to the device at 192.168.1.101:

```
GET /live/io HTTP/1.1
Host: 192.168.1.101
Connection: keep-alive
Cookie: sid=60ab2b6b
```

Beginning of the response to the above request that contains states for port IN\_0 and OUT\_0:

```
HTTP/1.1 200 OK
Pragma: no-cache
Expires: Thu, 01 Dec 2003 16:00:00 GMT
Connection: close
Content-Type: multipart/x-mixed-replace; boundary=IPCamIOStreamBoundary
Cache: no-cache
Accept-Ranges: none
X-KeepAlive: 0

-IPCamIOStreamBoundary
Content-Type: application/json
Content-Length: 104
X-Timestamp: 1620986982042

{
  "Active" : false,"Port" : "IN_0",
  "Timestamp" : "13265460582042"
  "Type" : "Input",
}

-IPCamIOStreamBoundary
Content-Type: application/json
Content-Length: 106
X-Timestamp: 1620986982042

{
  "Active" : false,"Port" : "OUT_0",
  "Timestamp" : "13265460582042"
  "Type" : "Output",
}
```



```
-IPCamlIOStreamBoundary
```

## 13. REFERENCE

### 13.1 ANALYTICS

The Analytics category is a collection of methods for managing analytics engines, detectors and querying events.

#### Methods

Method	Description
<b>Analytics/GetEvents</b>	Get the buffered events
<b>Analytics/StartEvents</b>	Start the event buffering for the calling session
<b>Analytics/StopEvents</b>	Stop the event buffering for the calling session
<b>Analytics/TriggerEngine</b>	Manually trigger an analytics engine
<b>ANPR</b>	
<b>Analytics/GetAnprEngine</b>	Get the current configuration of the ANPR engine
<b>Analytics/GetAnprEngineDefaults</b>	Get the default configuration of the ANPR engine
<b>Analytics/GetAnprEngineState</b>	Get the current state of the ANPR engine
<b>Analytics/SetAnprEngine</b>	Change the configuration of the ANPR engine
<b>Detectors</b>	
<b>Analytics/CreateDetector</b>	Create a new detector instance
<b>Analytics/DeleteAllDetectors</b>	Delete all detector instances
<b>Analytics/DeleteDetector</b>	Delete the detector instance
<b>Analytics/DisableDetector</b>	Disable the detector
<b>Analytics/EnableDetector</b>	Enable the detector
<b>Analytics/GetDetector</b>	Get the configuration of the detector
<b>Analytics/GetDetectorDefaults</b>	Get the default configuration of a detector type
<b>Analytics/GetDetectorState</b>	Get the state of the detector
<b>Analytics/GetDetectors</b>	Get the active detector instances on this device
<b>Analytics/GetSupportedDetectors</b>	Get the supported detector types on this device
<b>Analytics/SetDetector</b>	Set the configuration of the detector
<b>Tracker</b>	

Page 85/246



Analytics/GetTracker	Get the current configuration of the tracker
Analytics/GetTrackerDefaults	Get the default configuration of the tracker
Analytics/SetTracker	Change the configuration of the tracker



### 13.1.1. Analytics/CreateDetector

Create a new detector instance with the specified type and unique id.

#### Specification

User level	ADMINISTRATOR
Request data	<b>DetectorCreateConfiguration</b>
Response data	<i>none</i>
Exceptions	<p><b>DetectorIdMissingException</b>: The ID of the new detector instance must be specified.</p> <p><b>DetectorIdExistsException</b>: The ID of the new detector instance is already in use.</p> <p><b>DetectorLimitReachedException</b>: Cannot create more detectors of this type. See <b>InstanceLimit</b> in <b>Analytics/GetSupportedDetectors</b>.</p> <p><b>InvalidDetectorTypeException</b>: The specified detector type is unknown. See <b>DetectorClass</b> in <b>Analytics/GetSupportedDetectors</b>.</p>

### 13.1.2. Analytics/DeleteAllDetectors

Deletes all detector instances except built-in detectors

#### Specification

User level	ADMINISTRATOR
Request data	<i>none</i>
Response data	<i>none</i>
Exceptions	<i>none</i>

### 13.1.3. Analytics/DeleteDetector

Deletes a detector instance. Built-in detectors cannot be deleted.

**See also:** [Analytics/DisableDetector](#), [Analytics/EnableDetector](#), [Analytics/GetDetector](#), [Analytics/GetDetectorState](#)

#### Specification

User level	ADMINISTRATOR
Request data	<b>DetectorRequest</b>
Response data	<i>none</i>
Exceptions	<b>DetectorNotFoundException</b> : The specified detector does not exist. <b>AccessDeniedException</b> : The detector specified cannot be removed because it is a built-in detector.

### 13.1.4. Analytics/DisableDetector

Disable the selected detector. A disabled detector will not process signals and analytics. A disabled detector will not emit events except ones that indicate change in configuration and initialization state.

**See also:** [Analytics/DeleteDetector](#), [Analytics/EnableDetector](#), [Analytics/GetDetector](#), [Analytics/GetDetectorState](#)

#### Specification

User level	ADMINISTRATOR
Request data	<b>DetectorRequest</b>
Response data	<i>none</i>
Exceptions	<b>DetectorNotFoundException</b> : The specified detector does not exist.

### 13.1.5. Analytics/EnableDetector

Enable the selected detector so it may resume processing signals and analytics. Enabling an already enabled detector has no effect.

See also: [Analytics/DeleteDetector](#), [Analytics/DisableDetector](#), [Analytics/GetDetector](#), [Analytics/GetDetectorState](#)

#### Specification

User level	ADMINISTRATOR
Request data	<a href="#">DetectorRequest</a>
Response data	<i>none</i>
Exceptions	<b>DetectorNotFoundException</b> : The specified detector does not exist

### 13.1.6. Analytics/GetAnprEngine

Get the current configuration of the ANPR engine

See also: [Analytics/GetAnprEngineDefaults](#), [Analytics/SetAnprEngine](#)

#### Specification

User level	ADMINISTRATOR
Request data	<i>none</i>
Response data	<a href="#">AnprEngineConfiguration</a>
Exceptions	<i>none</i>

### 13.1.7. Analytics/GetAnprEngineDefaults

Get the default configuration of the ANPR engine

See also: [Analytics/GetAnprEngine](#), [Analytics/SetAnprEngine](#)

Specification

User level	ADMINISTRATOR
Request data	<i>none</i>
Response data	<a href="#">AnprEngineConfiguration</a>
Exceptions	<i>none</i>

### 13.1.8. Analytics/GetAnprEngineState

Get the current state of the ANPR engine

Specification

User level	ADMINISTRATOR
Request data	<i>none</i>
Response data	AnprEngineState
Exceptions	<i>none</i>

### 13.1.9. Analytics/GetDetector

Get the current configuration of the selected detector. The content of the response varies depending on the detector type.

See also: [Analytics/DeleteDetector](#), [Analytics/DisableDetector](#), [Analytics/EnableDetector](#), [Analytics/GetDetectorDefaults](#), [Analytics/GetDetectorState](#), [Analytics/SetDetector](#)

#### Specification

User level	ADMINISTRATOR
Request data	<a href="#">DetectorRequest</a>
Response data	<a href="#">Detector</a>
Exceptions	<a href="#">DetectorNotFoundException</a> : The specified detector does not exist

### 13.1.10. Analytics/GetDetectorDefaults

Get the default configuration of the specified detector type. The default parameters will be used when creating a detector without specifying any detector specific configuration.

See also: [Analytics/GetDetector](#), [Analytics/SetDetector](#)

#### Specification

User level	ADMINISTRATOR
Request data	<a href="#">DetectorClassRequest</a>
Response data	<a href="#">Detector</a>
Exceptions	<i>none</i>



### 13.1.11. Analytics/GetDetectorState

Get the current state of the detector.

The detector state indicates if the detector is properly initialized and ready to process data.

See also: [Analytics/DeleteDetector](#), [Analytics/DisableDetector](#), [Analytics/EnableDetector](#), [Analytics/GetDetector](#)

#### Specification

User level	USER
Request data	<b>DetectorRequest</b>
Response data	<b>DetectorState</b>
Exceptions	<b>DetectorNotFoundException</b> : The specified detector does not exist.

### 13.1.12. Analytics/GetDetectors

Get the active detector instances on this device

#### Specification

User level	USER
Request data	<i>none</i>
Response data	<b>DetectorList</b>
Exceptions	<i>none</i>

### 13.1.13. Analytics/GetEvents

Get all events collected since the last call or since the buffering was started. Events may be dropped when the internal buffer allocated for this session is full.

#### Specification

User level	USER
Request data	<i>none</i>
Response data	<b>BufferedEvents</b>
Exceptions	<b>StreamNotStartedException</b> : Event buffering was not started on this session

#### 13.1.14. Analytics/GetSupportedDetectors

Lists all of the supported detector types on this device along other statistics of each type

##### Specification

User level	USER
Request data	<i>none</i>
Response data	<b>SupportedDetectors</b>
Exceptions	<i>none</i>

#### 13.1.15. Analytics/GetTracker

Get the current configuration of the tracker

**See also:** [Analytics/GetTrackerDefaults](#), [Analytics/SetTracker](#)

##### Specification

User level	ADMINISTRATOR
Request data	<i>none</i>
Response data	<b>TrackerConfiguration</b>
Exceptions	<i>none</i>

#### 13.1.16. Analytics/GetTrackerDefaults

Get the default parameters used by the tracker when parameters are missing during a **Analytics/SetTracker** configuration.

**See also:** [Analytics/GetTracker](#), [Analytics/SetTracker](#)

##### Specification

User level	ADMINISTRATOR
Request data	<i>none</i>
Response data	<b>TrackerConfiguration</b>
Exceptions	<i>none</i>

### 13.1.17. Analytics/SetAnprEngine

Change the configuration of the ANPR engine

**See also:** [Analytics/GetAnprEngine](#), [Analytics/GetAnprEngineDefaults](#)

#### Specification

User level	ADMINISTRATOR
Request data	<a href="#">AnprEngineConfiguration</a>
Response data	<i>none</i>
Exceptions	<i>none</i>

### 13.1.18. Analytics/SetDetector

Update the configuration of the selected detector. The required configuration parameters depend on the detector type.

**See also:** [Analytics/GetDetector](#), [Analytics/GetDetectorDefaults](#)

#### Specification

User level	ADMINISTRATOR
Request data	<a href="#">Detector</a>
Response data	<i>none</i>
Exceptions	<b>DetectorNotFoundException</b> : The specified detector does not exist

### 13.1.19. Analytics/SetTracker

Change the configuration of the tracker

**See also:** [Analytics/GetTracker](#), [Analytics/GetTrackerDefaults](#)

#### Specification

User level	ADMINISTRATOR
Request data	<a href="#">TrackerConfiguration</a>
Response data	<i>none</i>
Exceptions	<i>none</i>

### 13.1.20. Analytics/StartEvents

Start the event buffering on this session. If the event buffering was already started this method does nothing. Buffered events can be queried using the **Analytics/GetEvents** method and stopped with **Analytics/ StopEvents**.

The events can be filtered by detectors by specifying their IDs. For more details see the input parameters of this method.

#### Specification

User level	USER
Request data	<b>BufferedEventsRequest</b>
Response data	<i>none</i>
Exceptions	<i>none</i>

### 13.1.21. Analytics/StopEvents

Stop the event buffering for the calling session

#### Specification

User level	USER
Request data	<i>none</i>
Response data	<i>none</i>
Exceptions	<i>none</i>

### 13.1.22. Analytics/TriggerEngine

Manually trigger an analytics engine

#### Specification

User level	USER
Request data	<b>AnalyticsEngineTrigger</b>
Response data	<b>AnalyticsEngineTriggerResponse</b>
Exceptions	<b>InvalidTriggerException</b> : The specified engine does not exist or doesn't support triggers.

## 13.2 STORAGE

The Storage category is a collection of methods for managing the on-board storage and querying stored data.

### Methods

Method	Description
<b>Storage/GetEvents</b>	Perform a query on the stored events
<b>Storage/GetStatistics</b>	Get general statistics from the storage subsystem

### 13.2.1 Storage/GetEvents

Get the list of events from the storage device that match the specified parameters.

#### Specification

User level	USER
Request data	<b>StorageEventsRequest</b>
Response data	<b>StorageEvents</b>
Exceptions	<b>EventsNotFoundException</b> : Events could not be retrieved due to read error

### 13.2.2 Storage/GetStatistics

Get general statistics from the storage subsystem

#### Specification

User level	USER
Request data	<i>none</i>
Response data	<b>StorageStatistics</b>
Exceptions	<i>none</i>

## 13.3 SYSTEM

The **System** category is a collection of methods that allow configuring general aspects of the device like name, time or user accounts. When connecting to a device for the first time it is recommended to use the **System/ GetDevice** method to get general information about it.



## Methods

Method	Description
<b>System/ClearSecurityHistory</b>	Release the block on all clients that are currently banned
<b>System/FactoryReset</b>	Factory reset the settings and reboot
<b>System/GetDevice</b>	Get general information about the device
<b>System/GetSecurityHistory</b>	List the active session and blocked clients
<b>System/GetSecuritySettings</b>	Get the security settings
<b>System/GetVersion</b>	Get the version of the JSON API
<b>System/Reboot</b>	Start the reboot of the device
<b>System/RunTest</b>	Testing method for checking JSON API
<b>System/SetDevice</b>	Change the name and description of the device
<b>System/SetSecuritySettings</b>	Change the security settings
<b>Date &amp; time</b>	
<b>System/GetNtpSettings</b>	Get the NTP settings
<b>System/GetTime</b>	Get the current timestamp
<b>System/SetNtpSettings</b>	Change the NTP settings
<b>System/SetTime</b>	Change the current timestamp
<b>I/O</b>	
<b>System/GetGpioSettings</b>	Get the available digital inputs and outputs on this device
<b>System/GetGpioStates</b>	Get the last known state of available digital inputs and outputs on this device
<b>System/SetGpioInputSettings</b>	Change the configuration of a digital input port
<b>System/SetGpioOutput</b>	Change the state of a digital output port
<b>System/SetGpioOutputSettings</b>	Change the configuration of a digital output port
<b>System/TriggerGpioOutput</b>	Send an impulse to a digital output port
<b>Users</b>	
<b>System/AddUser</b>	Add a new user account
<b>System/DeleteUser</b>	Remove a user account
<b>System/GetCurrentUser</b>	Get the user of the current session
<b>System/GetUsers</b>	List all users accounts on the device. The password field is present but will not contain any information.
<b>System/ModifyUser</b>	Modify the properties of a user account



### 13.3.1 System/AddUser

Add a new user account

**See also:** [System/DeleteUser](#), [System/GetCurrentUser](#), [System/ModifyUser](#)

Specification

User level	ADMINISTRATOR
Request data	<b>User</b>
Response data	<i>none</i>
Exceptions	<b>UserValueException</b> : An invalid parameter was sent <b>UserExistsException</b> : A user with the same name already exists

### 13.3.2 System/ClearSecurityHistory

Release the block on all clients that are currently banned

Specification

User level	ADMINISTRATOR
Request data	<i>none</i>
Response data	<i>none</i>
Exceptions	<i>none</i>

### 13.3.3 System/DeleteUser

Remove a user account

**See also:** [System/AddUser](#), [System/GetCurrentUser](#), [System/ModifyUser](#)

Specification

User level	ADMINISTRATOR
Request data	<b>UserId</b>
Response data	<i>none</i>
Exceptions	<b>DeleteSelfException</b> : A user cannot remove its own account <b>UserNotExistsException</b> : Tried to remove a non-existing user account

### 13.3.4 System/FactoryReset

Request a soft factory reset of the device. The device will restore all except the network settings to factory defaults and request a reboot. For a full factory reset the physical reset button on the device must be pressed if available.

