ADAPTIVE **RECOGNITION**

LYNET

User Manual

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

© Copyright Adaptive Recognition Inc.
All rights reserved.

RELEASE DATE
18.11.2024.

LYNET

USER MANUAL

Document version: 2024.11.18.

Table of Contents

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

1. OVERVIEW

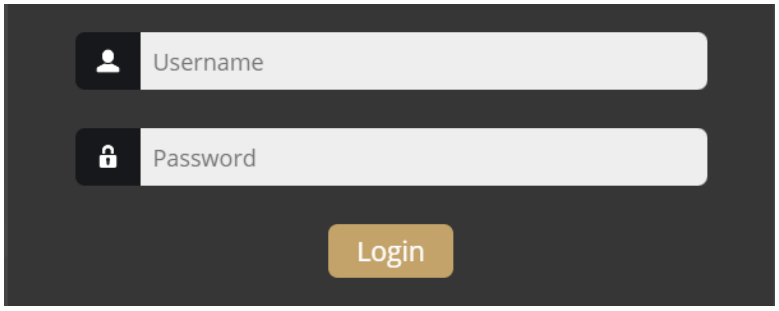
The Lynet camera has its own web interface through which you can access the camera settings, the **LIVE**, the **PLAYBACK**, the **EVENTS** and the **SETTINGS** interfaces.

> 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 camera 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]**.

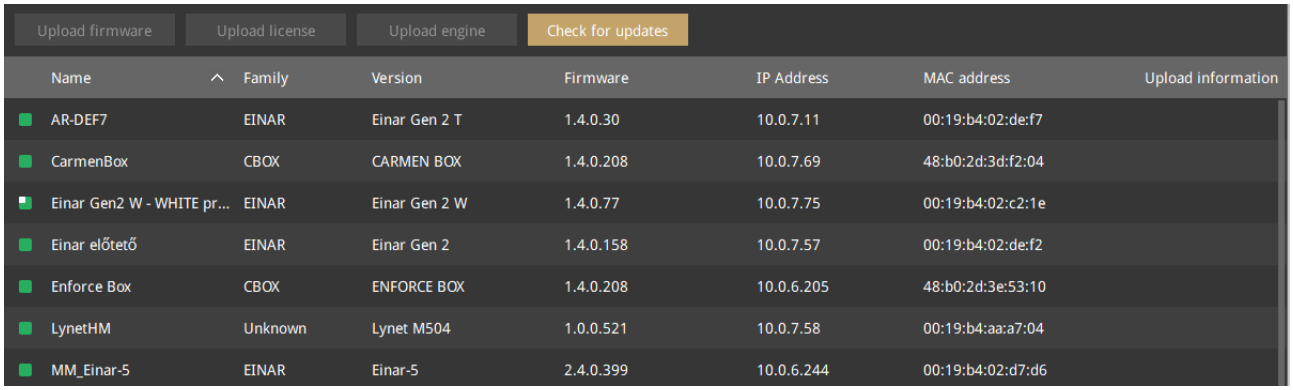
   The default user account is the following:
   Username: admin
   Password: admin



If you cannot connect to the camera's web interface, please refer to section 4.8 of the Installation Guide.

2. AR DEVICE TOOL

With the Harbard Device Tool, you can discover Lynet, Einar or Visus cameras, Carmen Box; 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: Harbard DeviceTool.



2.1. FINDING CAMERAS

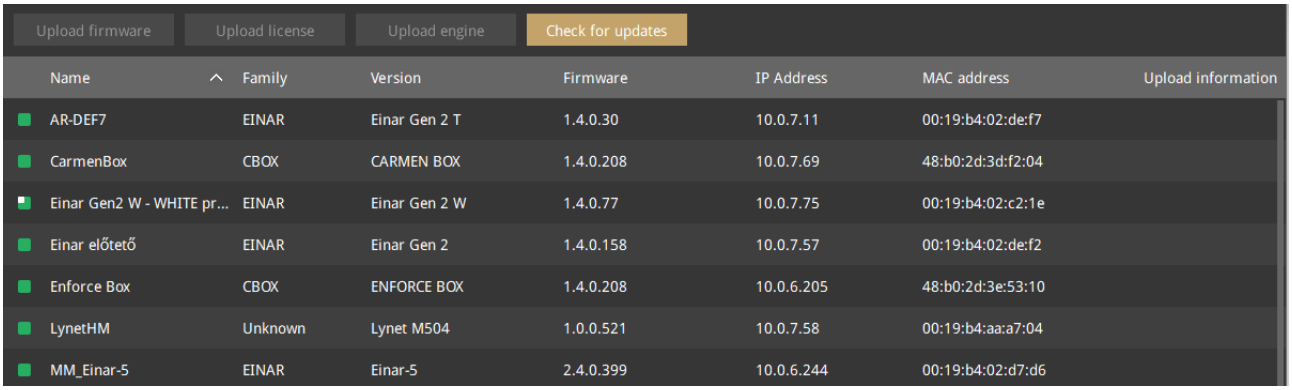
Once started, the program lists the AR devices/cameras detected on the local network if the devices/cameras are in the same network segment as the computer. The device/camera name, product family name, type, firmware version, IP address, MAC address and brief information about the current upload process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

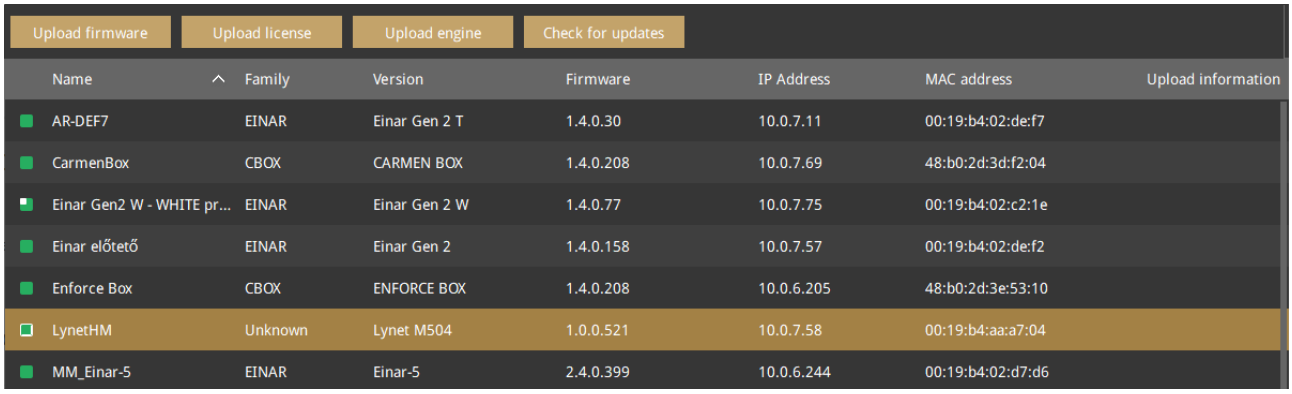
If newer firmware or engine are available for any of the listed cameras/devices, a star sign is added into the green indicator.

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

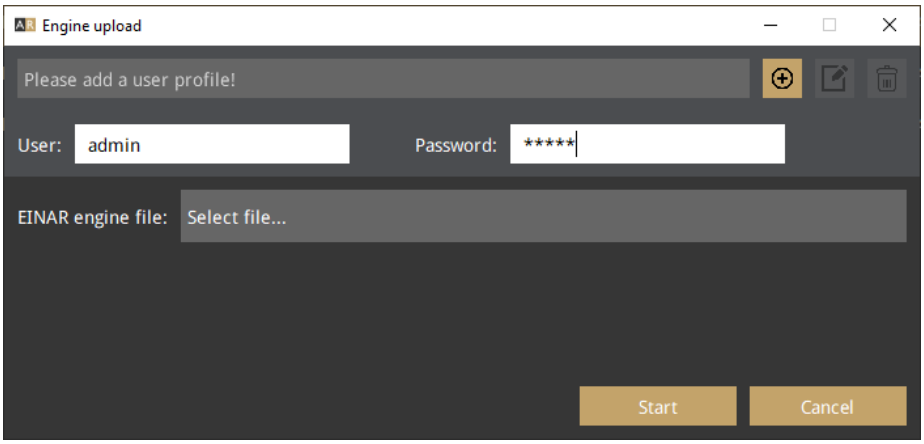


2.2. FIRMWARE, LICENSE, AND ENGINE UPLOAD - MANUALLY

In addition to find devices/cameras, you can also use the Harbard Device Tool 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.



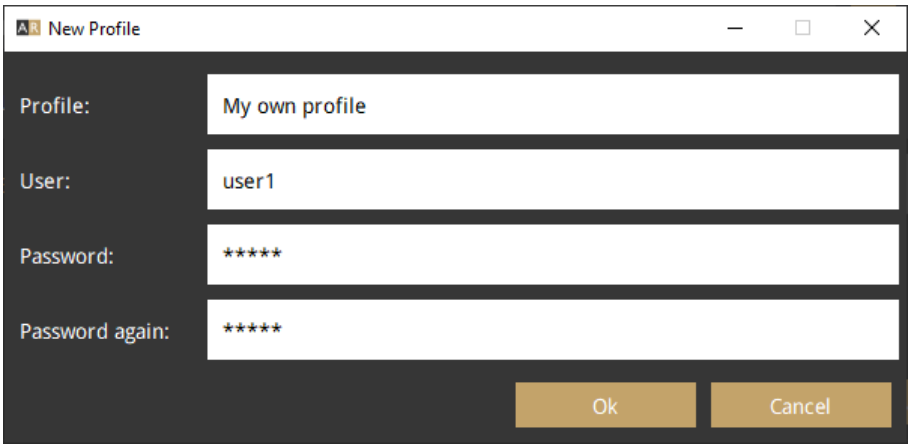
Select the device(s)/camera(s) you want to update and press the **[Upload firmware]**, **[Upload License]** or **[Upload engine]** buttons that become active.
In the window that appears, enter the username and password to access the camera, select the file you want to upload and click **[Start]** to 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.

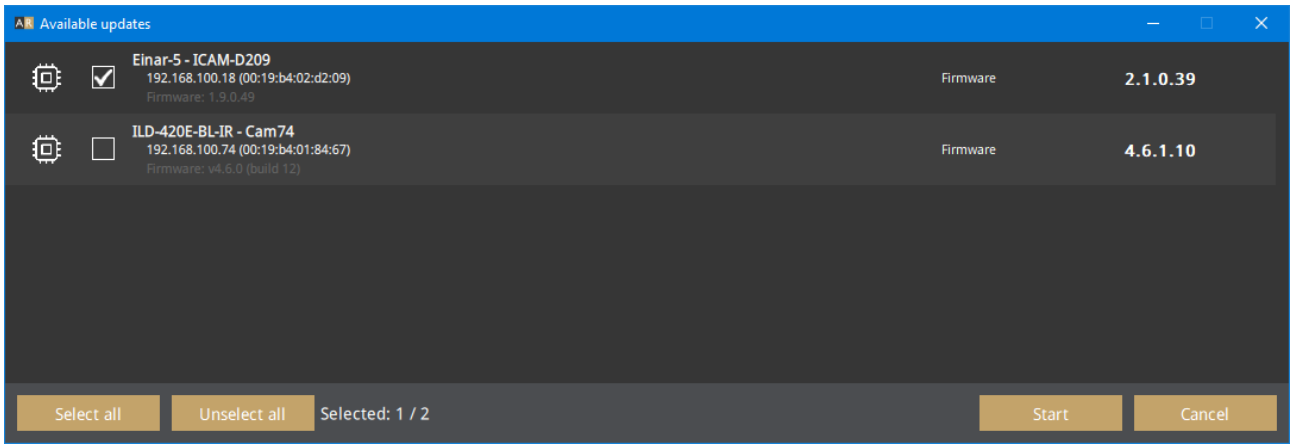
Click on **[Please add a user profile!]** or the **[+]** button and enter the required information. For further uploads, you will only need to select the user profile.

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



2.3. FIRMWARE AND ENGINE – CHECKING FOR UPDATES

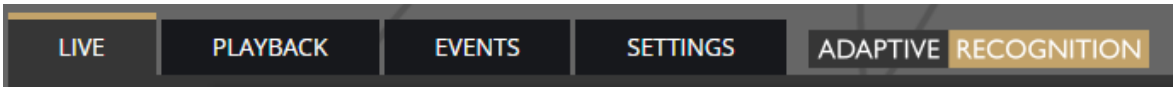
If your PC is connected to the internet, you can check if newer firmware or engine are available for any of your cameras/devices. Press **[Check for updates]** button, select device(s)/camera(s) you want to update and press **[Start]**. Confirmation and credentials must be provided.



Harbard Device Tool 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 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

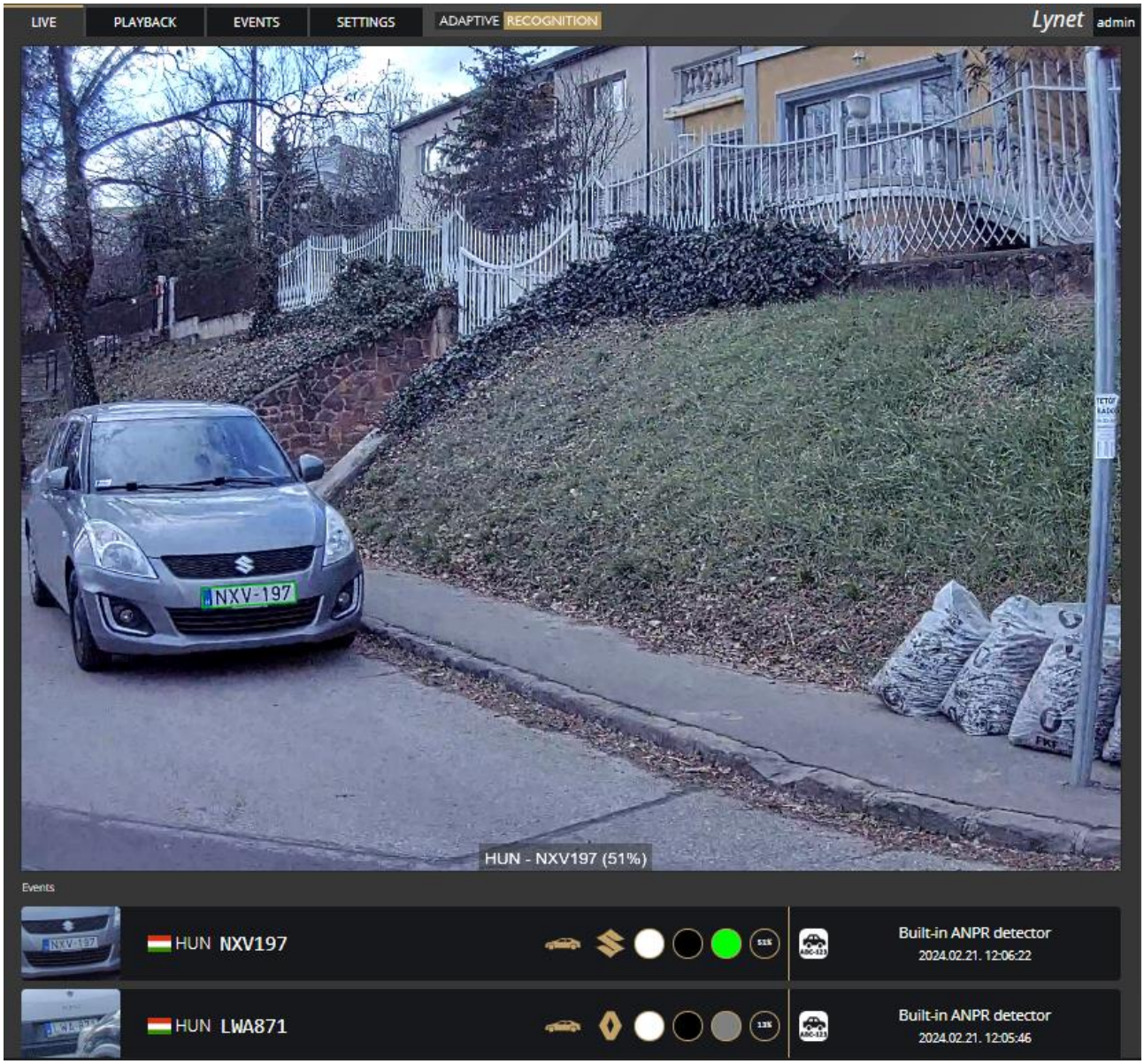
- Image
- Video
- Lens

ANALYTICS

- General
- Uploaders
- Detectors

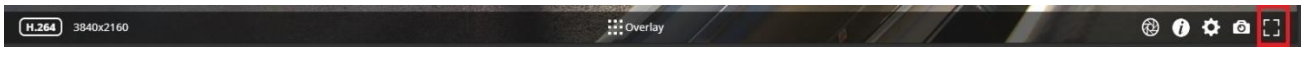
4. LIVE

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



4.1. FULL-SCREEN MODE

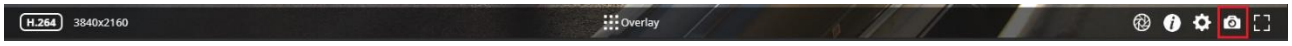
The camera's live stream can be displayed on full screen by clicking on the icon located in the bottom-right corner of the image.



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

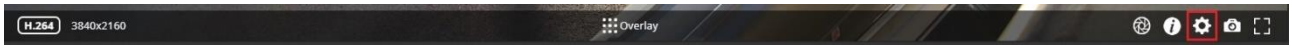
4.2. SAVING IMAGE

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

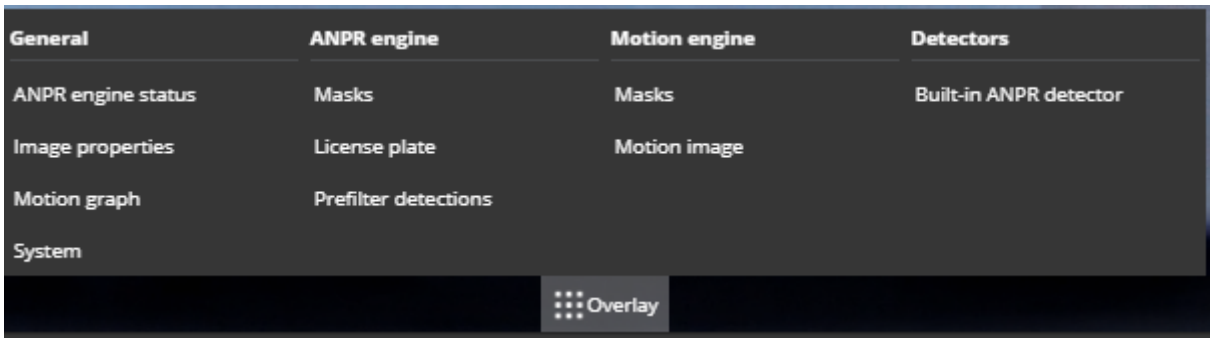
Next to the **[Streams]** icon is the **[Help]** button. It brings up keyboard shortcuts on how to use and navigate the video feed. To exit from the Help OSD, press the **[Help]** button or click in the grey area.

4.5. ZOOM & FOCUS

The last icon on the right is the **[Zoom & Focus]** button. It brings up Zoom and Focus controls to fine-tune the lens of the camera. If you press the **[Autofocus]** button, the camera starts to look for the correct focus value by itself to achieve the sharpest image possible. With the **[Focus to point]** button you can select a specific point of the image what you want to be the sharpest region in the picture. To exit from the Zoom & Focus OSD, press the **[Zoom & Focus]** button.

4.6. 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, eg. ***{Media / Image}***. The OSD layers come in handy for observing the internal workflow of the camera, setting up the camera or troubleshooting.



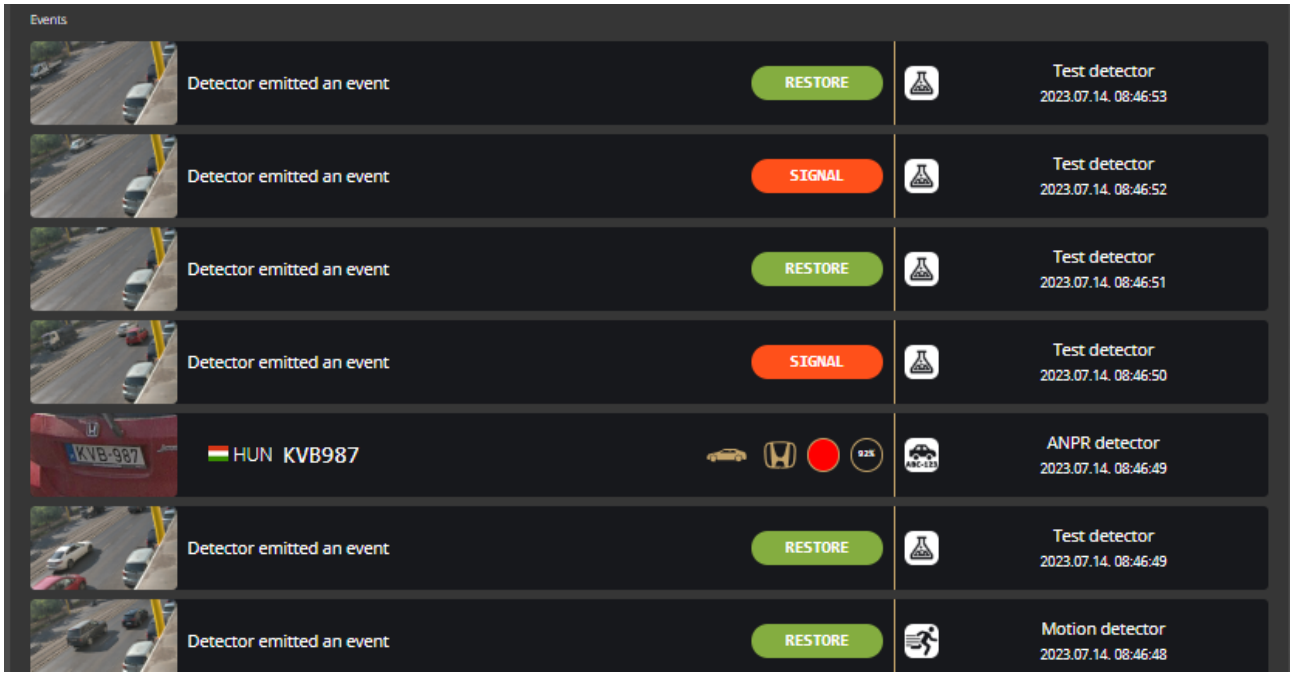
The most important parameters related to number plate recognition can be found on the **ANPR engine status** OSD, these are the following:

Found/Read: Number of images on which the camera has started to detect („Read") license plate, and („Found") it. Values will be reseted after reboot.
Prefilter: „found" means the device detected license plate in that image (and in the ANPR mask).
„not-found" means that the device tried it, but did not detect any license plate in that image.
„idle" means that the device has not tried to detect plates. The prefilter does not run on every images.
Carmen-Engine: „found" is displayed when the Carmen ANPR recognition module has processed an image. „no-input" means the engine do not receive image from the prefilter.
Plate: Recognised license plate result of Carmen engine.
Avg. char height: Character height of recognised license plate.
Confidence: Confidence value of recognised license plate.
Confidence threshold: Minimum confidence value previously set in ***{ANPR engine}***. License plates with confidence value below this threshold will be discarded.