#### Specification

User level	ADMINISTRATOR
Request data	<i>none</i>
Response data	<i>none</i>
Exceptions	<i>none</i>

### 13.3.5 System/GetCurrentUser

Get the user of the current session

**See also:** [System/AddUser](#), [System/DeleteUser](#), [System/ModifyUser](#)

#### Specification

User level	USER
Request data	<i>none</i>
Response data	<b>UserInfo</b>
Exceptions	<i>none</i>

### 13.3.6 System/GetDevice

This method is used for discovering the capabilities of a device after a successful authentication. The response contains the availability of various modules, firmware and product information and lists of supported features.

See also: [System/SetDevice](#)

#### Specification

User level	USER
Request data	none
Response data	SystemSettingsResponse
Exceptions	none

### 13.3.7 System/GetGpioSettings

Get the available digital inputs and outputs on this device

#### Specification

User level	USER
Request data	<i>none</i>
Response data	<b>GpioSettings</b>
Exceptions	<i>none</i>

### 13.3.8 System/GetGpioStates

Get the last known state of available digital inputs and outputs on this device

#### Specification

User level	USER
Request data	<i>none</i>
Response data	<b>GpioStates</b>
Exceptions	<i>none</i>

### 13.3.9 System/GetNtpSettings

Get the NTP settings

**See also:** [System/SetNtpSettings](#)

Specification

User level	USER
Request data	<i>none</i>
Response data	<a href="#">NtpSettings</a>
Exceptions	<i>none</i>

### 13.3.10 System/GetSecurityHistory

List the active session and blocked clients

Specification

User level	USER
Request data	<i>none</i>
Response data	<a href="#">SecurityHistory</a>
Exceptions	<i>none</i>

### 13.3.11 System/GetSecuritySettings

Get the security settings of the device that controls allowed authentication attempts and blocking duration. If the number of authentication fails by a client exceeds the limit the client will be blocked for the specified duration and all authentication attempts - regardless of the used credentials - will be ignored until the block expires.

**See also:** [System/SetSecuritySettings](#)

Specification

User level	USER
Request data	<i>none</i>
Response data	<a href="#">SecuritySettings</a>
Exceptions	<i>none</i>

### 13.3.12 System/GetTime

Get the current timestamp

See also: [System/SetTime](#)

Specification

User level	USER
Request data	<i>none</i>
Response data	<a href="#">TimeSettings</a>
Exceptions	<i>none</i>

### 13.3.13 System/GetUsers

List all users accounts on the device. The password field is present but will not contain any information.

Specification

User level	ADMINISTRATOR
Request data	<i>none</i>
Response data	<a href="#">Users</a>
Exceptions	<i>none</i>

### 13.3.14 System/GetVersion

Get the version of the JSON API. The individual commands' structure and the commands itself may change without the API version changing. Only major structural or workflow changes are reflected here.

Specification

User level	USER
Request data	<i>none</i>
Response data	<a href="#">ApiVersion</a>
Exceptions	<i>none</i>

### 13.3.15 System/ModifyUser

Modify the properties of a user account

**see also:** [System/AddUser](#), [System/DeleteUser](#), [System/GetCurrentUser](#)

#### Specification

User level	ADMINISTRATOR
Request data	<a href="#">User</a>
Response data	<i>none</i>
Exceptions	<b>UserValueException</b> : An invalid parameter was sent <b>Modify-SelfException</b> : A user cannot modify its own role <b>UserNotExistsException</b> : Tried to modify a non-existing user account

### 13.3.16 System/Reboot

Request the device the reboot. The device will reboot shortly after the request.

#### Specification

User level	ADMINISTRATOR
Request data	<a href="#">RebootSettings</a>
Response data	<i>none</i>
Exceptions	<i>none</i>

### 13.3.17 System/RunTest

This method is used for testing the functionality of the JSON API and making implementation easier. This method does not execute actual logic on the device but just returns canned responses.

#### Specification

User level	USER
Request data	<b>TestInput</b>
Response data	<b>TestOutput</b>
Exceptions	<b>TestException</b> : This is an exception thrown when the <b>ThrowException</b> of the input is set to true

### 13.3.18 System/SetDevice

Change the name, description and location of the device usually visible on user interfaces.

**See also:** [System/GetDevice](#)

#### Specification

User level	ADMINISTRATOR
Request data	<i>none</i>
Response data	<b>SystemSettings</b>
Exceptions	<i>none</i>

### 13.3.19 System/SetGpioInputSettings

Change the configuration of a digital input port

**See also:** [System/SetGpioOutput](#), [System/SetGpioOutputSettings](#), [System/TriggerGpioOutput](#)

Specification

User level	ADMINISTRATOR
Request data	<a href="#">GpioInputPort</a>
Response data	<i>none</i>
Exceptions	<i>none</i>

### 13.3.20 System/SetGpioOutput

Change the state of a digital output port

**See also:** [System/SetGpioInputSettings](#), [System/SetGpioOutputSettings](#), [System/TriggerGpio-](#)

Specification

User level	OPERATOR
Request data	<a href="#">GpioOutputPortState</a>
Response data	<i>none</i>
Exceptions	<i>none</i>



### 13.3.21 System/SetGpioOutputSettings

Change the configuration of a digital output port

**See also:** [System/SetGpioInputSettings](#), [System/SetGpioOutput](#), [System/TriggerGpioOutput](#)

Specification

User level	ADMINISTRATOR
Request data	<a href="#">GpioOutputPort</a>
Response data	<i>none</i>
Exceptions	<i>none</i>

### 13.3.22 System/SetNtpSettings

Change the NTP settings

**See also:** [System/GetNtpSettings](#)

Specification

User level	ADMINISTRATOR
Request data	<a href="#">NtpSettings</a>
Response data	<i>none</i>
Exceptions	<i>none</i>

### 13.3.23 System/SetSecuritySettings

Change the security settings

**See also:** [System/GetSecuritySettings](#)

Specification

User level	ADMINISTRATOR
Request data	<a href="#">SecuritySettings</a>
Response data	<i>none</i>
Exceptions	<i>none</i>

### 13.3.24 System/SetTime

Change the current timestamp

**See also:** [System/GetTime](#)

Specification

User level	ADMINISTRATOR
Request data	<a href="#">TimeSettings</a>
Response data	<i>none</i>
Exceptions	<i>none</i>

### 13.3.25 System/TriggerGpioOutput

Send an impulse to a digital output port

**See also:** [System/SetGpioInputSettings](#), [System/SetGpioOutput](#), [System/SetGpioOutputSettings](#)

#### Specification

User level	OPERATOR
Request data	<b>GpioPortId</b>
Response data	<i>none</i>
Exceptions	<i>none</i>

## 13.4 STRUCTS

### 13.4.1 ActiveSession

Active session information

#### Structure

Parameter	Type	Description
LastSeen	int64	Elapsed time in milliseconds since the last activity on this session
Source	string	Source of the session, usually an IP address
User	string	The authenticated user name on the session

#### Pseudo code

```
{  
  "LastSeen": ...,  
  "Source": "...",  
  "User": "..."  
}
```

### 13.4.2 AnalyticsEngineTrigger

Properties of a manual engine trigger.

The Count property defines the number of successful reads before the trigger is considered done. By setting this property to zero you can cancel still active manual triggers.

See also: [Analytics/TriggerEngine](#)

#### Structure

Parameter	Type	Description
Count	int32	Number of triggers to issue
Target	string	Name of engine to trigger (only "Anpr" is supported)
TriggerSource		Advanced settings for Software trigger mode
Name	string	Unique name of the trigger that will be attached to triggered events

#### Pseudo code

```
{  
  "Count": ...,  
  "Target": "...", "TriggerSource":  
  {  
    "Name": "..."  
  }  
}
```

### 13.4.3 AnalyticsEngineTrigger

Properties of a manual engine trigger.

See also: [Analytics/TriggerEngine](#)

#### Structure

Parameter	Type	Description
Name	string	Name of the trigger
Source	string	Name of the triggered engine
Timestamp	int64	Timestamp when the trigger was received by the device

#### Pseudo code

```
{  
  "Name": ...,  
  "Source": "...",  
  "Timestamp":  
}
```

### 13.4.4 AnprEngineConfiguration

Configuration of the ANPR engine.

The engine only operates inside the specified mask and emits an event for each recognized license plate that meet the configured criteria.

By default the engine is automatically triggered by the on-board plate finder and accepts external triggers aswell. This can be changed using the **TriggerModes** option. When using external triggers the engine reads license plates until the specified count is reached. Setting the **InterruptOnRecognition** to true aborts the read after the first successful license plate read. The on-board plate finder - if enabled - is paused while there is an active external trigger.

Available trigger modes are:

- **PlateFinder:** Engine is triggered automatically by the on-board license plate finder
- **Software:** Engine can be triggered using the **Analytics/TriggerEngine** call
- **Hardware:** Engine is triggered by a configured GPIO input port

The **HardwareTriggerSettings/TriggerMode** option controls how the activation of the input port triggers the engine when hardware trigger is used.

- **Impulse:** Activation of the input port triggers the engine to make **ReadCount** number of successful reads
- **State:** The engine continuously tries to read license plates while the input port is active

See also: **Analytics/GetAnprEngine**, **Analytics/GetAnprEngineDefaults**, **Analytics/SetAnprEngine**

## Structure

Parameter	Type	Description
Config		
ColorRecognition	bool	Set to enable color recognition on license plates
Confidence	int8	Minimum accepted confidence value
CountryPreference	string	Preferred country code
Direction	bool	Set to enable direction recognition on license plates
HardwareTriggerSettings		Advanced settings for Hardware trigger mode
InterruptOnRecognition	bool	When enabled stops further recognition after a successful read
Port	string	Name of the GPIO input port that triggers the engine
ReadCount	int32	Number of successful reads before the trigger ends in Impulse mode
TriggerMode	string	Activation mode of the trigger
InterruptOnRecognition	bool	(deprecated) When enabled stops further recognition after a successful read. Ignored when InterruptOnRecognition is specified in HardwareTriggerSettings and SoftwareTriggerSettings.
MMR	bool	Set to enable MMR recognition on license plates
Masks	List/ Array/ int16	List of polygon coordinates that define the operating area of the engine
RecognitionMode	string	Type of traffic the device processes
SoftwareTriggerSettings		Advanced settings for Software trigger mode
InterruptOnRecognition	bool	When enabled stops further recognition after a successful read
TriggerMode	string	(deprecated) Source of triggers that activates the ANPR engine. This setting is overwritten if TriggerModes is specified as well.
TriggerModes	List/ string	Source of triggers that activates the ANPR engine
Type	string	Type of to run
ValidInTimeWindow	int32	Ignore same license plates for this duration (milliseconds)



## Pseudo code

```

{
  "Config":
  {
    "ColorRecognition": ...,
    "Confidence": ...,
    "CountryPreference": "...",
    "Direction": ...,
    "HardwareTriggerSettings":
    {
      "InterruptOnRecognition": ...,
      "Port": "...",
      "ReadCount": ...,
      "TriggerMode": "..."
    },
    "InterruptOnRecognition": ...,
    "MMR": ...,
    "Masks":
    {
      "0": [ ..., ..., ... ],
      "1": [ ..., ..., ... ]
    },
    "RecognitionMode": "...",
    "SoftwareTriggerSettings":
    {
      "InterruptOnRecognition": ...
    },
    "TriggerMode": "...",
    "TriggerModes":
    {
      "0": "...",
      "1": "..."
    },
    "Type": "...",
    "ValidInTimeWindow": ...
  }
}

```

### 13.4.5 AnprEngineState

Current state of the ANPR engine

See also: [Analytics/GetAnprEngineState](#)

#### Structure

Parameter	Type	Description
Config		
Active	bool	Reports if the engine is loaded and functioning properly
Configured	bool	Engine configuration state
Version	string	Currently used engine version information

#### Pseudo code

```
{
  "Config":
  {
    "Active": ...,
    "Configured": ...,
    "Version": "..."
  }
}
```

### 13.4.6 ApiVersion

JSON API information

See also: [System/GetVersion](#)

#### Structure

Parameter	Type	Description
Version	int32	Current version of the JSON API

#### Pseudo code

```
{
  "Version": ...
}
```

### 13.4.7 BufferedEvents

Query collected events in a sessions buffer.

When **Analytics/GetEvents** is called all events from the internal buffer are returned then deleted and subsequent calls will only return events emitted after this call. If too many events are emitted or the duration between two **Analytics/GetEvents** calls are too long the internal buffer may fill up and events may be discarded until the buffer is emptied. The number of discarded events can be monitored using the **DiscardedEvents** property.

See also: **Analytics/GetEvents**

#### Structure

Parameter	Type	Description
DiscardedEvents	int32	Number of events discarded since the start of buffering
EventList	List/ <b>Event</b>	List of events
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the detected event
EventTriggerTime	int64	Wall clock timestamp in milliseconds when the conditions for the event were met
State	string	State of the detector after the event was emitted (see <b>DetectorState</b> )

## Pseudo code

```
{
  "DiscardedEvents": ...,
  "EventList":
  {
    "0":
    {
      "DetectorClassID": ...,
      "DetectorEventType": "...",
      "DetectorID": "{...}", "DetectorVersion": ...,
      "EventCode": ...,
      "EventID": "{...}",
      "EventTime": ...,
      "EventTriggerTime": ...,
      "State": "..."
    },
    "1":
    {
      "DetectorClassID": ...,
      "DetectorEventType": "...",
      "DetectorID": "{...}", "DetectorVersion": ...,
      "EventCode": ...,
      "EventID": "{...}",
      "EventTime": ...,
      "EventTriggerTime": ...,
      "State": "..."
    }
  }
}
```

### 13.4.8 BufferedEventsRequest

Parameters for starting event buffering on the current session.

When the Filter parameter is filled with detector IDs only events from those detectors will be buffered and other events will be discarded. If not specified or left empty all events will be available for query.

See also: [Analytics/StartEvents](#)

#### Structure

Parameter	Type	Description
Filter	List/guid	List of detector IDs

#### Pseudo code

```
{
  "Filter":
  {
    "0": "{...}",
    "1": "{...}"
  }
}
```

### 13.4.9 Detector

Configuration of the detector. The contents of this data collection depends on the selected detector type.

See also: [Analytics/GetDetector](#), [Analytics/GetDetectorDefaults](#), [Analytics/SetDetector](#)

#### Structure

Parameter	Type	Description
Config	<b>DetectorConfiguration</b>	Contains further configuration options specific to the detectortype
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Class	string	Detector type name
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
RestoreDelayMs	int64	<i>unused</i>
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an event is emitted. Not all detectors may use this field.
DetectorID	guid	(optional) Unique ID of the detector instance. This option should only be specified when requesting data from the device.

## Pseudo code

```
{
  "Config":
  {
    "BuiltIn": ...,
    "Class": "...",
    "Description": "...",
    "DetectorClassID": ..., "De-
    tectorID": "{...}",
    "DisplayName": "...",
    "Enabled": ...,
    "FpsLimit": ...,
    "RestoreDelayMs": ...,
    "Version": ..., "Violation-
    TimeMs": ...
  },
  "DetectorID": "{...}",
}
```



### 13.4.10 DetectorClassRequest

Property the uniquely identifies a detector type.

See also: [Analytics/GetDetectorDefaults](#)

#### Structure

Parameter	Type	Description
DetectorClass	string	String id of the detector type.

#### Pseudo code

```
{  
  "DetectorClass": "..."  
}
```



## 13.4.11 DetectorConfiguration

Inherited by: [DetectorConfigurationANPR](#), [DetectorConfigurationIO](#), [DetectorConfiguration-](#)

## Structure

Parameter	Type	Description
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Class	string	Detector type name
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
RestoreDelayMs	int64	<i>unused</i>
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an event is emitted. Not all detectors may use this field.

## Pseudo code

```
{
  "BuiltIn": ...,
  "Class": "...",
  "Description": "...",
  "DetectorClassID": ..., "De-
  tectorID": "{...}",
  "DisplayName": "...",
  "Enabled": ...,
  "FpsLimit": ...,
  "RestoreDelayMs": ...,
  "Version": ..., "Violation-
  TimeMs": ...
}
```

## 13.4.12 DetectorConfigurationANPR → DetectorConfiguration

Configuration of the ANRP detector.

By default the detector signals for all license plates. When whitelist is enabled events will only be emitted for license plates found in the filter.

## Structure

Parameter	Type	Description
Filter	string	New-line separated list of license plates to signal for
Whitelist	bool	Enable filter usage
Inherited from <b>DetectorConfiguration</b> :		
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Class	string	Detector type name
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
RestoreDelayMs	int64	<i>unused</i>
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an event is emitted. Not all detectors may use this field.

## Pseudo code

```
{  
  "BuiltIn": ...,  
  "Class": "...",  
  "Description": "...",  
  "DetectorClassID": ..., "De-  
tectorID": "{...}",  
  "DisplayName": "...",  
  "Enabled": ...,  
  "Filter": "...",  
  "FpsLimit": ...,  
  "RestoreDelayMs": ...,  
  "Version": ...,  
  "ViolationTimeMs": ...,  
  "Whitelist": ...  
}
```



## 13.4.13 DetectorConfigurationEmergencyLane → TrackingDetectorConfiguration

## Structure

Parameter	Type	Description
Masks	List/Array/ int16	Mask defining the working area of the detector (see <a href="#">Geometry-Polygons</a> )
Inherited from <a href="#">TrackingDetectorConfiguration</a> :		
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Class	string	Detector type name
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
RestoreDelayMs	int64	unused
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an event is emitted. Not all detectors may use this field.
Center	bool	Set to true to operate using an object's center point instead of all corners
Confidence	int8	Minimum allowed object confidence when <b>ConfidenceEnabled</b> is set to true
ConfidenceEnabled	bool	Set to true to use a confidence threshold for object monitoring
ObjectTypes	List/string	List of object types that are monitored or empty list for all types

## Pseudo code

```
{
  "BuiltIn": ...,
  "Center": ...,
  "Class": "...",
  "Confidence": ..., "ConfidenceEnabled": ..., "Description": "...",
  "DetectorClassID": ..., "DetectorID": "{...}",
  "DisplayName": "...",
  "Enabled": ...,
  "FpsLimit": ...,
  "Masks":
  {
    "0": [ ..., ..., ... ],
    "1": [ ..., ..., ... ]
  },
  "ObjectTypes":
  {
    "0": "...",
    "1": "..."
  },
  "RestoreDelayMs": ...,
  "Version": ..., "ViolationTimeMs":...
}
```

## 13.4.14 DetectorConfigurationForbiddenZone →TrackingDetectorConfiguration

## Structure

Parameter	Type	Description
Masks	List/Array/ int16	Mask defining the working area of the detector (see <a href="#">Geometry-Polygons</a> )
Inherited from <a href="#">TrackingDetectorConfiguration</a> :		
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Class	string	Detector type name
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
RestoreDelayMs	int64	unused
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an event is emitted. Not all detectors may use this field.
Center	bool	Set to true to operate using an object's center point instead of all corners
Confidence	int8	Minimum allowed object confidence when <b>ConfidenceEnabled</b> is set to true
ConfidenceEnabled	bool	Set to true to use a confidence threshold for object monitoring
ObjectTypes	List/string	List of object types that are monitored or empty list for all types

## Pseudo code

```
{
  "BuiltIn": ...,
  "Center": ...,
  "Class": "...",
  "Confidence": ..., "ConfidenceEnabled": ..., "Description": "...",
  "DetectorClassID": ..., "DetectorID": "{...}",
  "DisplayName": "...",
  "Enabled": ...,
  "FpsLimit": ...,
  "Masks":
  {
    "0": [ ..., ..., ... ],
    "1": [ ..., ..., ... ]
  },
  "ObjectTypes":
  {
    "0": "...",
    "1": "..."
  },
  "RestoreDelayMs": ...,
  "Version": ..., "ViolationTimeMs": ...
}
```



## 13.4.15 DetectorConfigurationIO → DetectorConfiguration

Configuration of the IO detector.

The detector will signal when the configured input port leaves the normal state and ends when the port normalizes.

## Structure

Parameter	Type	Description
InputPort	string	Name of the input port to monitor
Inherited from <b>DetectorConfiguration</b> :		
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Class	string	Detector type name
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
RestoreDelayMs	int64	unused
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an event is emitted. Not all detectors may use this field.



## Pseudo code

```
{  
  "BuiltIn": ...,  
  "Class": "...",  
  "Description": "...",  
  "DetectorClassID": ..., "De-  
tectorID": "{...}",  
  "DisplayName": "...",  
  "Enabled": ...,  
  "FpsLimit": ...,  
  "InputPort": "...",  
  "RestoreDelayMs": ...,  
  "Version": ..., "Violation-  
TimeMs": ...  
}
```



## 13.4.16 DetectorConfigurationLane → TrackingDetectorConfiguration

## Structure

Parameter	Type	Description
Masks	List/Array/ int16	Mask defining the working area of the detector (see <a href="#">GeometryPolygons</a> )
Inherited from <a href="#">TrackingDetectorConfiguration</a> :		
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Class	string	Detector type name
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
RestoreDelayMs	int64	unused
Version	int32	Detector type version
Violation-TimeMs	int64	Violations have to be present for this duration before an event is emitted. Not all detectors may use this field.
Center	bool	Set to true to operate using an object's center point instead of all corners
Confidence	int8	Minimum allowed object confidence when <b>ConfidenceEnabled</b> is set to true
ConfidenceEnabled	bool	Set to true to use a confidence threshold for object monitoring
ObjectTypes	List/string	List of object types that are monitored or empty list for all types

## Pseudo code

```
{
  "BuiltIn": ...,
  "Center": ...,
  "Class": "...",
  "Confidence": ..., "ConfidenceEnabled": ..., "Description": "...",
  "DetectorClassID": ..., "DetectorID": "{...}",
  "DisplayName": "...",
  "Enabled": ...,
  "FpsLimit": ...,
  "Masks":
  {
    "0": [ ..., ..., ... ],
    "1": [ ..., ..., ... ]
  },
  "ObjectTypes":
  {
    "0": "...",
    "1": "..."
  },
  "RestoreDelayMs": ...,
  "Version": ..., "ViolationTimeMs":...
}
```

## 13.4.17 DetectorConfigurationRedStop → TrackingDetectorConfiguration

Detector monitors for objects that cross Lines and leave the area through ExitLines after the light turns red and GracePeriod had elapsed. The TrafficLight type can be configured to be RogColumn (vertical road traffic light), RrwRailRoad (triangular railroad light) or RrwRailRoad2 (horizontal railroad light).

## Structure

Parameter	Type	Description
Direction	string	unused
ExitLines	List/IndexedTrackingDetector-Lines	List of segments defining the exit line of the detector
X0	int32	X coordinate of the start point
X1	int32	X coordinate of the end point
Y0	int32	Y coordinate of the start point
Y1	int32	Y coordinate of the end point
Id	int8	Index of the line
GracePeriod	int64	The grace period in milliseconds after the a light turns red where crossing is still allowed
Lines	List/GeometryLineSegment	List of segments defining the entry line for the detector (see GeometryLine)
X0	int32	X coordinate of the start point
X1	int32	X coordinate of the end point
Y0	int32	Y coordinate of the start point
Y1	int32	Y coordinate of the end point
TrafficLight		
X0	int32	X coordinate of the top left corner
X1	int32	X coordinate of the bottom right corner
Y0	int32	Y coordinate of the top left corner
Y1	int32	Y coordinate of the bottom right
Type	string	Type of the traffic light
Inherited from TrackingDetectorConfiguration:		
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Class	string	Detector type name

Parameter	Type	Description
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
RestoreDelayMs	int64	unused
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an event is emitted. Not all detectors may use this field.
Center	bool	Set to true to operate using an object's center point instead of all corners
Confidence	int8	Minimum allowed object confidence when <b>ConfidenceEnabled</b> is set to true
ConfidenceEnabled	bool	Set to true to use a confidence threshold for object monitoring
ObjectTypes	List/string	List of object types that are monitored or empty list for all types



## Pseudo code

```

{
  "BuiltIn": ...,
  "Center": ...,
  "Class": "...",
  "Confidence": ..., "ConfidenceEnabled": ..., "Description": "...",
  "DetectorClassID": ..., "DetectorID": "{...}",
  "Direction": "...",
  "DisplayName": "...",
  "Enabled": ..., "ExitLines":
  {
    "0":
    {
      "Id": ...,
      "X0": ...,
      "X1": ...,
      "Y0": ...,
      "Y1": ...
    },
    "1":
    {
      "Id": ...,
      "X0": ...,
      "X1": ...,
      "Y0": ...,
      "Y1": ...
    }
  },
  "FpsLimit": ...,
  "GracePeriod": ...,
  "Lines":
  {
    "0":
    {
      "X0": ...,
      "X1": ...,
      "Y0": ...,
      "Y1": ...
    },
    "1":
    {
      "X0": ...,
      "X1": ...,
      "Y0": ...,
      "Y1": ...
    }
  },
  },

```

```
"ObjectTypes":  
{  
  "0": "...",  
  "1": "...",  
},  
"RestoreDelayMs": ...,  
"TrafficLight":  
{  
  "Type": "...",  
  "X0": ...,  
  "X1": ...,  
  "Y0": ...,  
  "Y1": ...,  
},  
"Version": ..., "Violation-  
TimeMs": ...  
}
```



## 13.4.18 DetectorConfigurationStopViolation →TrackingDetectorConfiguration

## Structure

Parameter	Type	Description
Direction	string	Direction of crossing that is monitored
Lines	List/GeometryLineSegment	List of segments (see <a href="#">GeometryLine</a> ). Objects must stop before this line before crossing it.
X0	int32	X coordinate of the start point
X1	int32	X coordinate of the end point
Y0	int32	Y coordinate of the start point
Y1	int32	Y coordinate of the end point
Inherited from <a href="#">TrackingDetectorConfiguration</a> :		
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Class	string	Detector type name
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
RestoreDelayMs	int64	unused
Version	int32	Detector type version
Violation-TimeMs	int64	Violations have to be present for this duration before an event is emitted. Not all detectors may use this field.
Center	bool	Set to true to operate using an object's center point instead of all corners
Confidence	int8	Minimum allowed object confidence when ConfidenceEnabled is set to true
ConfidenceEnabled	bool	Set to true to use a confidence threshold for object monitoring
ObjectTypes	List/string	List of object types that are monitored or empty list for all types



## Pseudo code

```

{
  "BuiltIn": ...,
  "Center": ...,
  "Class": "...",
  "Confidence": ..., "ConfidenceEnabled": ..., "Description": "...",
  "DetectorClassID": ..., "DetectorID": "{...}",
  "Direction": "...",
  "DisplayName": "...",
  "Enabled": ...,
  "FpsLimit": ...,
  "Lines":
  {
    "0":
    {
      "X0": ...,
      "X1": ...,
      "Y0": ...,
      "Y1": ...
    },
    "1":
    {
      "X0": ...,
      "X1": ...,
      "Y0": ...,
      "Y1": ...
    }
  },
  "ObjectTypes":
  {
    "0": "...",
    "1": ..."
  },
  "RestoreDelayMs": ...,
  "Version": ..., "ViolationTimeMs": ...
}

```

## 13.4.19 DetectorConfigurationStoppedObject →TrackingDetectorConfiguration

## Structure

Parameter	Type	Description
Masks	List/Array/ int16	Mask defining the working area of the detector (see <a href="#">GeometryPolygons</a> )
Inherited from <a href="#">TrackingDetectorConfiguration</a> :		
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Class	string	Detector type name
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
RestoreDelayMs	int64	unused
Version	int32	Detector type version
Violation-TimeMs	int64	Violations have to be present for this duration before an event is emitted. Not all detectors may use this field.
Center	bool	Set to true to operate using an object's center point instead of all corners
Confidence	int8	Minimum allowed object confidence when <b>ConfidenceEnabled</b> is set to true
ConfidenceEnabled	bool	Set to true to use a confidence threshold for object monitoring
ObjectTypes	List/string	List of object types that are monitored or empty list for all types

## Pseudo code

```
{
  "BuiltIn": ...,
  "Center": ...,
  "Class": "...",
  "Confidence": ..., "ConfidenceEnabled": ..., "Description": "...",
  "DetectorClassID": ..., "DetectorID": "{...}",
  "DisplayName": "...",
  "Enabled": ...,
  "FpsLimit": ...,
  "Masks":
  {
    "0": [ ..., ..., ... ],
    "1": [ ..., ..., ... ]
  },
  "ObjectTypes":
  {
    "0": "...",
    "1": "..."
  },
  "RestoreDelayMs": ...,
  "Version": ..., "ViolationTimeMs":...
}
```



### 13.4.20 DetectorConfigurationTest → DetectorConfiguration

Configure the test detector.