Recognition memory: Recognised license plates which will not be recognised again. The time range while the recognised license plates remain in the memory can be set in ***{ANPR engine}***/"Ignore same plate for (s)".

4.7. EVENT PREVIEW

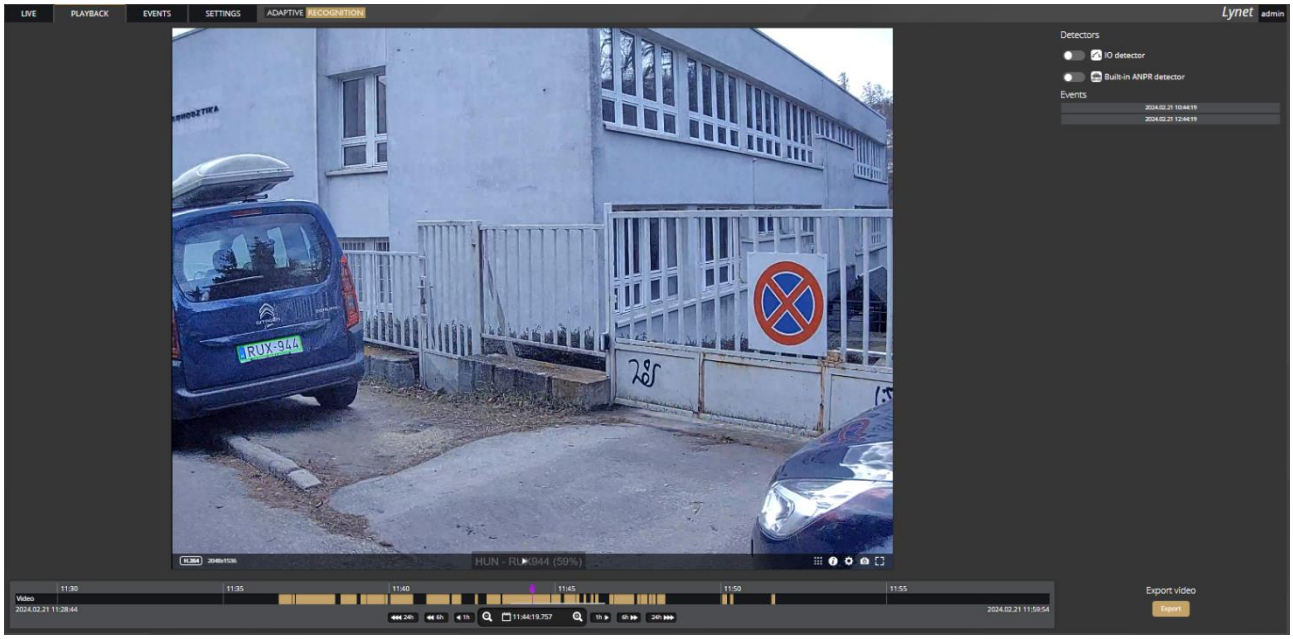
You can find the event preview section under the live stream image, displaying the notifications about the latest received events.
A orange colored "**SIGNAL**" text indicates a start of a longer event that lasts for multiple frames. A long end of event is marked with the green "**RESTORE**" text.



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

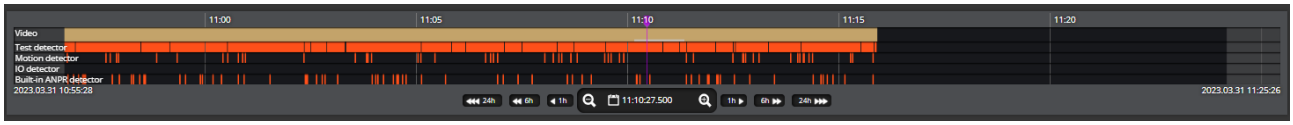
In case the storage is turned off but the storage device is available, the previously recorded elements can be viewed and played if the storage function is switched on.

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.

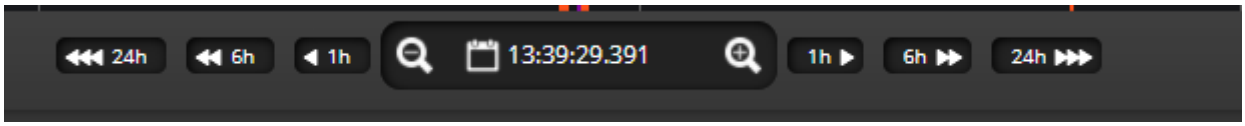
The **red markers** point where events have taken place.



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 play back.

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.

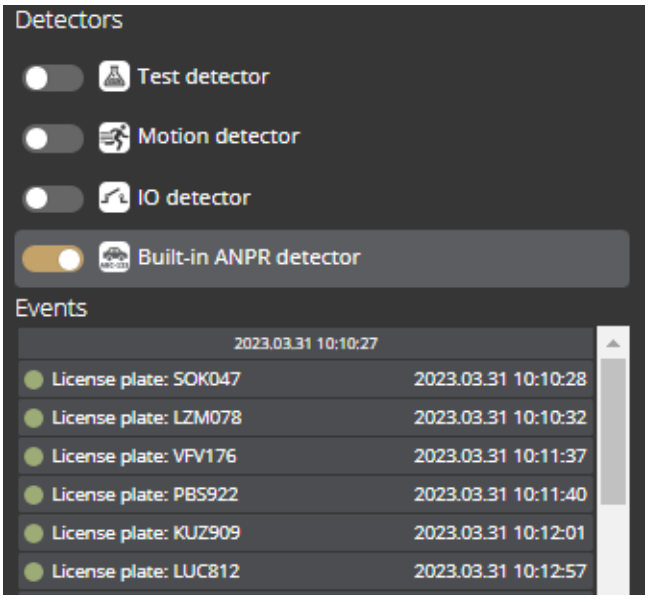
The small image that appears when the mouse cursor is positioned over the timeline shows a preview image of the video near that location.

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.

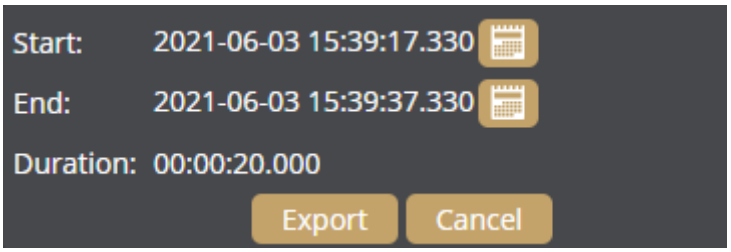


The events and timeline of each detector can be turned on/off by clicking on the appropriate detector button. Clicking on an event in the list, navigates the playback to the date and time of the event.

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.

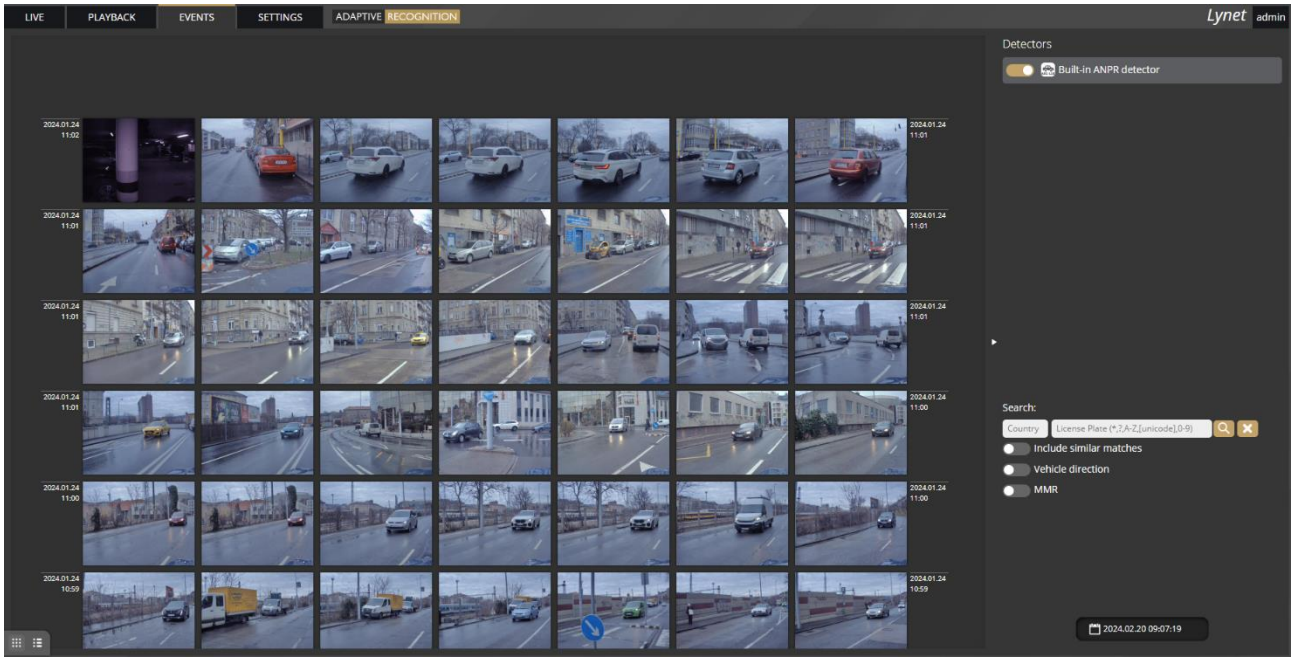


Drag the arrows with the mouse, and click the **[Calendar buttons]** next to "**Start**" and "**End**" to modify the exported time range. The duration of the video to be exported is displayed in the bottom line ("**Duration**").