Based on the configuration the detector will emit signal/restore pairs or plain events periodically.

When Timeout is larger than zero the detector repeats the cycle of emitting a signal after Interval and restoring it after Timeout.

When Timeout is set to zero the detector will simply emit an event every Interval milliseconds.

#### Structure

Parameter	Type	Description
Interval	int64	Duration of normal state in milliseconds
Timeout	int64	Duration of signalling state in milliseconds
Inherited from <b>DetectorConfiguration</b> :		
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Class	string	Detector type name
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
RestoreDelayMs	int64	<i>unused</i>
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an event is emitted. Not all detectors may use this field.

## Pseudo code

```
{  
  "BuiltIn": ...,  
  "Class": "...",  
  "Description": "...",  
  "DetectorClassID": ..., "De-  
tectorID": "{...}",  
  "DisplayName": "...",  
  "Enabled": ...,  
  "FpsLimit": ...,  
  "Interval": ...,  
  "RestoreDelayMs": ...,  
  "Timeout": ...,  
  "Version": ..., "Violation-  
TimeMs": ...  
}
```



## 13.4.21 DetectorConfigurationTrafficLine → TrackingDetectorConfiguration

## Structure

Parameter	Type	Description
Direction	string	Direction of crossing that is monitored
Lines	List/GeometryLineSegment	List of segments defining the line that is monitored for crossing objects (see GeometryLine)
X0	int32	X coordinate of the start point
X1	int32	X coordinate of the end point
Y0	int32	Y coordinate of the start point
Y1	int32	Y coordinate of the end point
Inherited from TrackingDetectorConfiguration:		
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Class	string	Detector type name
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
RestoreDelayMs	int64	unused
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an event is emitted. Not all detectors may use this field.
Center	bool	Set to true to operate using an object's center point instead of all corners
Confidence	int8	Minimum allowed object confidence when <b>ConfidenceEnabled</b> is set to true
ConfidenceEnabled	bool	Set to true to use a confidence threshold for object monitoring
ObjectTypes	List/string	List of object types that are monitored or empty list for all types

## Pseudo code

```

{
  "BuiltIn": ...,
  "Center": ...,
  "Class": "...",
  "Confidence": ..., "ConfidenceEnabled": ..., "Description": "...",
  "DetectorClassID": ..., "DetectorID": "{...}",
  "Direction": "...",
  "DisplayName": "...",
  "Enabled": ...,
  "FpsLimit": ...,
  "Lines":
  {
    "0":
    {
      "X0": ...,
      "X1": ...,
      "Y0": ...,
      "Y1": ...
    },
    "1":
    {
      "X0": ...,
      "X1": ...,
      "Y0": ...,
      "Y1": ...
    }
  },
  "ObjectTypes":
  {
    "0": "...",
    "1": ..."
  },
  "RestoreDelayMs": ...,
  "Version": ..., "ViolationTimeMs": ...
}

```

## 13.4.22 DetectorConfigurationUTurn → TrackingDetectorConfiguration

Detector monitors for objects that perform a complete U-turn while crossing the line in the specified direction.



## Structure

Parameter	Type	Description
Direction	string	Direction of crossing that is monitored
Lines	<b>GeometryPolygons</b>	List of segments defining the line that is parallel and inbetween the two straights of the U path (see <b>GeometryLine</b> )
Masks	List/Array/int16	List of masks. Each mask is a list of coordinates where odd and even indices are x and y coordinates of a corner in the polygon (x0, y0, x1, y1, ...).
Inherited from <b>TrackingDetectorConfiguration</b> :		
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Class	string	Detector type name
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
RestoreDelayMs	int64	unused
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an event is emitted. Not all detectors may use this field.
Center	bool	Set to true to operate using an object's center point instead of all corners
Confidence	int8	Minimum allowed object confidence when <b>ConfidenceEnabled</b> is set to true



Parameter	Type	Description
ConfidenceEnabled	bool	Set to true to use a confidence treshold for object monitoring
ObjectTypes	List/string	List of object types that are monitored or empty list for all types

## Pseudo code

```
{
  "BuiltIn": ...,
  "Center": ...,
  "Class": "...",
  "Confidence": ..., "ConfidenceEnabled": ..., "Description": "...",
  "DetectorClassID": ..., "DetectorID": "{...}",
  "Direction": "...",
  "DisplayName": "...",
  "Enabled": ...,
  "FpsLimit": ...,
  "Lines":
  {
    "Masks":
    {
      "0": [ ..., ..., ... ],
      "1": [ ..., ..., ... ]
    }
  },
  "ObjectTypes":
  {
    "0": "...",
    "1": "..."
  },
  "RestoreDelayMs": ...,
  "Version": ..., "ViolationTimeMs": ...
}
```

### 13.4.23 DetectorConfigurationWhiteLineViolation →TrackingDetectorConfiguration

#### Structure

Parameter	Type	Description
Direction	string	Direction of crossing that is monitored
Lines	List/Geometry-Polygons	List of segments defining the white line on the road surface (see GeometryLine)
X0	int32	X coordinate of the start point
X1	int32	X coordinate of the end point
Y0	int32	Y coordinate of the start point
Y1	int32	Y coordinate of the end point
Inherited from TrackingDetectorConfiguration:		
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Class	string	Detector type name
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
RestoreDelayMs	int64	unused
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an event is emitted. Not all detectors may use this field.
Center	bool	Set to true to operate using an object's center point instead of all corners
Confidence	int8	Minimum allowed object confidence when <b>ConfidenceEnabled</b> is set to true
ConfidenceEnabled	bool	Set to true to use a confidence threshold for object monitoring
ObjectTypes	List/string	List of object types that are monitored or empty list for all types

## Pseudo code

```

{
  "BuiltIn": ...,
  "Center": ...,
  "Class": "...",
  "Confidence": ..., "ConfidenceEnabled": ..., "Description": "...",
  "DetectorClassID": ..., "DetectorID": "{...}",
  "Direction": "...",
  "DisplayName": "...",
  "Enabled": ...,
  "FpsLimit": ...,
  "Lines":
  {
    "0":
    {
      "X0": ...,
      "X1": ...,
      "Y0": ...,
      "Y1": ...
    },
    "1":
    {
      "X0": ...,
      "X1": ...,
      "Y0": ...,
      "Y1": ...
    }
  },
  "ObjectTypes":
  {
    "0": "...",
    "1": ..."
  },
  "RestoreDelayMs": ...,
  "Version": ..., "ViolationTimeMs": ...
}

```

## 13.4.24 DetectorConfigurationWrongTurn → TrackingDetectorConfiguration

Detector monitors for objects that cross the lines in the order of their sequence number.

## Structure

Parameter	Type	Description
LineGroup	List/ <b>GeometryLineGroup</b>	Mask defining the working area of the detector (see <b>GeometryLineGroups</b> )
Lines	List/ <b>GeometryPolygons</b>	List of line segments
X0	int32	X coordinate of the start point
X1	int32	X coordinate of the end point
Y0	int32	Y coordinate of the start point
Y1	int32	Y coordinate of the end point
SequenceNumber	int32	Numeric id of this group for ordering
Inherited from <b>TrackingDetectorConfiguration</b> :		
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Class	string	Detector type name
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
RestoreDelayMs	int64	unused
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an event is emitted. Not all detectors may use this field.
Center	bool	Set to true to operate using an object's center point instead of all corners
Confidence	int8	Minimum allowed object confidence when <b>ConfidenceEnabled</b> is set to true

Parameter	Type	Description
ConfidenceEnabled	bool	Set to true to use a confidence threshold for object monitoring
ObjectTypes	List/string	List of object types that are monitored or empty list for all types



## Pseudo code

```

{
  "BuiltIn": ...,
  "Center": ...,
  "Class": "...",
  "Confidence": ..., "ConfidenceEnabled": ..., "Description": "...",
  "DetectorClassID": ..., "DetectorID": "{...}",
  "DisplayName": "...",
  "Enabled": ...,
  "FpsLimit": ...,
  "LineGroups":
  {
    "0":
    {
      "Lines":
      {
        "0":
        {
          "X0": ...,
          "X1": ...,
          "Y0": ...,
          "Y1": ...
        },
        "1":
        {
          "X0": ...,
          "X1": ...,
          "Y0": ...,
          "Y1": ...
        }
      },
      "SequenceNumber": ...
    },
    "1":
    {
      "Lines":
      {
        "0":
        {
          "X0": ...,
          "X1": ...,
          "Y0": ...,
          "Y1": ...
        },
        "1":
        {
          "X0": ...,
          "X1": ...

```

```
        "Y0": ...,
        "Y1": ...
      },
    },
    "SequenceNumber": ...
  }
},
"ObjectTypes":
{
  "0": "...",
  "1": "..."
},
"RestoreDelayMs": ...,
"Version": ...,
"ViolationTimeMs": ...
}
```



## 13.4.25 DetectorConfigurationWrongWay → TrackingDetectorConfiguration

Detector monitors for objects that move in the specified direction inside the mask. The monitored direction can be extended using AngleRange. For example the value of Angle=90 and AngleRange=10 sets the monitored direction range to 80° - 100°.

## Structure

Parameter	Type	Description
Angle	double	Angle of forbidden direction in degrees. Value of 0° points right and 90° points up.
AngleRange	double	Extends monitored angle in both direction with this degree value
LocationX	int32	X coordinate of the visual aid used for configuration. Does not affect the operation of the detector.
LocationY	int32	Y coordinate of the visual aid used for configuration. Does not affect the operation of the detector.
Masks	List/Array/ int16	Mask defining the working area of the detector (see <a href="#">Geometry-Polygons</a> )
Inherited from <a href="#">TrackingDetectorConfiguration</a> :		
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Class	string	Detector type name
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
RestoreDelayMs	int64	unused
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an event is emitted. Not all detectors may use this field.
Center	bool	Set to true to operate using an object's center point instead of all corners
Confidence	int8	Minimum allowed object confidence when <b>ConfidenceEnabled</b> is set to true
ConfidenceEnabled	bool	Set to true to use a confidence threshold for object monitoring
ObjectTypes	List/string	List of object types that are monitored or empty list for all types



## Pseudo code

```
{
  "Angle": ...,
  "AngleRange": ...,
  "BuiltIn": ...,
  "Center": ...,
  "Class": "...",
  "Confidence": ..., "ConfidenceEnabled": ..., "Description": "...",
  "DetectorClassID": ..., "DetectorID": "{...}",
  "DisplayName": "...",
  "Enabled": ...,
  "FpsLimit": ...,
  "LocationX": ...,
  "LocationY": ...,
  "Masks":
  {
    "0": [ ..., ..., ... ],
    "1": [ ..., ..., ... ]
  },
  "ObjectTypes":
  {
    "0": "...",
    "1": "..."
  },
  "RestoreDelayMs": ...,
  "Version": ..., "ViolationTimeMs": ...
}
```



### 13.4.26 DetectorCreateConfiguration

Initial settings for a new detector instance.

See also: [Analytics/CreateDetector](#)

#### Structure

Parameter	Type	Description
DetectorClass	string	Detector type
DetectorID	guid	Unique ID of the detector instance

#### Pseudo code

```
{  
  "DetectorClass": "...",  
  "DetectorID": "{...}"  
}
```

### 13.4.27 DetectorInfo

Collection of properties defining an instance of a detector type.

A built-in detector is a special instance that is created by the device the first time it is booted and it cannot be deleted by the user.

#### Structure

Parameter	Type	Description
BuiltIn	bool	Indicates if this is a built-in detector or added by a user
Description	string	Description of the detector instance
DetectorClass	string	Detector type
DetectorClassID	int32	Detector type ID
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of the detector instance
State	string	Current state of the detector
Version	int32	Version of this detector

#### Pseudo code

```
{
  "BuiltIn": ...,
  "Description": "...",
  "DetectorClass": "...",
  "DetectorClassID": ..., "De-
  tectorID": "{...}",
  "DisplayName": "...",
  "State": "...",
  "Version": ...
}
```

## 13.4.28 DetectorList

See also: [Analytics/GetDetectors](#)

## Structure

Parameter	Type	Description
Detectors	List/ <b>DetectorInfo</b>	List of the currently available detector instances
BuiltIn	bool	Indicates if this is a built-in detector or added by a user
Description	string	Description of the detector instance
DetectorClass	string	Detector type
DetectorClassID	int32	Detector type ID
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of the detector instance
State	string	Current state of the detector
Version	int32	Version of this detector

## Pseudo code

```
{
  "Detectors":
  {
    "0":
    {
      "BuiltIn": ...,
      "Description": "...",
      "DetectorClass": "...",
      "DetectorClassID": ..., "DetectorID": "{...}",
      "DisplayName": "...",
      "State": "...",
      "Version": ...
    }, "1":
    {
      "BuiltIn": ...,
      "Description": "...",
      "DetectorClass": "...",
      "DetectorClassID": ..., "DetectorID": "{...}",
      "DisplayName": "...",
      "State": "...",
      "Version": ...
    }
  }
}
```

### 13.4.29 DetectorRequest

Collection of properties that uniquely identifies a detector instance.

See also: [Analytics/DeleteDetector](#), [Analytics/DisableDetector](#), [Analytics/EnableDetector](#), [Analytics/GetDetector](#), [Analytics/GetDetectorState](#)

#### Structure

Parameter	Type	Description
DetectorID	guid	Unique ID of the detector instance

#### Pseudo code

```
{  
  "DetectorID": "{...}"  
}
```

### 13.4.30 DetectorState

The detector state value

Numeric value	String value	Description
0	dsNotConfigured	Detector is not configured or the current configuration is invalid
1	dsInit	Detector is currently initializing the state machine and loading configuration
2	dsError	Detector is in an erroneous state and cannot operate
3	dsUnableToOperate	The current device environment does not allow normal operation of detector. This state does not require user interaction and the detector will resume operation once impeding factors are resolved.
4	dsNormal	Detector operation is normal
5	dsSignal	Detector raised one or more signals that are still active. Detector operation is normal.
6	dsDisabled	Detector is disabled and does not process data

See also: [Analytics/GetDetectorState](#)

#### Structure

Parameter	Type	Description
State	int32	Numeric id of the current detector state

#### Pseudo code

```
{
  "State": ...
}
```

### 13.4.31 DetectorTypeInfo

Collection of properties defining a detector type. The device won't allow creation of the more that **InstanceLimit** of one type including the build-in detectors.

#### Structure

Parameter	Type	Description
DetectorClass	string	Detector type
InstanceCount	int32	Currently available detectory of this type
InstanceLimit	int32	Maximum number of this type allowed on the device
Version	int32	Available version of this detector type

#### Pseudo code

```
{  
  "DetectorClass": "...",  
  "InstanceCount": ...,  
  "InstanceLimit": ...,  
  "Version": ...  
}
```

### 13.4.32 Event

Descriptor of an event emitted by a detector.

- **DetectorEventType** uses the following values:
- **detSimpleEvent**: Basic event type where the event has no duration.
- **detSignal**: Signals the start of a longer event. The associated detector will also enter signal state until all signalled events are ended.
- **detRestore**: Ends a previously signalled long event. The **EventID** of the start and end events are the same. The associated detector will return to normal state if **all** signals are ended

Restore event types usually don't contain additional information about the previously started event and only serve to mark the end of a detected occurrence.

**EventCode** is a detector specific numeric code to identify what change caused the event. The following are common event codes used by all detectors:

- **2**: Detector finished initialization
- **3**: Detector failed to initialize and stopped working
- **4**: Detector is unable to operate under the current conditions
- **5**: Detector started initializing
- **6**: Detector was created (by user)
- **7**: Detector was destroyed (by user)
- **100**: Generic event code to mark signal/restore event pairs

Event codes above 100 are detector type specific and may overlap.

Inherited by: **EventANPR**, **EventEmergencyLane**, **EventForbiddenZone**, **EventIO**, **EventLane**, **EventRedStop**, **EventStopViolation**, **EventStoppedObject**, **EventTest**, **EventTrafficLine**, **EventUTurn**, **EventWhiteLineViolation**, **EventWrongTurn**, **EventWrongWay**

#### Structure

Parameter	Type	Description
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the detected event
EventTriggerTime	int64	Wall clock timestamp in milliseconds when the conditions for the event were met
State	string	State of the detector after the event was emitted (see <b>DetectorState</b> )



## Pseudo code

```
{  
  "DetectorClassID": ...,  
  "DetectorEventType": "...",  
  "DetectorID": "{...}", "DetectorVersion": ...,  
  "EventCode": ...,  
  "EventID": "{...}",  
  "EventTime": ...,  
  "EventTriggerTime": ...,  
  "State": "..."  
}
```



## 13.4.33 EventANPR → Event

License plate detection event.

## Structure

Parameter	Type	Description
EventInfo	<b>EventANPRLicensePlate</b>	May contain detector specific additional information
BackgroundColor	string	Background color of the license plate in #RRGGBB format
CharacterSize	int32	Average character size of the license plate
Confidence	double	Confidence of the detection
Coords	Array/int16	Coordinates of the found license plate's boundaries
Country	string	License plate county code
CountryCode	int32	Numeric license plate country code
DedicatedAreaColor	string	Dedicated area color of the license plate in #RRGGBB format
Direction	string	Estimated direction of the vehicle. Possible values are <b>Approaching</b> , <b>Moving away</b> or <b>Unknown</b> .
MMR		Make and model recognition results
Category	string	Vehicle category
CategoryConfidence	double	Confidence of the category recognition
Color	string	Color of vehicle in #RRGGBB format
ColorConfidence	double	Confidence of the color recognition
Make	string	Make of the vehicle
MakeAndModelConfidence	double	Confidence of the make and model recognitions
Model	string	Model of the vehicle
Text	string	License plate text
TextColor	string	Text color of the license plate in #RRGGBB format
TriggerSource		Properties of the trigger that started the license plate recognition
Name	string	Unique name of the trigger

Parameter	Type	Description
Source	string	Type of the trigger (see <b>TriggerModes</b> at <b>AnprEngineConfiguration</b> )
Timestamp	int64	Timestamp of when the trigger was activated
<b>Inherited from <b>Event</b>:</b>		
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the event creation
EventTriggerTime	int64	Wall clock timestamp in milliseconds when the conditions for the event were met
State	string	State of the detector after the event was emitted (see <b>DetectorState</b> )

## Pseudo code

```

{
  "DetectorClassID": ...,
  "DetectorEventType": "...",
  "DetectorID": "{...}", "DetectorVersion": ...,
  "EventCode": ...,
  "EventID": "{...}",
  "EventInfo":
  {
    "BackgroundColor": "...",
    "CharacterSize": ...,
    "Confidence": ...,
    "Coords": [ ..., ..., ... ],
    "Country": "...",
    "CountryCode": ...,
    "DedicatedAreaColor": "...",
    "Direction": "...",
    "MMR":
    {
      "Category": "...", "CategoryConfidence": ..., "Color": "...",
      "ColorConfidence": ...,
      "Make": "...",
      "MakeAndModelConfidence": ...,
      "Model": "..."
    },
    "Text": "...",
    "TextColor": "...", "TriggerSource":
    {
      "Name": "...",
      "Source": "...",
      "Timestamp": ...
    }
  },
  "EventTime": ...,
  "EventTriggerTime": ...,
  "State": "..."
}

```

## 13.4.34 EventANPRLicensePlate

License plate properties

## Structure

Parameter	Type	Description
BackgroundColor	string	Background color of the license plate in #RRGGBB format
CharacterSize	int32	Average character size of the license plate
Confidence	double	Confidence of the detection
Coords	Array/ int16	Coordinates of the found license plate's boundaries
Country	string	License plate county code
CountryCode	int32	Numeric license plate country code
DedicatedAreaColor	string	Dedicated area color of the license plate in #RRGGBB format
Direction	string	Estimated direction of the vehicle. Possible values are <b>Approaching</b> , <b>Moving away</b> or <b>Unknown</b> .
MMR		Make and model recognition results
Category	string	Vehicle category
CategoryConfidence	double	Confidence of the category recognition
Color	string	Color of vehicle in #RRGGBB format
ColorConfidence	double	Confidence of the color recognition
Make	string	Make of the vehicle
MakeAndModelConfidence	double	Confidence of the make and model recognitions
Model	string	Model of the vehicle
Text	string	License plate text
TextColor	string	Text color of the license plate in #RRGGBB format
TriggerSource		Properties of the trigger that started the license plate recognition
Name	string	Unique name of the trigger
Source	string	Type of the trigger (see <b>TriggerModes</b> at <a href="#">AnprEngineConfiguration</a> )
Timestamp	int64	Timestamp of when the trigger was activated

## Pseudo code

```
{
  "BackgroundColor": "...",
  "CharacterSize": ...,
  "Confidence": ...,
  "Coords": [ ..., ..., ... ],
  "Country": "...",
  "CountryCode": ...,
  "DedicatedAreaColor": "...",
  "Direction": "...",
  "MMR":
  {
    "Category": "...", "CategoryConfidence": ..., "Color": "...",
    "ColorConfidence": ...,
    "Make": "...",
    "MakeAndModelConfidence": ...,
    "Model": "..."
  },
  "Text": "...",
  "TextColor": "...", "TriggerSource":
  {
    "Name": "...",
    "Source": "...",
    "Timestamp": ...
  }
}
```

## 13.4.35 EventEmergencyLane → Event

## Structure

Parameter	Type	Description
EventInfo	<b>TrackedObjectInfo</b>	Details of the object that entered the emergency lane
Center		
X	int16	X coordinate of the center of the object
Y	int16	Y coordinate of the center of the object
Confidence	double	Confidence of object tracking and categorization on a scale of 0 to 1
Coords	Array/int16	Coordinate pairs of the object's bounding box (x0,y0,x1,y1,...)
Id	int64	Unique id of the tracked object
StartTime	int64	Wall clock timestamp in milliseconds of the moment the object first appeared
State	string	State of object when the event was created
Type	string	Type of object
<b>Inherited from <b>Event</b>:</b>		
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the event creation
EventTriggerTime	int64	Wall clock timestamp in milliseconds when the conditions for the event were met
State	string	State of the detector after the event was emitted (see <b>DetectorState</b> )

## Pseudo code

```
{
  "DetectorClassID": ...,
  "DetectorEventType": "...",
  "DetectorID": "{...}", "DetectorVersion": ...,
  "EventCode": ...,
  "EventID": "{...}",
  "EventInfo":
  {
    "Center":
    {
      "X": ...,
      "Y": ...
    },
    "Confidence": ...,
    "Coords": [ ..., ..., ... ],
    "Id": ...,
    "StartTime": ...,
    "State": "...",
    "Type": "..."
  },
  "EventTime": ...,
  "EventTriggerTime": ...,
  "State": "..."
}
```



## 13.4.36 EventForbiddenZone → Event

## Structure

Parameter	Type	Description
EventInfo	<b>TrackedObjectInfo</b>	Details of the object that entered the emergency lane
Center		
X	int16	X coordinate of the center of the object
Y	int16	Y coordinate of the center of the object
Confidence	double	Confidence of object tracking and categorization on a scale of 0 to 1
Coords	Array/int16	Coordinate pairs of the object's bounding box (x0,y0,x1,y1,...)
Id	int64	Unique id of the tracked object
StartTime	int64	Wall clock timestamp in milliseconds of the moment the object first appeared
State	string	State of object when the event was created
Type	string	Type of object
<b>Inherited from <b>Event</b>:</b>		
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the event creation
EventTriggerTime	int64	Wall clock timestamp in milliseconds when the conditions for the event were met
State	string	State of the detector after the event was emitted (see <b>DetectorState</b> )

## Pseudo code

```
{
  "DetectorClassID": ...,
  "DetectorEventType": "...",
  "DetectorID": "{...}", "DetectorVersion": ...,
  "EventCode": ...,
  "EventID": "{...}",
  "EventInfo":
  {
    "Center":
    {
      "X": ...,
      "Y": ...
    },
    "Confidence": ...,
    "Coords": [ ..., ..., ... ],
    "Id": ...,
    "StartTime": ...,
    "State": "...",
    "Type": "..."
  },
  "EventTime": ...,
  "EventTriggerTime": ...,
  "State": "..."
}
```



### 13.4.37 EventIO → Event

Input port activation event

#### Structure

Parameter	Type	Description
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the detected event
EventTriggerTime	int64	Wall clock timestamp in milliseconds when the conditions for the event were met
State	string	State of the detector after the event was emitted (see <a href="#">DetectorState</a> )

#### Pseudo code

```
{  
  "DetectorClassID": ...,  
  "DetectorEventType": "...",  
  "DetectorID": "{...}", "DetectorVersion": ...,  
  "EventCode": ...,  
  "EventID": "{...}",  
  "EventTime": ...,  
  "EventTriggerTime": ...,  
  "State": "..."  
}
```

## 13.4.38 EventLane → Event

## Structure

Parameter	Type	Description
EventInfo	<b>TrackedObjectInfo</b>	Details of the object that entered lane
Center		
X	int16	X coordinate of the center of the object
Y	int16	Y coordinate of the center of the object
Confidence	double	Confidence of object tracking and categorization on a scale of 0 to 1
Coords	Array/int16	Coordinate pairs of the object's bounding box (x0,y0,x1,y1,...)
Id	int64	Unique id of the tracked object
StartTime	int64	Wall clock timestamp in milliseconds of the moment the object first appeared
State	string	State of object when the event was created
Type	string	Type of object
<b>Inherited from <b>Event</b>:</b>		
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the event creation
EventTriggerTime	int64	Wall clock timestamp in milliseconds when the conditions for the event were met
State	string	State of the detector after the event was emitted (see <b>DetectorState</b> )

## Pseudo code

```
{
  "DetectorClassID": ...,
  "DetectorEventType": "...",
  "DetectorID": "{...}", "DetectorVersion": ...,
  "EventCode": ...,
  "EventID": "{...}",
  "EventInfo":
  {
    "Center":
    {
      "X": ...,
      "Y": ...
    },
    "Confidence": ...,
    "Coords": [ ..., ..., ... ],
    "Id": ...,
    "StartTime": ...,
    "State": "...",
    "Type": "..."
  },
  "EventTime": ...,
  "EventTriggerTime": ...,
  "State": "..."
}
```

## 13.4.39 EventRedStop → Event

## Structure

Parameter	Type	Description
EventInfo	<b>RedStopViolationInfo</b>	Details of the object that ran the red light.
Center		
X	int16	X coordinate of the center of the object
Y	int16	Y coordinate of the center of the object
Confidence	double	Confidence of object tracking and categorization on a scale of 0 to 1
Coords	Array/int16	Coordinate pairs of the object's bounding box (x0,y0,x1,y1,...)
Id	int64	Unique id of the tracked object
StartTime	int64	Wall clock timestamp in milliseconds of the moment the object first appeared
State	string	State of object when the event was created
Type	string	Type of object
OrangeTimestamp	int64	Wall clock timestamp in milliseconds when the light entered orange state
RedTimestamp	int64	Wall clock timestamp in milliseconds when the light entered red state
<b>Inherited from <b>Event</b>:</b>		
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the event creation
EventTriggerTime	int64	Wall clock timestamp in milliseconds when the conditions for the event were met
State	string	State of the detector after the event was emitted (see <b>DetectorState</b> )

## Pseudo code

```
{
  "DetectorClassID": ...,
  "DetectorEventType": "...",
  "DetectorID": "{...}", "DetectorVersion": ...,
  "EventCode": ...,
  "EventID": "{...}",
  "EventInfo":
  {
    "Center":
    {
      "X": ...,
      "Y": ...
    },
    "Confidence": ...,
    "Coords": [ ..., ..., ... ],
    "Id": ...,
    "OrangeTimestamp": ...,
    "RedTimestamp": ...,
    "StartTime": ...,
    "State": "...",
    "Type": "..."
  },
  "EventTime": ...,
  "EventTriggerTime": ...,
  "State": "..."
}
```

## 13.4.40 EventStopViolation → Event

## Structure

Parameter	Type	Description
EventInfo	<b>TrackedObjectInfo</b>	Details of the object that did not stop for the stop sign
Center		
X	int16	X coordinate of the center of the object
Y	int16	Y coordinate of the center of the object
Confidence	double	Confidence of object tracking and categorization on a scale of 0 to 1
Coords	Array/int16	Coordinate pairs of the object's bounding box (x0,y0,x1,y1,...)
Id	int64	Unique id of the tracked object
StartTime	int64	Wall clock timestamp in milliseconds of the moment the object first appeared
State	string	State of object when the event was created
Type	string	Type of object
<b>Inherited from Event:</b>		
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the event creation
EventTriggerTime	int64	Wall clock timestamp in milliseconds when the conditions for the event were met
State	string	State of the detector after the event was emitted (see <b>DetectorState</b> )