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

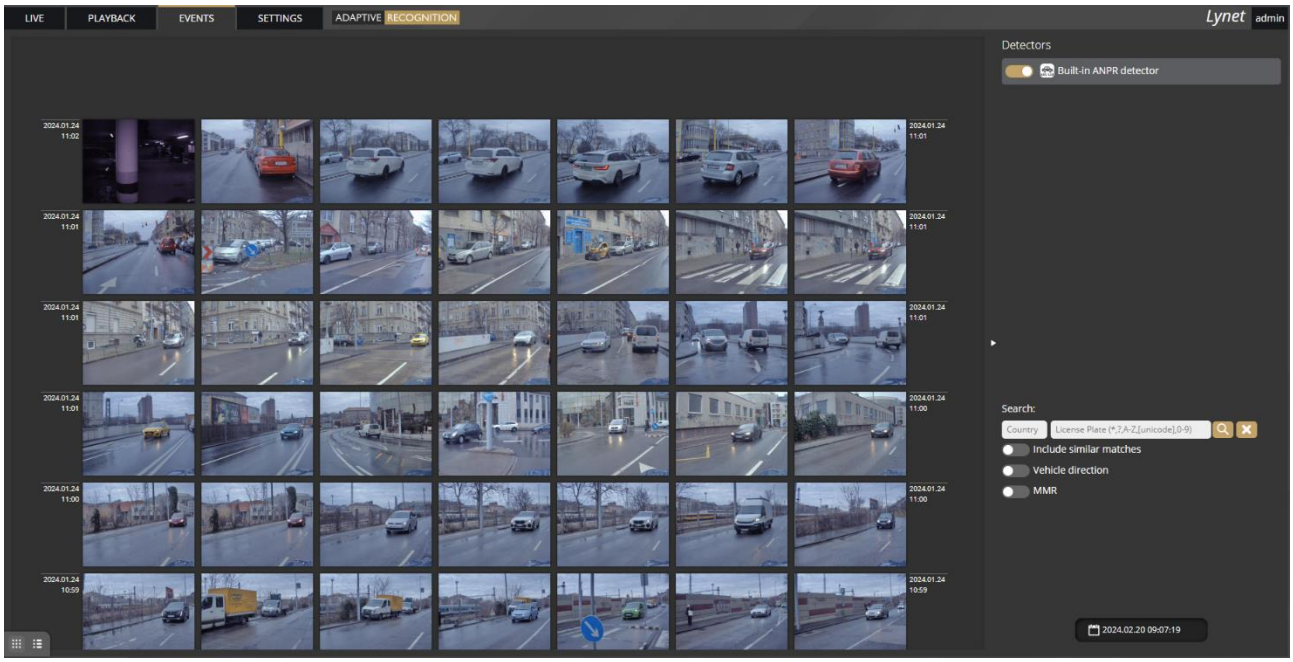
6. EVENTS

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

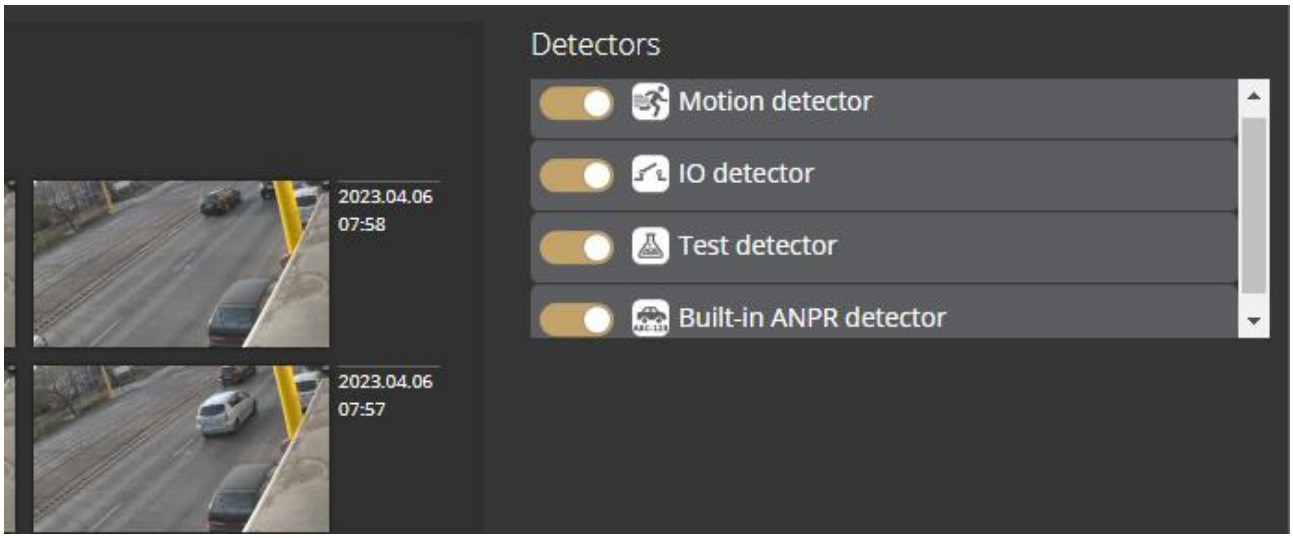


You can navigate between the recorded events in the event browser by scrolling through them with your **scroll wheel**. The events appear as small images. The latest events are at the top.

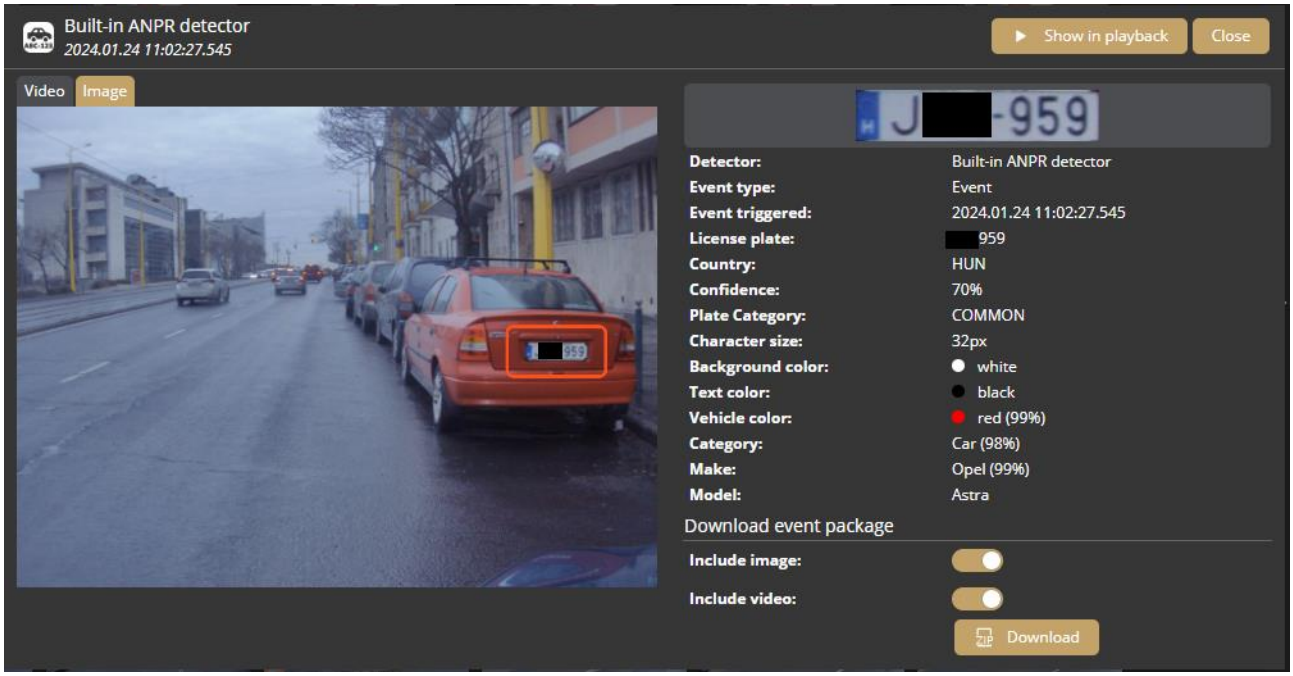
lUse the buttons in the bottom left corner to switch between **Grid view** and **Detailed view**.



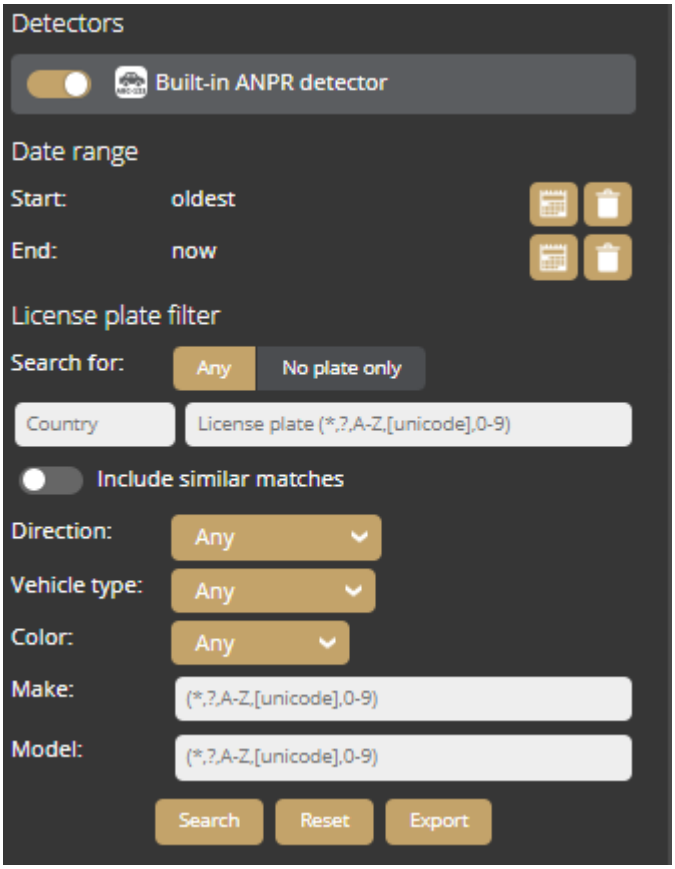
Hovering the cursor over an event, the detector related to the event is highlighted in the list on the right. Simultaneously, a video clip of the event will be loaded and played automatically.



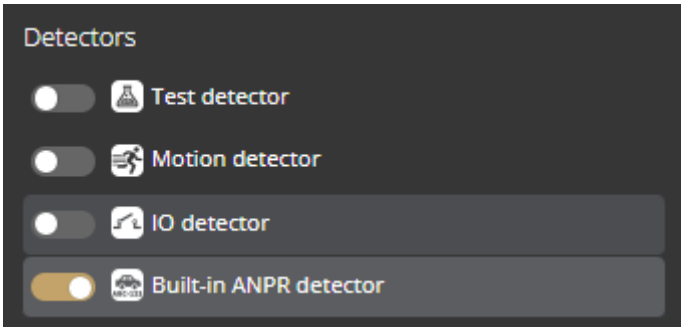
Clicking on an event brings up a detailed view of that event, including a video clip and any related image. The interface can be redirected to the **PLAYBACK** menu item by clicking on the **[Show in playback]** button. The data belonging to the event can be saved as a ZIP file by clicking on the **[Download]**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



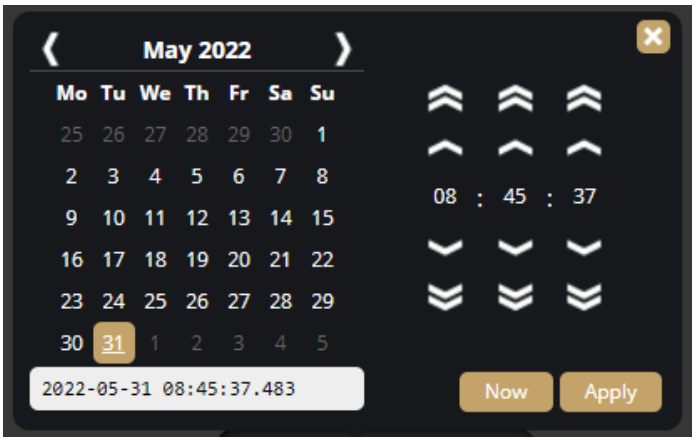
In the top right corner, you can choose to display Event details or Filter options.



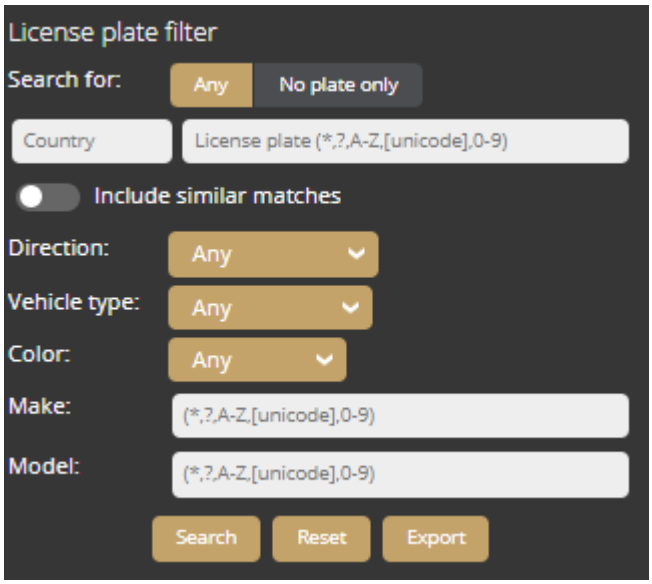
The configured detectors are displayed on the right. By moving the cursor over the detector, the events related to the detector will be highlighted in the event browser. By clicking on the detector, the display of its events can be turned on/off.