## Pseudo code

```
{
  "DetectorClassID": ...,
  "DetectorEventType": "...",
  "DetectorID": "{...}", "DetectorVersion": ...,
  "EventCode": ...,
  "EventID": "{...}",
  "EventInfo":
  {
    "Center":
    {
      "X": ...,
      "Y": ...
    },
    "Confidence": ...,
    "Coords": [ ..., ..., ... ],
    "Id": ...,
    "StartTime": ...,
    "State": "...",
    "Type": "..."
  },
  "EventTime": ...,
  "EventTriggerTime": ...,
  "State": "..."
}
```

## 13.4.41 EventStoppedObject → Event

## Structure

Parameter	Type	Description
EventInfo	<b>TrackedObjectInfo</b>	Details of the object that stopped in the zone
Center		
X	int16	X coordinate of the center of the object
Y	int16	Y coordinate of the center of the object
Confidence	double	Confidence of object tracking and categorization on a scale of 0 to 1
Coords	Array/int16	Coordinate pairs of the object's bounding box (x0,y0,x1,y1,...)
Id	int64	Unique id of the tracked object
StartTime	int64	Wall clock timestamp in milliseconds of the moment the object first appeared
State	string	State of object when the event was created
Type	string	Type of object
<b>Inherited from Event:</b>		
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the event creation
EventTriggerTime	int64	Wall clock timestamp in milliseconds when the conditions for the event were met
State	string	State of the detector after the event was emitted (see <b>DetectorState</b> )

## Pseudo code

```
{
  "DetectorClassID": ...,
  "DetectorEventType": "...",
  "DetectorID": "{...}", "DetectorVersion": ...,
  "EventCode": ...,
  "EventID": "{...}",
  "EventInfo":
  {
    "Center":
    {
      "X": ...,
      "Y": ...
    },
    "Confidence": ...,
    "Coords": [ ..., ..., ... ],
    "Id": ...,
    "StartTime": ...,
    "State": "...",
    "Type": "..."
  },
  "EventTime": ...,
  "EventTriggerTime": ...,
  "State": "..."
}
```

## 13.4.42 EventTest → Event

Basic test event

## Structure

Parameter	Type	Description
Index	int64	A numeric counter that increments when the detector emitted an event of anytype
Inherited from <b>Event</b> :		
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the detected event
EventTriggerTime	int64	Wall clock timestamp in milliseconds when the conditions for the event were met
State	string	State of the detector after the event was emitted (see <b>DetectorState</b> )

## Pseudo code

```
{
  "DetectorClassID": ...,
  "DetectorEventType": "...",
  "DetectorID": "{...}", "DetectorVersion": ...,
  "EventCode": ...,
  "EventID": "{...}",
  "EventTime": ...,
  "EventTriggerTime": ...,
  "Index": ...,
  "State": "..."
}
```

## 13.4.43 EventTrafficLine → Event

## Structure

Parameter	Type	Description
EventInfo	<b>TrackedObjectInfo</b>	Details of the object that crossed the line
Center		
X	int16	X coordinate of the center of the object
Y	int16	Y coordinate of the center of the object
Confidence	double	Confidence of object tracking and categorization on a scale of 0 to 1
Coords	Array/int16	Coordinate pairs of the object's bounding box (x0,y0,x1,y1,...)
Id	int64	Unique id of the tracked object
StartTime	int64	Wall clock timestamp in milliseconds of the moment the object first appeared
State	string	State of object when the event was created
Type	string	Type of object
Inherited from <b>Event</b> :		
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the event creation
EventTriggerTime	int64	Wall clock timestamp in milliseconds when the conditions for the event were met
State	string	State of the detector after the event was emitted (see <b>DetectorState</b> )

## Pseudo code

```
{
  "DetectorClassID": ...,
  "DetectorEventType": "...",
  "DetectorID": "{...}", "DetectorVersion": ...,
  "EventCode": ...,
  "EventID": "{...}",
  "EventInfo":
  {
    "Center":
    {
      "X": ...,
      "Y": ...
    },
    "Confidence": ...,
    "Coords": [ ..., ..., ... ],
    "Id": ...,
    "StartTime": ...,
    "State": "...",
    "Type": "..."
  },
  "EventTime": ...,
  "EventTriggerTime": ...,
  "State": "..."
}
```

## 13.4.44 EventUTurn → Event

## Structure

Parameter	Type	Description
EventInfo	<b>TrackedObjectInfo</b>	Details of the object that performed an illegal U-turn
Center		
X	int16	X coordinate of the center of the object
Y	int16	Y coordinate of the center of the object
Confidence	double	Confidence of object tracking and categorization on a scale of 0 to 1
Coords	Array/int16	Coordinate pairs of the object's bounding box (x0,y0,x1,y1,...)
Id	int64	Unique id of the tracked object
StartTime	int64	Wall clock timestamp in milliseconds of the moment the object first appeared
State	string	State of object when the event was created
Type	string	Type of object
Inherited from <b>Event</b> :		
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the event creation
EventTriggerTime	int64	Wall clock timestamp in milliseconds when the conditions for the event were met
State	string	State of the detector after the event was emitted (see <b>DetectorState</b> )

## Pseudo code

```
{
  "DetectorClassID": ...,
  "DetectorEventType": "...",
  "DetectorID": "{...}", "DetectorVersion": ...,
  "EventCode": ...,
  "EventID": "{...}",
  "EventInfo":
  {
    "Center":
    {
      "X": ...,
      "Y": ...
    },
    "Confidence": ...,
    "Coords": [ ..., ..., ... ],
    "Id": ...,
    "StartTime": ...,
    "State": "...",
    "Type": "..."
  },
  "EventTime": ...,
  "EventTriggerTime": ...,
  "State": "..."
}
```





## 13.4.45 EventWhiteLineViolation → Event

## Structure

Parameter	Type	Description
EventInfo	<b>TrackedObjectInfo</b>	Details of the object that crossed the white line
Center		
X	int16	X coordinate of the center of the object
Y	int16	Y coordinate of the center of the object
Confidence	double	Confidence of object tracking and categorization on a scale of 0 to 1
Coords	Array/int16	Coordinate pairs of the object's bounding box (x0,y0,x1,y1,...)
Id	int64	Unique id of the tracked object
StartTime	int64	Wall clock timestamp in milliseconds of the moment the object first appeared
State	string	State of object when the event was created
Type	string	Type of object
Inherited from <b>Event</b> :		
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the event creation
EventTriggerTime	int64	Wall clock timestamp in milliseconds when the conditions for the event were met
State	string	State of the detector after the event was emitted (see <b>DetectorState</b> )

## Pseudo code

```
{
  "DetectorClassID": ...,
  "DetectorEventType": "...",
  "DetectorID": "{...}", "DetectorVersion": ...,
  "EventCode": ...,
  "EventID": "{...}",
  "EventInfo":
  {
    "Center":
    {
      "X": ...,
      "Y": ...
    },
    "Confidence": ...,
    "Coords": [ ..., ..., ... ],
    "Id": ...,
    "StartTime": ...,
    "State": "...",
    "Type": "..."
  },
  "EventTime": ...,
  "EventTriggerTime": ...,
  "State": "..."
}
```



## 13.4.46 EventWrongTurn → Event

## Structure

Parameter	Type	Description
EventInfo	<b>TrackedObjectInfo</b>	Details of the object that turned in the wrong direction
Center		
X	int16	X coordinate of the center of the object
Y	int16	Y coordinate of the center of the object
Confidence	double	Confidence of object tracking and categorization on a scale of 0 to 1
Coords	Array/int16	Coordinate pairs of the object's bounding box (x0,y0,x1,y1,...)
Id	int64	Unique id of the tracked object
StartTime	int64	Wall clock timestamp in milliseconds of the moment the object first appeared
State	string	State of object when the event was created
Type	string	Type of object
Inherited from <b>Event</b> :		
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the event creation
EventTriggerTime	int64	Wall clock timestamp in milliseconds when the conditions for the event were met
State	string	State of the detector after the event was emitted (see <b>DetectorState</b> )

## Pseudo code

```
{
  "DetectorClassID": ...,
  "DetectorEventType": "...",
  "DetectorID": "{...}", "DetectorVersion": ...,
  "EventCode": ...,
  "EventID": "{...}",
  "EventInfo":
  {
    "Center":
    {
      "X": ...,
      "Y": ...
    },
    "Confidence": ...,
    "Coords": [ ..., ..., ... ],
    "Id": ...,
    "StartTime": ...,
    "State": "...",
    "Type": "..."
  },
  "EventTime": ...,
  "EventTriggerTime": ...,
  "State": "..."
}
```

## 13.4.47 EventWrongWay → Event

## Structure

Parameter	Type	Description
EventInfo	<b>TrackedObjectInfo</b>	Details of the object that is moving in the wrong direction
Center		
X	int16	X coordinate of the center of the object
Y	int16	Y coordinate of the center of the object
Confidence	double	Confidence of object tracking and categorization on a scale of 0 to 1
Coords	Array/int16	Coordinate pairs of the object's bounding box (x0,y0,x1,y1,...)
Id	int64	Unique id of the tracked object
StartTime	int64	Wall clock timestamp in milliseconds of the moment the object first appeared
State	string	State of object when the event was created
Type	string	Type of object
Inherited from <b>Event</b> :		
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the event creation
EventTriggerTime	int64	Wall clock timestamp in milliseconds when the conditions for the event were met
State	string	State of the detector after the event was emitted (see <b>DetectorState</b> )

## Pseudo code

```
{
  "DetectorClassID": ...,
  "DetectorEventType": "...",
  "DetectorID": "{...}", "DetectorVersion": ...,
  "EventCode": ...,
  "EventID": "{...}",
  "EventInfo":
  {
    "Center":
    {
      "X": ...,
      "Y": ...
    },
    "Confidence": ...,
    "Coords": [ ..., ..., ... ],
    "Id": ...,
    "StartTime": ...,
    "State": "...",
    "Type": "..."
  },
  "EventTime": ...,
  "EventTriggerTime": ...,
  "State": "..."
}
```



### 13.4.48 GPSSettings

#### Structure

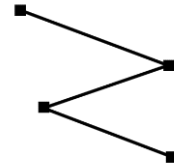
Parameter	Type	Description
Latitude	double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees

#### Pseudo code

```
{  
  "Latitude": ...,  
  "Longitude": ...  
}
```

### 13.4.49 GeometryLine

Segmented line with at least one segment, each consisting of a start and end point



#### Structure

Parameter	Type	Description
Lines	List/ <b>GeometryLineSegment</b>	List of line segments
X0	int32	X coordinate of the start point
X1	int32	X coordinate of the end point
Y0	int32	Y coordinate of the start point
Y1	int32	Y coordinate of the end point
SequenceNumber	int32	Numeric id of this group for ordering

#### Pseudo code

```
{
  "Lines":
  {
    "0":
    {
      "X0": ...,
      "X1": ...,
      "Y0": ...,
      "Y1": ...
    },
    "1":
    {
      "X0": ...,
      "X1": ...,
      "Y0": ...,
      "Y1": ...
    }
  }
  "SequenceNumber": ....
}
```



### 13.4.50 GeometryLineGroup

Segmented line with at least one segment, each consisting of a start and end point and an index for sorting.

#### Structure

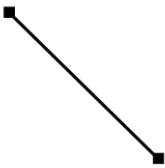
Parameter	Type	Description
Lines	List/ <b>GeometryLineGroup</b>	List of line group
Lines	List/ <b>GeometryLineSegment</b>	List of line segments
X0	int32	X coordinate of the start point
X1	int32	X coordinate of the end point
Y0	int32	Y coordinate of the start point
Y1	int32	Y coordinate of the end point
SequenceNumber	int32	Numeric id of this group for ordering

## Pseudo code

```
{
  "LineGroups":
  {
    "0":
    {
      "Lines":
      {
        "0":
        {
          "X0": ...,
          "X1": ...,
          "Y0": ...,
          "Y1": ...
        },
        "1":
        {
          "X0": ...,
          "X1": ...,
          "Y0": ...,
          "Y1": ...
        }
      },
      "SequenceNumber": ...
    },
    "1":
    {
      "Lines":
      {
        "0":
        {
          "X0": ...,
          "X1": ...,
          "Y0": ...,
          "Y1": ...
        },
        "1":
        {
          "X0": ...,
          "X1": ...,
          "Y0": ...,
          "Y1": ...
        }
      },
      "SequenceNumber": ...
    }
  }
}
```

13.4.51 GeometryLineSegment

Straight line with two points defining the start and end of the line



Structure

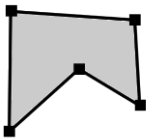
Parameter	Type	Description
X0	int32	X coordinate of the start point
X1	int32	X coordinate of the end point
Y0	int32	Y coordinate of the start point
Y1	int32	Y coordinate of the end point

Pseudo code

```
{
  "X0": ...,
  "X1": ...,
  "Y0": ...,
  "Y1": ...
}
```

### 13.4.52 GeometryPolygons

List of polygons. A polygon has at least 3 points with and an arbitrary shape.



#### Structure

Parameter	Type	Description
Masks	List/Array/ int16	List of masks. Each mask is a list of coordinates where odd and even indices are x and y coordinates of a corner in the polygon (x0, y0, x1, y1, ...).