To set the Date range, click the calendar icon in the Start and End rows. After setting the time and clicking on the **[Apply]** button in the calendar, the browser jumps to the specified time.



An additional license plate search form is available. Similar license plates can be listed if the "**Include similar matches**" option is turned on. Searching based on vehicle direction; color, make and model data is also available. You can filter events without license plates by using the **No plate only** option.



Once the filtering is set, events matching the filtering criteria are displayed. In In addition, the data of events matching the filtering conditions can be downloaded from the camera in .csv format by clicking on the Export button.

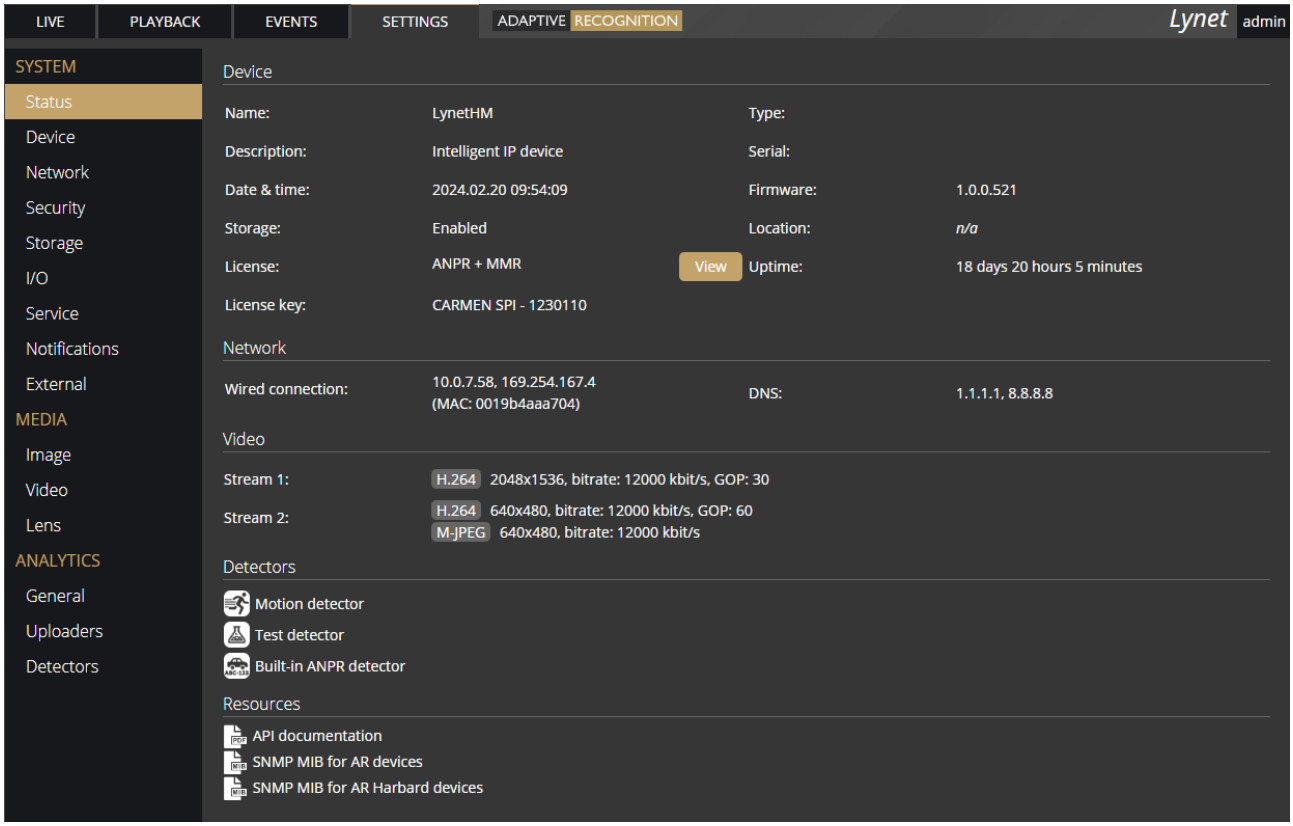
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	Timestamp,"Detector name","Detector ID","Event ID","Event code","Event type","ANPR text","ANPR confidence","ANPR country","ANPR direction","ANPR category","ANPR text color","ANPR make","ANPR model"																					
2	2024-04-07 10:19:01.917,"Built-in ANPR detector","{B2B78D79-B325-6C42-94C7-05C9B97CA66E}","{0FB10EEA-AAA5-E74A-93D9-73792B7F8C85}",114,"Event","SNM111",87,"HUN","Moving away","CAR","#000000","Volvo","XC60"																					
3	2024-04-07 10:20:49.619,"Built-in ANPR detector","{B2B78D79-B325-6C42-94C7-05C9B97CA66E}","{4F8FDC93-536B-F942-81BF-7298482EFC46}",114,"Event","TBR111",84,"HUN","Unknown","CAR","#808080","Volvo","XC40"																					

7. SETTINGS

The SETTINGS page contains all customizable parameters of the camera.

7.1. SYSTEM / STATUS

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



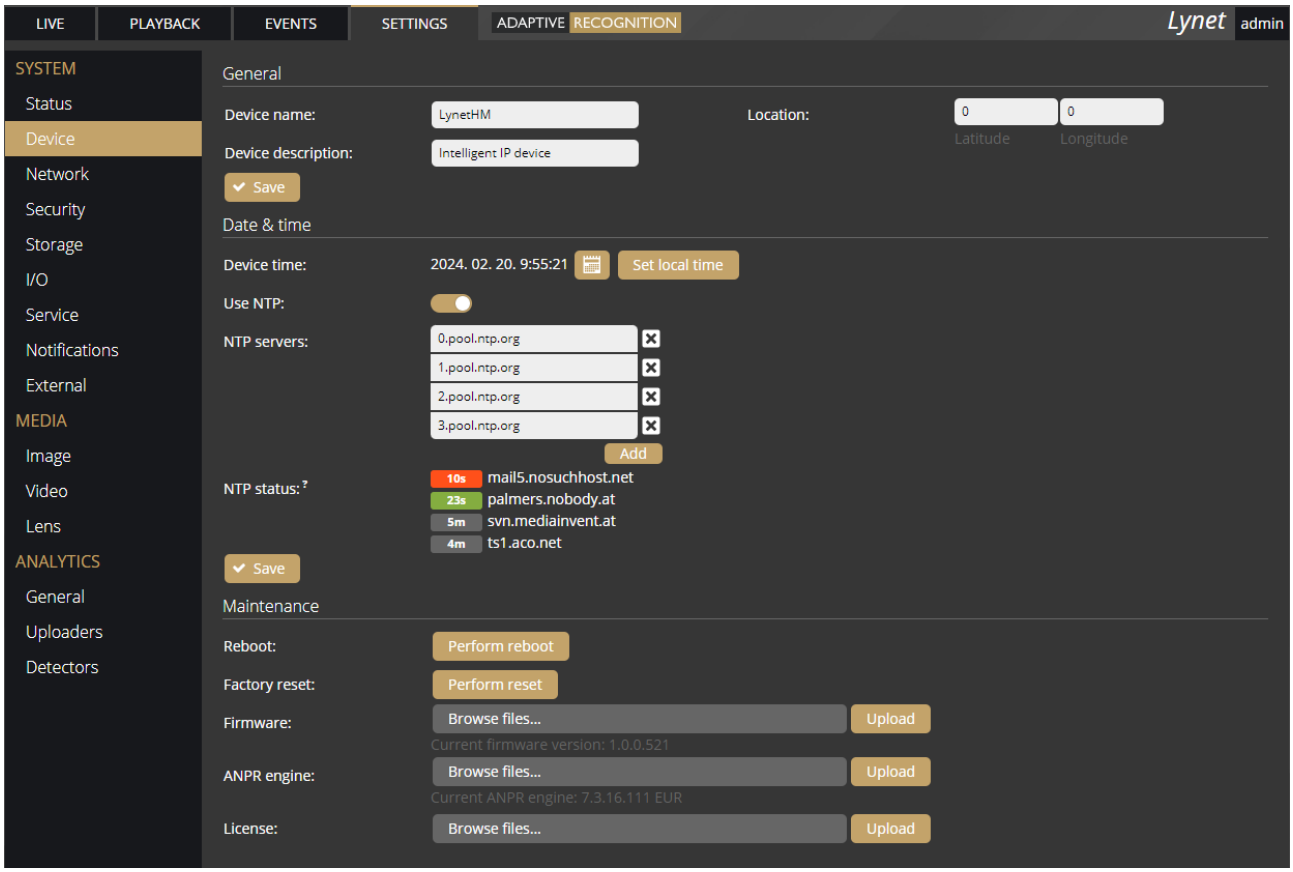
Click on the View button to display the license details. You can see which license is valid for which region and until which date.

In the Resources section you will find the API documentation and the SNMP MIB files.
More useful information on integration can be found on the following website:
https://github.com/adaptiverecognition/harbard-sdk

7.2. SYSTEM / DEVICE

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

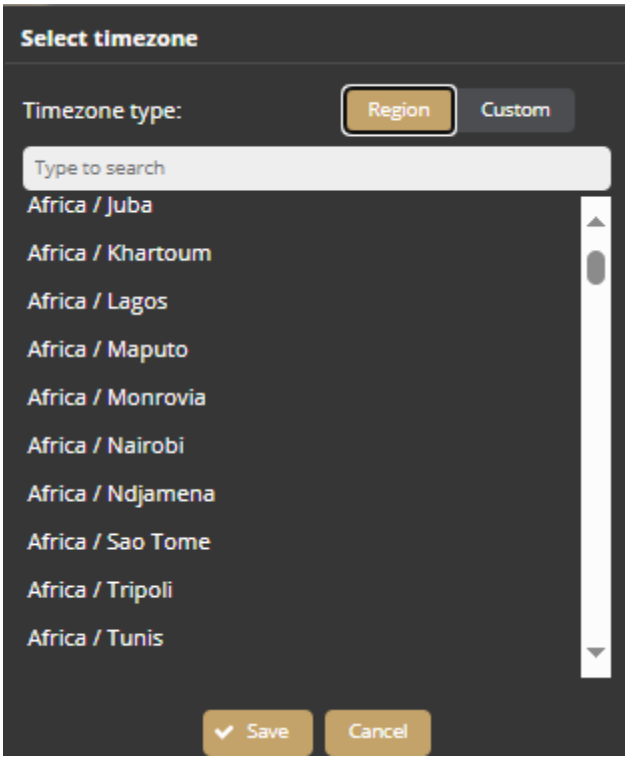
- Modify the name, description and location of the device
- Reboot the device remotely
- Perform a factory reset (after clicking on the button, the original manufacturer settings are restored except for the network settings)
- Set the date and time
- Upload firmware, ANPR engine and license
- Set location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

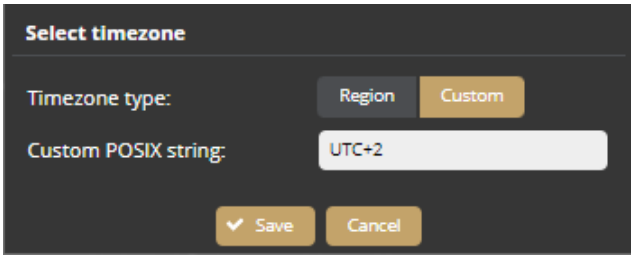


Date & time settings

The current time of the camera 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.

You can choose the **time zone** of the camera, select by region or enter a Custom setting.





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 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 camera 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 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.

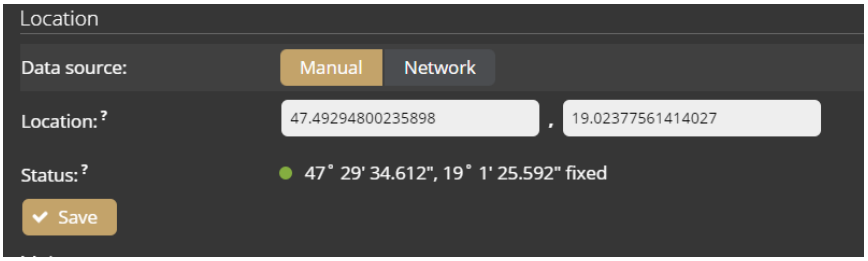
Green: Server is working and used for synchronization.

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

Location

GPS data can be entered manually or requested from a router with NMEA 0182 standard.

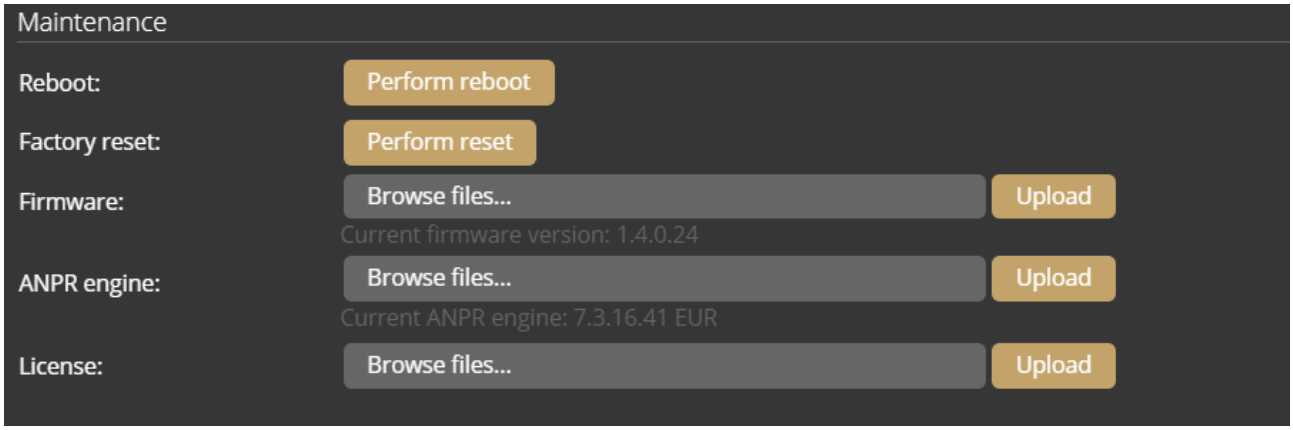
Location: Set the current location of the device in the geographic coordinate system using the decimal degree format.



Service URL: Set the URL of a server that can transmit location data using the NMEA 0182 standard.

Firmware, ANPR engine, License update

Click **[Browse files...]** on the field to be modified, then select the Firmware (.ifw), ANPR engine (.iep) 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 camera asks a security question whether you are sure about the modification. Choosing **[No]** interrupts the update process, and the camera operates with the previous settings. If you opt for **[Yes]**, the update continues. Updating and rebooting the camera may take a few minutes.

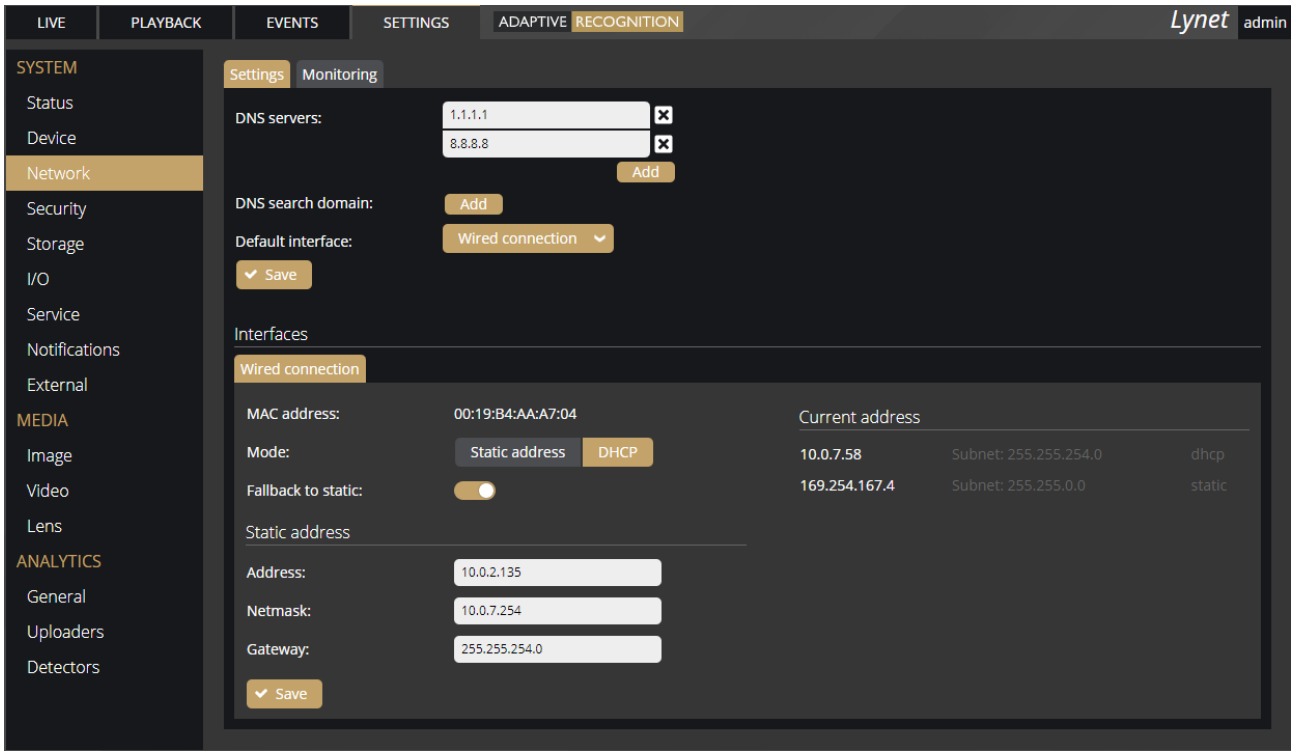
Important!

- During the update process **do not** unplug the camera.
- To use the camera with the on-board ANPR function, license file and engine file must be uploaded to the camera.

7.3. SYSTEM / NETWORK

The **Network** menu item hides the network settings. The IP address assigned to the camera can be static or dynamic. Click on the Add button to configure the DNS server.

Click on the Add button to configure the DNS search admin domain.

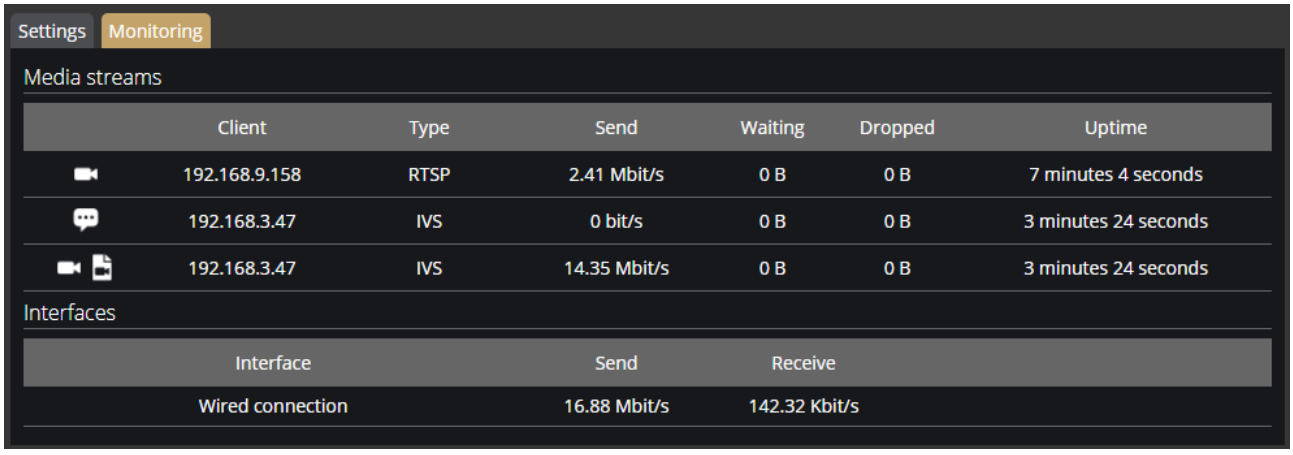


Fallback to static

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

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.



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

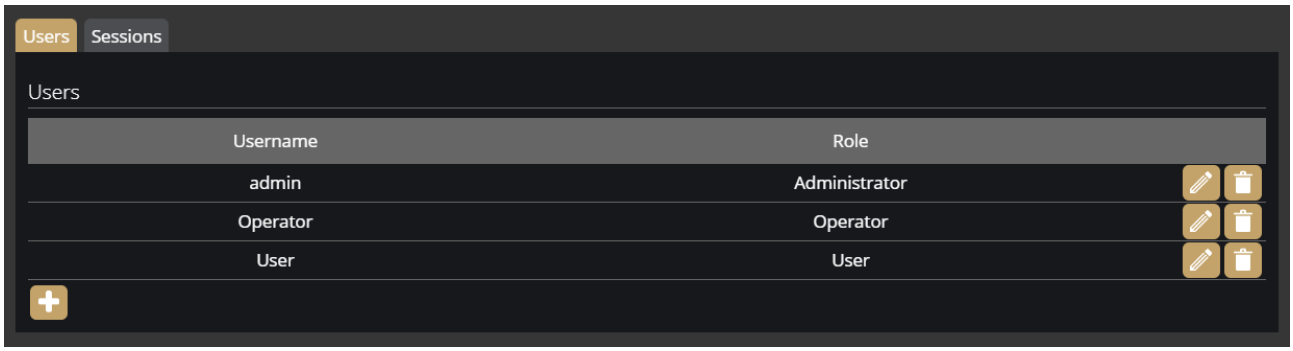
The default user and password are "**admin**".

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:

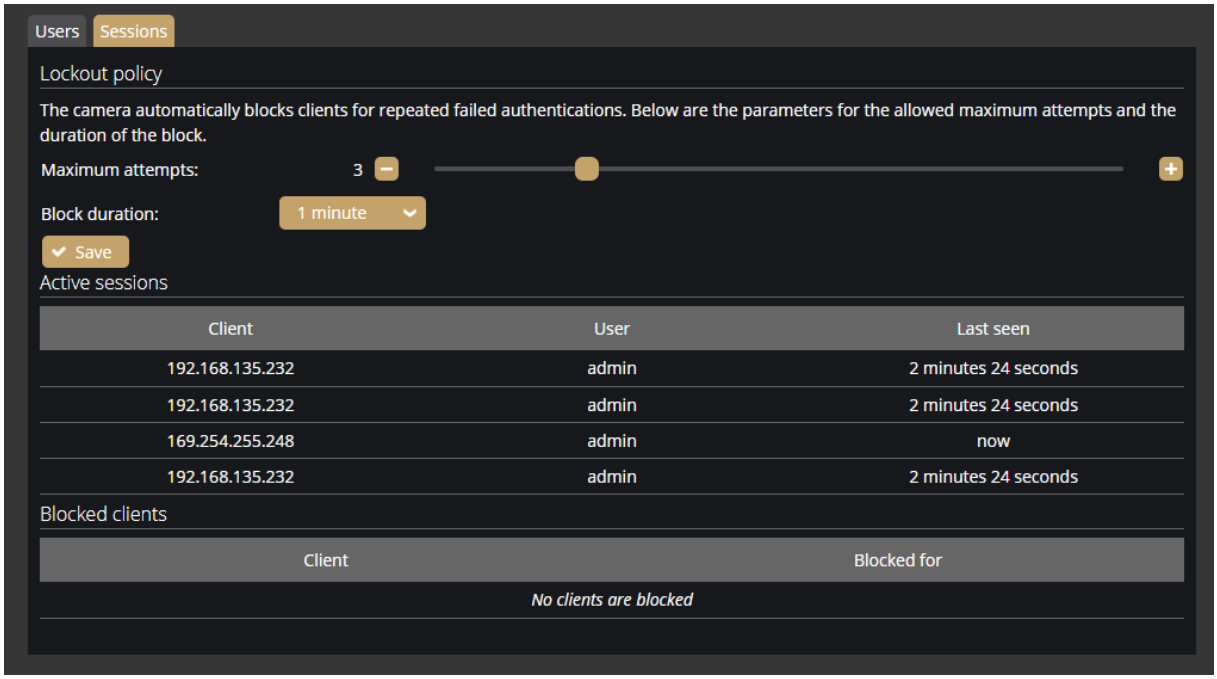
1. **Administrator**: The administrator can access and edit all parameters of a camera.
2. **User**: The user can view but not edit the parameters of the camera. Some pages containing sensitive information may be hidden.
3. **Operator**: The operator has the same privileges as a user but can operate the camera zoom and focus (if available).