#### Pseudo code

```
{
  "Masks":
  {
    "0": [ ..., ..., ... ],
    "1": [ ..., ..., ... ]
  }
}
```

### 13.4.53 GeometryRectangle

Rectangle where each side is parallel to the x or y axis of the image



#### Structure

Parameter	Type	Description
X0	int32	X coordinate of the top left corner
X1	int32	X coordinate of the bottom right corner
Y0	int32	Y coordinate of the top left corner
Y1	int32	Y coordinate of the bottom right

#### Pseudo code

```
{  
  "X0": ...,  
  "X1": ...,  
  "Y0": ...,  
  "Y1": ...  
}
```

### 13.4.54 GpioInputPort → GpioPort

Settings of a digital input port

See also: [System/SetGpioInputSettings](#)

#### Structure

Parameter	Type	Description
Inherited from <a href="#">GpioPort</a> :		
Port	string	Unique identifier of a digital input/output port
ActiveState	bool	State of the port that is considered active/triggered (HIGH/CLOSED = true, LOW/ OPEN = false)

#### Pseudo code

```
{  
  "ActiveState": ...,  
  "Port": "..."  
}
```

### 13.4.55 GpioOutputPort → GpioPort

Settings of a digital output port

See also: [System/SetGpioOutputSettings](#)

#### Structure

Parameter	Type	Description
ActiveTime	int32	Duration of the active state after the output is triggered
DetectorList	List/ guid	List of detector IDs that can automatically trigger this output with an event
OutputMode	string	Output signal form. Only the "Impulse" mode is supported.
Inherited from <b>GpioPort</b> :		
Port	string	Unique identifier of a digital input/output port
ActiveState	bool	State of the port that is considered active/triggered (HIGH/CLOSED = true, LOW/ OPEN = false)

#### Pseudo code

```
{  
  "ActiveState": ...,  
  "ActiveTime": ..., "De-  
tectorList":  
  {  
    "0": "{...}",  
    "1": "{...}"  
  },  
  "OutputMode": "...",  
  "Port": "..."  
}
```

### 13.4.56 GpioOutputPortState → GpioPortId

Settings for changing the state of a digital output port

See also: [System/SetGpioOutput](#)

#### Structure

Parameter	Type	Description
Active	bool	New state of the digital output port
Inherited from <a href="#">GpioPortId</a> :		
Port	string	Unique identifier of a digital input/output port

#### Pseudo code

```
{  
  "Active": ...,  
  "Port": "..."  
}
```



### 13.4.57 GpioPort → GpioPortId

Settings of a digital input/output port

Inherited by: [GpioInputPort](#), [GpioOutputPort](#)

See also: [System/SetGpioInputSettings](#), [System/SetGpioOutputSettings](#)

#### Structure

Parameter	Type	Description
ActiveState	bool	State of the port that is considered active/triggered (HIGH/CLOSED = true, LOW/OPEN = false)
Inherited from <a href="#">GpioPortId</a> :		
Port	string	Unique identifier of a digital input/output port

#### Pseudo code

```
{  
  "ActiveState": ...,  
  "Port": "..."  
}
```

## 13.4.58 GpioPortId

Inherited by: [GpioOutputPortState](#), [GpioPort](#), [GpioPortState](#)

See also: [System/SetGpioInputSettings](#), [System/SetGpioOutput](#), [System/SetGpioOutputSettings](#), [System/TriggerGpioOutput](#)

## Structure

Parameter	Type	Description
Port	string	Unique identifier of a digital input/output port

## Pseudo code

```
{  
  "Port": "..."  
}
```

### 13.4.59 GpioPortState → GpioPortId

State of a digital port

Inherited by: [GpioPortStateChange](#)

#### Structure

Parameter	Type	Description
Active	bool	Current state of the digital port
Timestamp	int64	Wall clock timestamp in milliseconds when the digital port changed to this state
Inherited from <a href="#">GpioPortId</a> :		
Port	string	Unique identifier of a digital input/output port

#### Pseudo code

```
{  
  "Active": ...,  
  "Port": "...",  
  "Timestamp": ...  
}
```

## 13.4.60 GpioPortStateChange → GpioPortState

## Structure

Parameter	Type	Description
Type	string	Value of "Input" or "Output" indicating the port type
Inherited from <b>GpioPortState</b> :		
Port	string	Unique identifier of a digital input/output port
Active	bool	Current state of the digital port
Timestamp	int64	Wall clock timestamp in milliseconds when the digital port changed to this state

## Pseudo code

```
{  
  "Active": ...,  
  "Port": "...",  
  "Timestamp": ...,  
  "Type": "..."  
}
```

### 13.4.61 GpioSettings

Settings of all digital input/output ports

See also: [System/GetGpioSettings](#)

#### Structure

Parameter	Type	Description
Inputs	Map/ <a href="#">GpioInputPort</a>	Settings of available digital input ports. Port name is used as mapkey.
Port	string	Unique identifier of a digital input/output port
ActiveState	bool	State of the port that is considered active/triggered (HIGH/CLOSED = true, LOW/OPEN = false)
Outputs	Map/ <a href="#">GpioOutput-Port</a>	Settings of available digital output ports. Port name is used asmap key.
Port	string	Unique identifier of a digital input/output port
ActiveState	bool	State of the port that is considered active/triggered (HIGH/CLOSED = true, LOW/OPEN = false)
ActiveTime	int32	Duration of the active state after the output is triggered
DetectorList	List/guid	List of detector IDs that can automatically trigger this output with an event
Output-Mode	string	Output signal form. Only the "Impulse" mode is supported.

## Pseudo code

```
{
  "Inputs":
  {
    "named_key0":
    {
      "ActiveState": ...,
      "Port": "..."
    },
    "named_key1":
    {
      "ActiveState": ...,
      "Port": "..."
    }
  },
  "Outputs":
  {
    "named_key0":
    {
      "ActiveState": ...,
      "ActiveTime": ..., "De-
      tectorList":
      {
        "0": "{...}",
        "1": "{...}"
      },
      "OutputMode": "...",
      "Port": "..."
    },
    "named_key1":
    {
      "ActiveState": ...,
      "ActiveTime": ..., "De-
      tectorList":
      {
        "0": "{...}",
        "1": "{...}"
      },
      "OutputMode": "...",
      "Port": "..."
    }
  }
}
```

### 13.4.62 GpioStates

Last known state of all digital input/output ports

See also: [System/GetGpioStates](#)

#### Structure

Parameter	Type	Description
Inputs	Map/ <a href="#">GpioPortState</a>	States of available digital input ports. Port name is used as map key.
Port	string	Unique identifier of a digital input/output port
Active	bool	Current state of the digital port
Timestamp	int64	Wall clock timestamp in milliseconds when the digital port changed to this state
Outputs	Map/ <a href="#">GpioPortState</a>	States of available digital output ports. Port name is used as mapkey.
Port	string	Unique identifier of a digital input/output port
Active	bool	Current state of the digital port
Timestamp	int64	Wall clock timestamp in milliseconds when the digital port changed to this state

## Pseudo code

```
{
  "Inputs":
  {
    "named_key0":
    {
      "Active": ...,
      "Port": "...",
      "Timestamp": ...
    },
    "named_key1":
    {
      "Active": ...,
      "Port": "...",
      "Timestamp": ...
    }
  },
  "Outputs":
  {
    "named_key0":
    {
      "Active": ...,
      "Port": "...",
      "Timestamp": ...
    },
    "named_key1":
    {
      "Active": ...,
      "Port": "...",
      "Timestamp": ...
    }
  }
}
```



## 13.4.63 IndexedTrackingDetectorLines → GeometryLineSegment

## Structure

Parameter	Type	Description
Id	int8	Index of the line
Inherited from <b>GeometryLineSegment</b> :		
X0	int32	X coordinate of the start point
X1	int32	X coordinate of the end point
Y0	int32	Y coordinate of the start point
Y1	int32	Y coordinate of the end point

## Pseudo code

```
{  
  "Id": ...,  
  "X0": ...,  
  "X1": ...,  
  "Y0": ...,  
  "Y1": ...  
}
```

## 13.4.64 LocationSettings

## Structure

Parameter	Type	Description
GPS	<b>GPSSettings</b>	Location as GPS coordinates
Latitude	double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees

## Pseudo code

```
{  
  "GPS":  
  {  
    "Latitude": ...,  
    "Longitude": ...  
  }  
}
```

### 13.4.65 ModuleAnalytics → SystemSettingsModule

Capabilities of the Analytics module. The feature list may contain but not limited to the following values:

Tracker	Supports the iTracking tracker engine (see <a href="#">Analytics/GetTracker</a> )
TrafficDetectors	Supports traffic focused detectors
CarmenEngine	Supports CARMEN license plate recognition (see <a href="#">Analytics/GetAn-prEngine</a> )

#### Structure

Parameter	Type	Description
Features	List/string	List of features available in this module
RequiredCarmen-Version	string	Minimum CARMEN version that can be uploaded to the device

#### Pseudo code

```
{
  "Features":
  {
    "0": "...",
    "1": "...",
  },
  "RequiredCarmenVersion": "..."
}
```

## 13.4.66 ModuleIO → SystemSettingsModule

Capabilities of the IO module

### Structure

Parameter	Type	Description
Inputs	List/string	Names of available input ports
Outputs	List/string	Names of available output ports

### Pseudo code

```
{  
  "Inputs":  
  {  
    "0": "...",  
    "1": "..."  
  },  
  "Outputs":  
  {  
    "0": "...",  
    "1": "..."  
  }  
}
```

### 13.4.67 ModuleMedia → SystemSettingsModule

Capabilities of the Media module. The feature map contains a list of features for each available sensor. Each feature list may contain but not limited to the following values:

InfraLed	Infrared LED illumination is available
MotorizedFocus	Focus can be adjusted using the motods on the lens
MotorizedZoom	Zoom can be adjusted using the motors on the lens
WDR	Supports wide dynamic range

#### Structure

Parameter	Type	Description
Features	Map/List/string	List of features available in this module
Sensors	int32	Number of sensors available
Streams	int32	Number of video stream configurations available

#### Pseudo code

```
{
  "Features":
  {
    "named_key0":
    {
      "0": "...",
      "1": "..."
    },
    "named_key1":
    {
      "0": "...",
      "1": "..."
    }
  },
  "Sensors": ...,
  "Streams": ...
}
```

### 13.4.68 NtpSettings

NTP client settings

See also: [System/GetNtpSettings](#), [System/SetNtpSettings](#)

#### Structure

Parameter	Type	Description
Enabled	bool	Enabled state of the device's NTP client
Servers	List/string	List of NTP server addresses or hostnames used when NTP is enabled

#### Pseudo code

```
{
  "Enabled": ...,
  "Servers":
  {
    "0": "...",
    "1": "...",
  }
}
```

### 13.4.69 OptionNumericRange

The numeric range option defines an item's allowed value range from a minimum to a maximum (inclusive). Values outside of the specified range will be ignored as if not sent.

#### Structure

Parameter	Type	Description
Default	numeric	Default value of the item if not set or the value set is out of range
Maximum	numeric	The maximum value the item accepts
Minimum	numeric	The minimum value the item accepts

#### Pseudo code

```
{  
  "Default": ...,  
  "Maximum": ...,  
  "Minimum": ...  
}
```

### 13.4.70 OptionValueList

The value list option defines a limited set of allowed values for an item. A value not present in the list will be ignored as if not sent.

#### Structure

Parameter	Type	Description
Default	string	Default value of the item if not set
Values	List/string	List of values the item can accept

#### Pseudo code

```
{  
  "Default": "...", "Val-  
  ues":  
  {  
    "0": "...",  
    "1": "..."  
  }  
}
```



### 13.4.71 RebootSettings

Reboot parameters

See also: [System/Reboot](#)

#### Structure

Parameter	Type	Description
Message	string	Optional message as the cause of the reboot used for diagnostic purposes

#### Pseudo code

```
{  
  "Message": "..."  
}
```

RedStopViolationInfo → TrackedObjectInfo

## Structure

Parameter	Type	Description
OrangeTimestamp	int64	Wall clock timestamp in milliseconds when the light entered orange state
RedTimestamp	int64	Wall clock timestamp in milliseconds when the light entered red state
Inherited from <b>TrackedObjectInfo</b> :		
Center		
X	int16	X coordinate of the center of the object
Y	int16	Y coordinate of the center of the object
Confidence	double	Confidence of object tracking and categorization on a scale of 0 to 1
Coords	Array/ int16	Coordinate pairs of the object's bounding box (x0,y0,x1,y1,...)
Id	int64	Unique id of the tracked object
StartTime	int64	Wall clock timestamp in milliseconds of the moment the object first appeared
State	string	State of object when the event was created
Type	string	Type of object

## Pseudo code

```

{
  "Center":
  {
    "X": ...,
    "Y": ...
  },
  "Confidence": ..., "Coords": [ ...,
  ..., ... ], "Id": ...,
  "OrangeTimestamp": ...,
  "RedTimestamp": ...,
  "StartTime": ...,
  "State": "...",
  "Type": "..."
}

```

### 13.4.72 SecurityHistory

List of security related information like blocked sources and active sessions

See also: [System/GetSecurityHistory](#)

#### Structure

Parameter	Type	Description
Blocked-Sources	Map/int64	A key/value mapping of blocked sources where the key is the sourceidentifier (usually an IP address) and the value is the duration in milliseconds until the source is unblocked
Sessions	List/ <b>ActiveSession</b>	List of currently active sessions
LastSeen	int64	Elapsed time in milliseconds since the last activity on this session
Source	string	Source of the session, usually an IP address
User	string	The authenticated user name on the session

#### Pseudo code

```
{
  "BlockedSources":
  {
    "named_key0": ...,
    "named_key1": ...
  },
  "Sessions":
  {
    "0":
    {
      "LastSeen": ...,
      "Source": "...",
      "User": "..."
    },
    "1":
    {
      "LastSeen": ...,
      "Source": "...",
      "User": "..."
    }
  }
}
```

### 13.4.73 SecuritySettings

Information required to identify a user account

See also: [System/GetSecuritySettings](#), [System/SetSecuritySettings](#)

#### Structure

Parameter	Type	Description
AuthenticationAttemptLimit	int32	Allowed number of failed authentication attempts before a source is blocked
SourceBlockDuration	int64	Block length in milliseconds

#### Pseudo code

```
{  
  "AuthenticationAttemptLimit": ...,  
  "SourceBlockDuration": ...  
}
```

### 13.4.74 StorageEvents → StorageEventsRequest

Result of a stored event query. The parameters of the original query are returned with **StartTime** and **EndTime** modified to reflect the actual timerange of the result.

The **Status** field will contain one of the following values:

- **OK**: The query returned successfully with at least one event
- **NO\_CONTENT**: The query returned successfully but no events were found that match the criteria
- **PARTIAL\_CONTENT** The query ended successfully but not all events could be returned due to resource constraints

When **PARTIAL\_CONTENT** is returned the device responds with a modified **EndTime** parameter that is the timestamp of the last event that could successfully be returned in this response. To query the rest of the events perform the same query with **StartTime** set to the previously returned **EndTime**.