Sessions

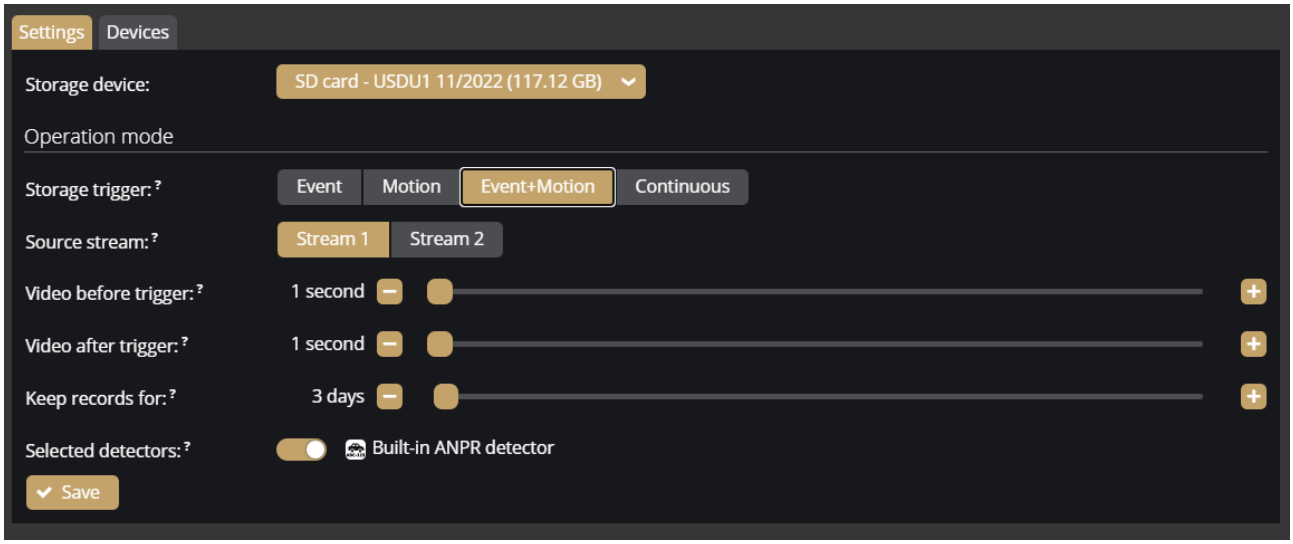
At **Lockout policy,** the maximum number of failed login attempts can be adjusted. After reaching the specified number, the device blocks that session. By default, after three failed login attempts, the camera blocks the IP address of the client for a minute. Note that the number of **Maximum attempts** 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 the selected detector(s).
- **Motion**: When the camera 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 or motion.

The properties of the video stream to be recorded for an event can be selected by specifying the **Source Stream** parameter. **Stream1** represents the higher resolution, primary stream and **Stream2** represents the lower resolution, secondary stream. See also ***{Media / Video}***.

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.

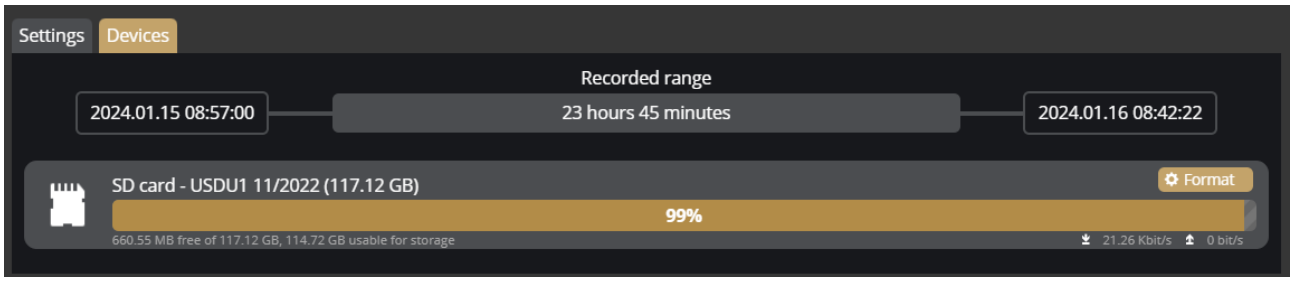
Keep records for: Recordings are automatically deleted from the storage medium after the specified time. You can set the number of days after which data is deleted from the SD card.

Selected detectors

It may not be necessary to record at every detector signaling. Thus, the user can select which detector signals should trigger the recording.

Devices

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



Formatting the SD card

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 is to start the operation. The capacity bar indicates the remaining time of the formatting process.

Important!

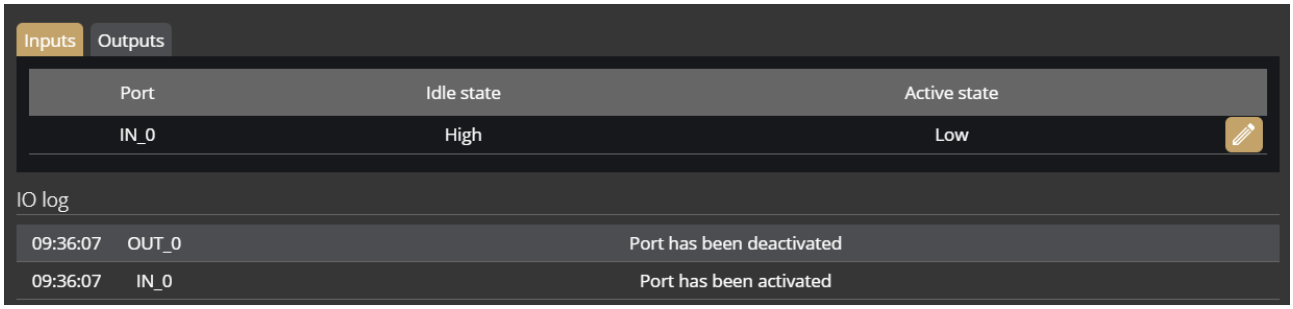
The formatting deletes every data from the SD card.

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

7.6. SYSTEM / IO

The I/O ports allow communication between the information processing system (in this case, the camera) and the outside world (e.g., a person, computers, alarms, barriers, etc.).
The conditions related to the information coming from the outside world to the camera and arriving through the input ports (input signals) can be adjusted by clicking the **[Edit]** button ( ) at the **Inputs** tab. When the **port's active state** is set to "**Low**", the signal is only received from a detector if the device connected to the camera sends about 0-1 VDC. When the **port's active state** is set to "**High**", then the signal is received from a detector if the device connected to the camera sends about 2-24 VDC.

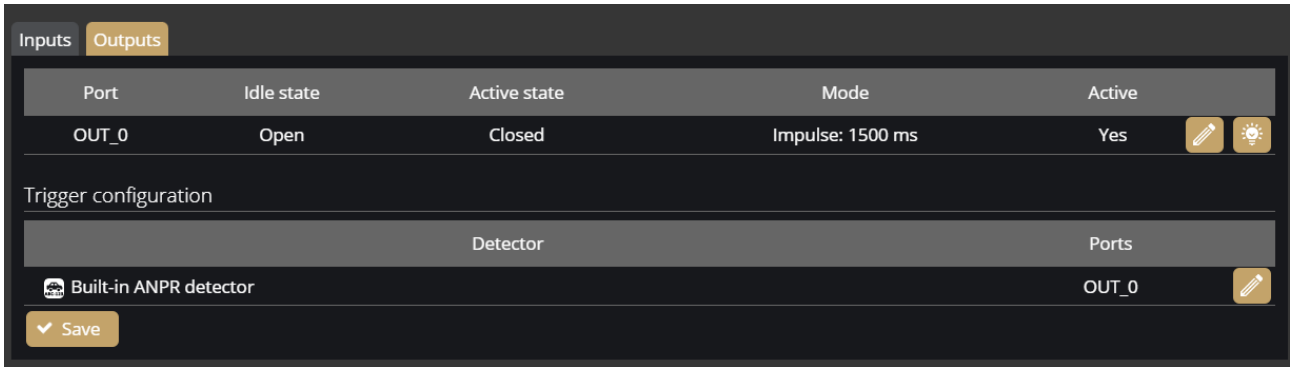
An ONVIF device recorded in the External menu can send a Digital Input signal to Lynet, which can trigger the camera to record an event. Lynet can also send a Digitalis Ouput signal to this ONVIF device. The Digitalis Output signal may be triggered by events produced by the selected detector.



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

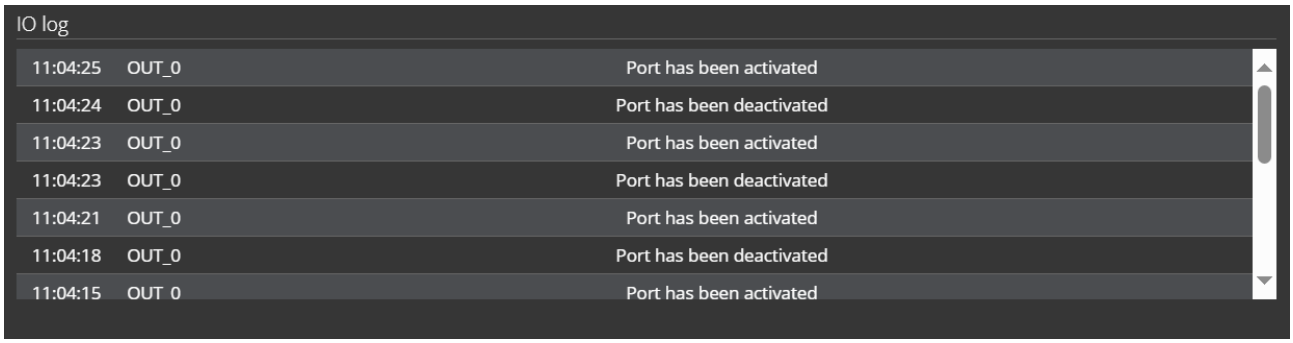
The following parameters can be adjusted after clicking on the **Edit** button  :

- **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.



It can be selectable which detector signal should activate the output port of the camera at **Trigger configuration** section.

You can monitor the state changes of the input/output ports at **IO log** section:



7.7. SYSTEM / SERVICE

AR Discovery Tool

- **Allow discovery:** You can allow the AR Discovery Tool to discover the camera.