See also: [Storage/GetEvents](#)

## Structure

Parameter	Type	Description
EventList	List/ <b>Event</b>	List of events that match the search criteria
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the detected event
EventTriggerTime	int64	Wall clock timestamp in milliseconds when the conditions for the event were met
State	string	State of the detector after the event was emitted (see <b>DetectorState</b> )
Status	string	Final status of the query
Inherited from <b>StorageEventsRequest</b> :		
EndTime	int64	Wall clock timestamp in milliseconds of the end of the search range
Filter	<b>StorageEventsRequest-Filter</b>	(optional) Additional filter parameters
FuzzySearch	bool	Set to true to allow fuzzy search that includes not only exact matches but similar matches too where one character may be different
Params	string	(optional) Comma separated list of key:value pairs
Pattern	string	String pattern to match for. May use placeholders to match any characters. A question mark (?) indicates one character, an asterisk (*) indicates zero or more.
ID	guid	(optional) Unique ID of the detector to search for
StartTime	int64	Wall clock timestamp in milliseconds of the beginning of the search range

## Pseudo code

```

{
  "EndTime": ...,
  "EventList":
  {
    "0":
    {
      "DetectorClassID": ...,
      "DetectorEventType": "...",
      "DetectorID": "{...}", "DetectorVersion": ...,
      "EventCode": ...,
      "EventID": "{...}",
      "EventTime": ...,
      "State": "..."
    },
    "1":
    {
      "DetectorClassID": ...,
      "DetectorEventType": "...",
      "DetectorID": "{...}", "DetectorVersion": ...,
      "EventCode": ...,
      "EventID": "{...}",
      "EventTime": ...,
      "EventTriggerTime": ...,
      "State": "..."
    }
  },
  "Filter":
  {
    "FuzzySearch": ...,
    "Params": "...",
    "Pattern": "..."
  },
  "ID": "{...}",
  "StartTime": ...,
  "Status": "..."
}

```

### 13.4.75 StorageEventsRequest

Search parameters for a stored event query

Inherited by: [StorageEvents](#)

See also: [Storage/GetEvents](#)

#### Structure

Parameter	Type	Description
EndTime	int64	Wall clock timestamp in milliseconds of the end of the search range
Filter	<a href="#">StorageEventsRequest-Filter</a>	(optional) Additional filter parameters
FuzzySearch	bool	Set to true to allow fuzzy search that includes not only exact matches but similar matches too where one character may be different
Params	string	(optional) Comma separated list of key:value pairs
Pattern	string	String pattern to match for. May use placeholders to match any characters. A question mark (?) indicates one character, an asterisk (*) indicates zero or more.
ID	guid	(optional) Unique ID of the detector to search for
StartTime	int64	Wall clock timestamp in milliseconds of the beginning of the search range

#### Pseudo code

```
{
  "EndTime": ...,
  "Filter":
  {
    "FuzzySearch": ...,
    "Params": "...",
    "Pattern": "..."
  },
  "ID": "{...}",
  "StartTime": ...
}
```



### 13.4.76 StorageEventsRequestFilter

Additional search parameters for a stored event query.

**Pattern** is used to filter out events whose metadata does not match the pattern.

**Params** can be used to specify modifiers for the search. As of now only "country" is supported (e.g.: "country:NOR" to search for license plates from Norway).

Currently only ANPR events have metadata in the form of license plate strings and country codes.

#### Structure

Parameter	Type	Description
FuzzySearch	bool	Set to true to allow fuzzy search that includes not only exact matches but similiar matches too where one character may be different
Params	string	(optional) Comma separated list of key:value pairs
Pattern	string	String pattern to match for. May use placeholders to match any characters. A question mark (?) indicates one character, an asterisk (*) indicates zero or more.

#### Pseudo code

```
{  
  "FuzzySearch": ...,  
  "Params": "...",  
  "Pattern": "..."  
}
```

### 13.4.77 StorageStatistics

General statistics from the storage subsystem

See also: [Storage/GetStatistics](#)

#### Structure

Parameter	Type	Description
EndTime	int64	Wall clock timestamp in milliseconds of the newest available data on the storage device
InUse	int64	Number of bytes in used on the used storage device
StartTime	int64	Wall clock timestamp in milliseconds of the oldest available data on the storage device
Total	int64	Total number of bytes available on the used storage device

#### Pseudo code

```
{  
  "EndTime": ...,  
  "InUse": ...,  
  "StartTime": ...,  
  "Total": ...  
}
```

## 13.4.78 SupportedDetectors

See also: [Analytics/GetSupportedDetectors](#)

## Structure

Parameter	Type	Description
DetectorTypes	List/ <b>DetectorTypeInfo</b>	List of supported detector types
DetectorClass	string	Detector type
InstanceCount	int32	Currently available detectory of this type
InstanceLimit	int32	Maximum number of this type allowed on the device
Version	int32	Available version of this detector type

## Pseudo code

```
{
  "DetectorTypes":
  {
    "0":
    {
      "DetectorClass": "...",
      "InstanceCount": ...,
      "InstanceLimit": ...,
      "Version": ...
    },
    "1":
    {
      "DetectorClass": "...",
      "InstanceCount": ...,
      "InstanceLimit": ...,
      "Version": ...
    }
  }
}
```

## 13.4.79 SystemSettings

Inherited by: [SystemSettingsResponse](#)

See also: [System/GetDevice](#), [System/SetDevice](#)

## Structure

Parameter	Type	Description
Description	string	User-specified description
Location	<a href="#">LocationSettings</a>	User-specified location
GPS	<a href="#">GPSSettings</a>	Location as GPS coordinates
Latitude	double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees
Name	string	User-specified name

## Pseudo code

```
{
  "Description": "...", "Location":
  {
    "GPS":
    {
      "Latitude": ...,
      "Longitude": ...
    }
  },
  "Name": "..."
}
```

## 13.4.80 SystemSettingsDevice

## Structure

Parameter	Type	Description
Description	string	Additional information about the product
FirmwareVersion	string	Firmware version in x.x.x.x format
ProductClass	string	Class name of the product lineup with similiar features
ProductDisplayName	string	Human-readable name of the product design. May be the same as ProductName.
ProductName	string	Name of the product design
ProductSubclass	string	Subclass of the lineup identifying a specific use-case
RequiredFirmware-Version	string	Minimum firmware version in x.x.x.x format that this device accepts when a new firmware is uploaded
Serial	string	Unique device serial number

## Pseudo code

```
{  
  "Description": "...",  
  "FirmwareVersion": "...",  
  "ProductClass": "...",  
  "ProductDisplayName": "...",  
  "ProductName": "...",  
  "ProductSubclass": "...", "Required-  
  FirmwareVersion": "...", "Serial": "..."  
}
```

13.4.81

SystemSettingsModule

Inherited by: [ModuleAnalytics](#), [ModuleIO](#), [ModuleMedia](#)

Structure

Parameter	Type	Description
-----------	------	-------------

Pseudo code

```
{  
  
}
```

## 13.4.82 SystemSettingsResponse → SystemSettings

See also: [System/GetDevice](#)

## Structure

Parameter	Type	Description
Device	<a href="#">SystemSettingsDevice</a>	General system properties
Description	string	Additional information about the product
FirmwareVersion	string	Firmware version in x.x.x.x format
ProductClass	string	Class name of the product lineup with similar features
ProductDisplayName	string	Human-readable name of the product design. May be the same as Product-Name.
ProductName	string	Name of the product design
ProductSubclass	string	Subclass of the lineup identifying a specific use-case
RequiredFirmware-Version	string	Minimum firmware version in x.x.x.x format that this device accepts when a new firmware is uploaded
Serial	string	Unique device serial number
InstanceId	int64	Unique ID that changes every time the system restarts
Modules	Map/ <a href="#">SystemSettings-Module</a>	List of module specific entries that describe each module's capabilities
Uptime	int64	Elapsed milliseconds since the system started
Inherited from <a href="#">SystemSettings</a> :		
Description	string	User-specified description
Location	<a href="#">LocationSettings</a>	User-specified location
GPS	<a href="#">GPSSettings</a>	Location as GPS coordinates
Latitude	double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees
Name	string	User-specified name

## Pseudo code

```
{
  "Description": "...", "Device":
  {
    "Description": "...",
    "FirmwareVersion": "...",
    "ProductClass": "...",
    "ProductDisplayName": "...",
    "ProductName": "...",
    "ProductSubclass": "...", "Required-
    FirmwareVersion": "...", "Serial": "..."
  },
  "InstanceId": ..., "Location":
  {
    "GPS":
    {
      "Latitude": ...,
      "Longitude": ...
    }
  },
  "Modules":
  {
    "named_key0":
    {
    },
    "named_key1":
    {
    }
  },
  "Name": "...",
  "Uptime": ...
}
```



### 13.4.83 TestInput

Configure the response given to the **System/RunTest** method. The **Text** may be set to anything or left empty. Using the **ThrowException** field, one can control the type of response the **RunTest** command may return.

- If this is false the response will be success (given no other higher level errors occur) and a **TestOutput** object will be returned.
- If this is true the response will be an error of a **TextException** type.

See also: **System/RunTest**

#### Structure

Parameter	Type	Description
Text	string	Arbitrary test input that the <b>System/RunTest</b> will return if no exceptions are thrown
ThrowException	bool	If this field is set to true the response to <b>System/RunTest</b> will be an exception

#### Pseudo code

```
{  
  "Text": "...",  
  "ThrowException": ...  
}
```

### 13.4.84 TestOutput

Response to a successful System/RunTest method call.

See also: [System/RunTest](#)

#### Structure

Parameter	Type	Description
Size	int32	Length of the original input text in bytes
Text	string	The original input text preceeded with the "Input recieved: " string
User	string	Name of the user executing the command

#### Pseudo code

```
{  
  "Size": ...,  
  "Text": "...",  
  "User": "..."  
}
```

### 13.4.85 TimeSettings

Device time settings

See also: [System/GetTime](#), [System/SetTime](#)

#### Structure

Parameter	Type	Description
Timestamp	int64	Current wall clock timestamp on the device (UTC)

#### Pseudo code

```
{  
  "Timestamp": ...  
}
```

## 13.4.86 TrackedObjectInfo

Inherited by: [RedStopViolationInfo](#)

## Structure

Parameter	Type	Description
Center		
X	int16	X coordinate of the center of the object
Y	int16	Y coordinate of the center of the object
Confidence	double	Confidence of object tracking and categorization on a scale of 0 to 1
Coords	Array/ int16	Coordinate pairs of the object's bounding box (x0,y0,x1,y1,...)
Id	int64	Unique id of the tracked object
StartTime	int64	Wall clock timestamp in milliseconds of the moment the object first appeared
State	string	State of object when the event was created
Type	string	Type of object

## Pseudo code

```
{
  "Center":
  {
    "X": ...,
    "Y": ...
  },
  "Confidence": ...,
  "Coords": [ ..., ..., ... ],
  "Id": ...,
  "StartTime": ...,
  "State": "...",
  "Type": "..."
}
```

### 13.4.87 TrackerConfiguration

Configuration of the iTracking engine.

The engine operates inside the configured mask or the whole image if none specified. Moving objects are tracked and categorized and sent to track based detectors for further analysis.

See also: [Analytics/GetTracker](#), [Analytics/GetTrackerDefaults](#), [Analytics/SetTracker](#)

#### Structure

Parameter	Type	Description
Config		
Masks	List/Array/int16	Mask defining the working area of the tracker (see <a href="#">GeometryPolygons</a> )

#### Pseudo code

```
{
  "Config":
  {
    "Masks":
    {
      "0": [ ..., ..., ... ],
      "1": [ ..., ..., ... ]
    }
  },
}
```

## 13.4.88 TrackingDetectorConfiguration → DetectorConfiguration

Inherited by: [DetectorConfigurationEmergencyLane](#), [DetectorConfigurationForbiddenZone](#), [DetectorConfigurationLane](#), [DetectorConfigurationRedStop](#), [DetectorConfigurationStopViolation](#), [DetectorConfigurationStoppedObject](#), [DetectorConfigurationTrafficLine](#), [DetectorConfigurationUTurn](#), [DetectorConfigurationWhiteLineViolation](#), [DetectorConfigurationWrongTurn](#), [DetectorConfigurationWrongWay](#)

## Structure

Parameter	Type	Description
Center	bool	Set to true to operate using an object's center point instead of all corners
Confidence	int8	Minimum allowed object confidence when <b>ConfidenceEnabled</b> is set to true
ConfidenceEnabled	bool	Set to true to use a confidence threshold for object monitoring
ObjectTypes	List/string	List of object types that are monitored or empty list for all types
Inherited from <a href="#">DetectorConfiguration</a> :		
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Class	string	Detector type name
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
RestoreDelayMs	int64	unused
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an event is emitted. Not all detectors may use this field.

## Pseudo code

```
{
  "BuiltIn": ...,
  "Center": ...,
  "Class": "...",
  "Confidence": ..., "ConfidenceEnabled": ..., "Description": "...",
  "DetectorClassID": ..., "DetectorID": "{...}",
  "DisplayName": "...",
  "Enabled": ...,
  "FpsLimit": ..., "ObjectTypes":
  {
    "0": "...",
    "1": "...",
  },
  "RestoreDelayMs": ...,
  "Version": ..., "ViolationTimeMs": ...
}
```

## 13.4.89 User → UserInfo

All user account information

See also: [System/AddUser](#), [System/ModifyUser](#)

## Structure

Parameter	Type	Description
Password	string	User password (write only)
Inherited from <a href="#">UserInfo</a> :		
Name	string	User name
Role	string	User role

## Pseudo code

```
{  
  "Name": "...",  
  "Password": "...",  
  "Role": "..."  
}
```



### 13.4.90 UserId

Information required to identify a user account

Inherited by: [UserInfo](#)

See also: [System/AddUser](#), [System/DeleteUser](#), [System/GetCurrentUser](#), [System/ModifyUser](#)

#### Structure

Parameter	Type	Description
Name	string	User name

#### Pseudo code

```
{  
  "Name": "..."  
}
```

## 13.4.91 UserInfo → UserId

User account information

Inherited by: [User](#)

See also: [System/AddUser](#), [System/GetCurrentUser](#), [System/ModifyUser](#)

## Structure

Parameter	Type	Description
Role	string	User role
Inherited from <a href="#">UserId</a> :		
Name	string	User name

## Pseudo code

```
{  
  "Name": "...",  
  "Role": "..."  
}
```

### 13.4.92 Users

Contains information about all user accounts available on the device

See also: [System/GetUsers](#)

#### Structure

Parameter	Type	Description
Users	List/ <a href="#">User</a>	List of user accounts
Name	string	User name
Role	string	User role
Password	string	User password (write only)

#### Pseudo code

```
{
  "Users":
  {
    "0":
    {
      "Name": "...",
      "Password": "...",
      "Role": "..."
    },
    "1":
    {
      "Name": "...",
      "Password": "...",
      "Role": "..."
    }
  }
}
```

## CONTACT INFORMATION

### Headquarters:

Adaptive Recognition, Hungary Inc.  
Alkotás utca 41 HU  
1123 Budapest Hungary  
Web: [adaptiverecognition.com](http://adaptiverecognition.com)

### Service Address:

Adaptive Recognition, Hungary Inc.  
Ipari Park HRSZ1113/1 HU  
2074 Perbál Hungary  
Web: [adaptiverecognition.com/support/](http://adaptiverecognition.com/support/)

Adaptive Recognition Hungary Technical Support System (ATSS) is designed to provide you the fastest and most proficient assistance, so you can quickly get back to business.

Information regarding your hardware, latest software updates and manuals are easily accessible for customers via our [Documents Site \(www.adaptiverecognition.com/doc\)](http://www.adaptiverecognition.com/doc) after a quick registration.

### New User

If this is your first online support request, please contact your sales representative to register you in our Support System. More help [here \(www.adaptiverecognition.com/support\)](http://www.adaptiverecognition.com/support)!

### Returning User

All registered ATSS customers receive a personal access link via e-mail. If you previously received a confirmation message from ATSS, it contains the embedded link that allows you to securely enter the support site.