Bonjour:

- **Allow discovery:** You can allow the Bonjour to discover the camera.

IVS

- **Service port:** The service port of the IVS (Intellio Video System) can be specified by filling in the field.

ONVIF

- **Service enabled:** ONVIF feature can be enabled or disabled.

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.

SNMP

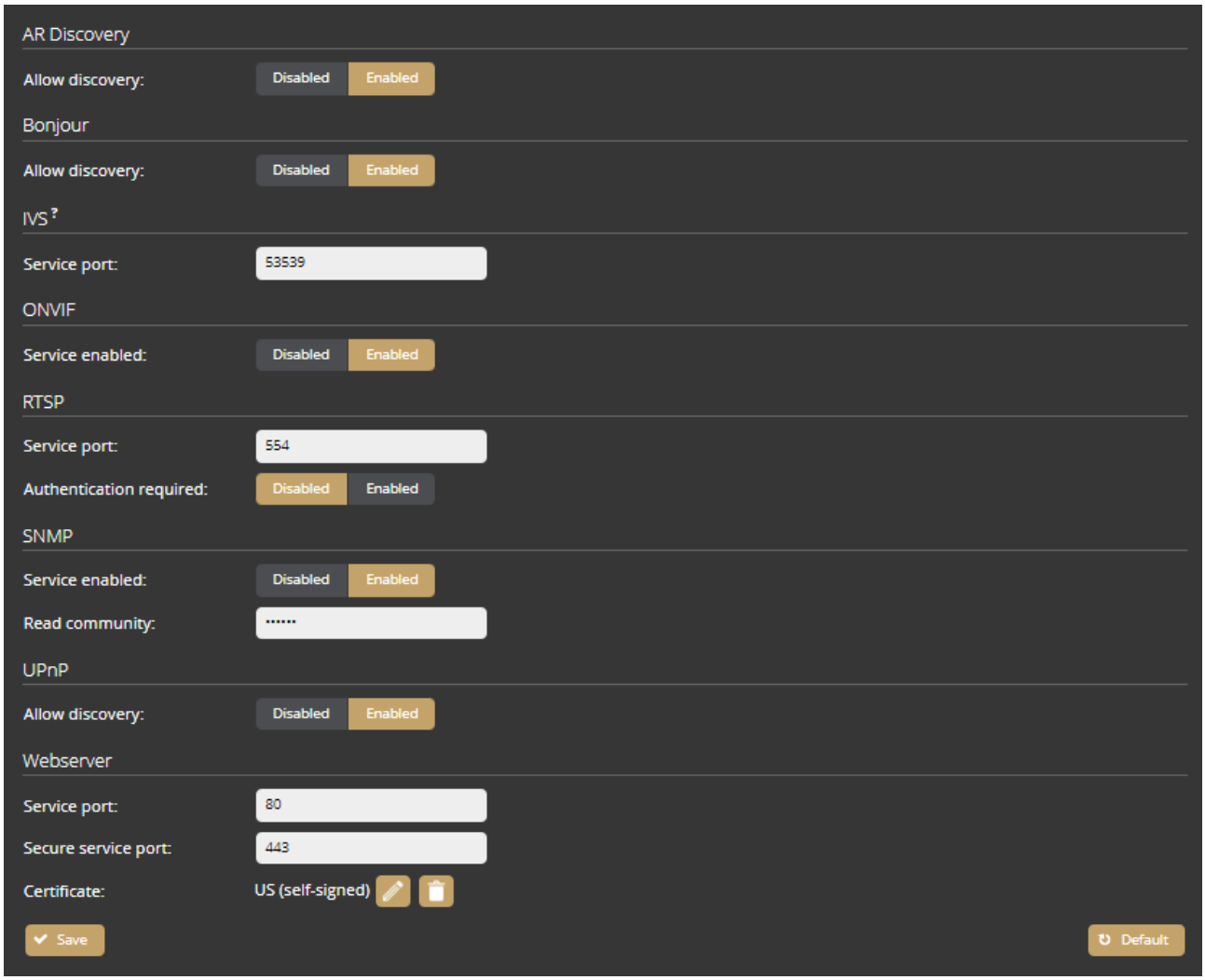
- **Service enabled:** The SNMP service can be enabled or disabled.
- **Read community:** Password required for the camera properties on SNMP.
- **MIB** file can be found in ATSS.

UPnP

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

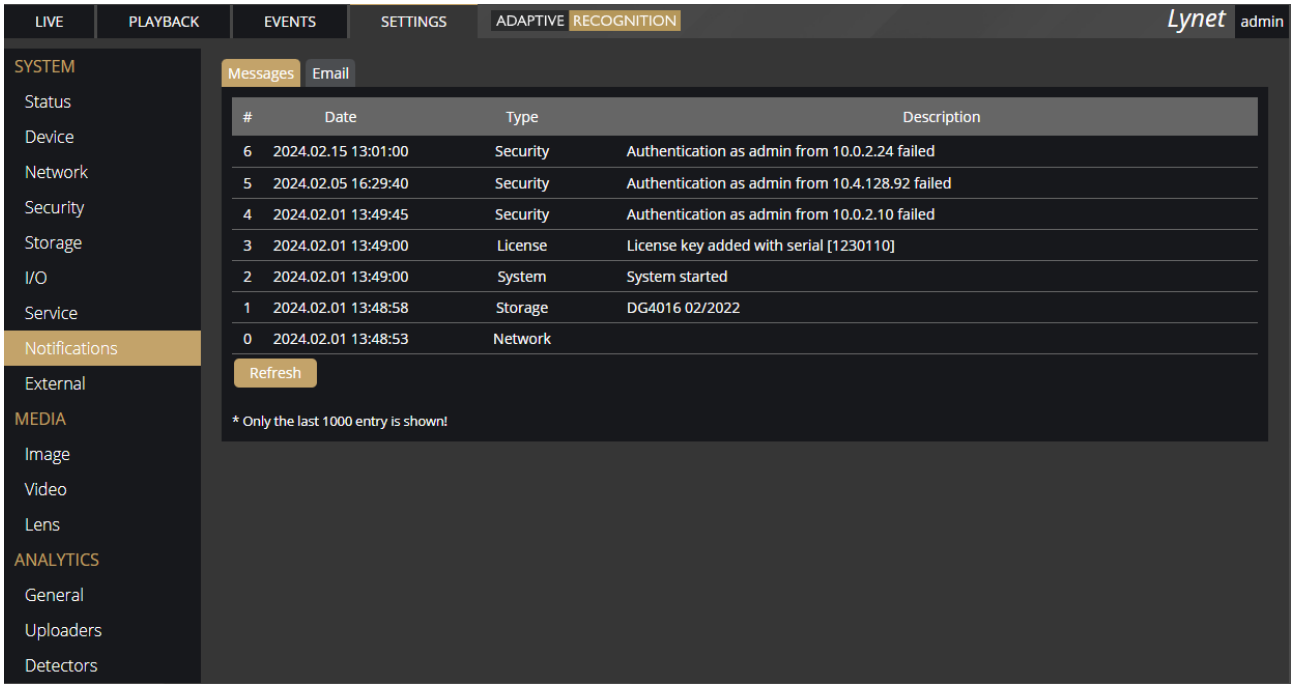
Webserver

- **Service port / Secure service port:** The service ports of the Webserver can be specified by filling in the field.
- **Certificate:** The camera has its own SSL certificate. To modify the SSL certificate, click on the Edit button and upload the Certificate and Key.


AR Discovery
Allow discovery:
Disabled
Enabled
Bonjour
Allow discovery:
Disabled
Enabled
IVS
Service port:
53539
ONVIF
Service enabled:
Disabled
Enabled
RTSP
Service port:
554
Authentication required:
Disabled
Enabled
SNMP
Service enabled:
Disabled
Enabled
Read community:
UPnP
Allow discovery:
Disabled
Enabled
Webserver
Service port:
80
Secure service port:
443
Certificate:
US (self-signed)
Save
Default

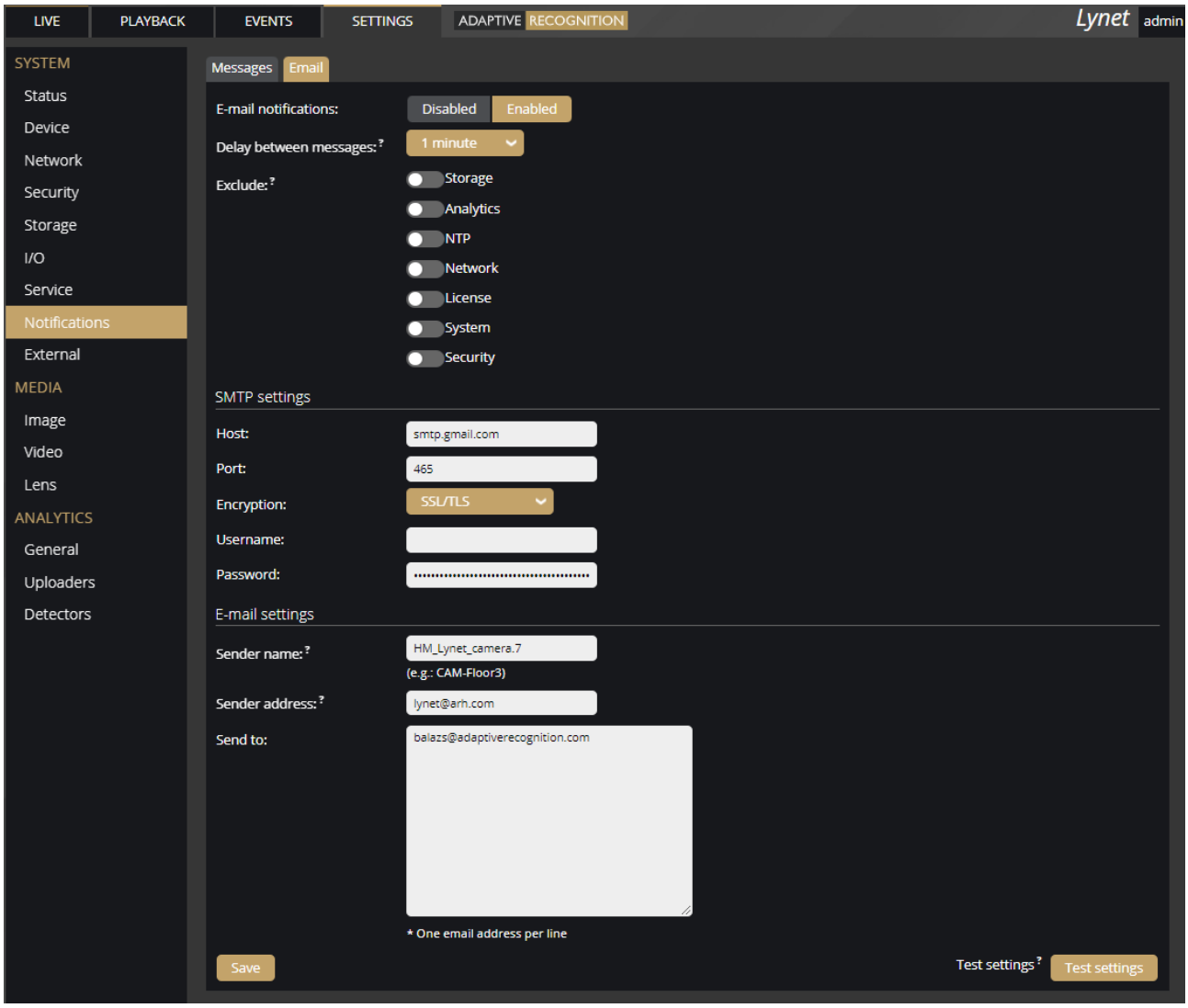
7.8. SYSTEM / NOTIFICATIONS

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



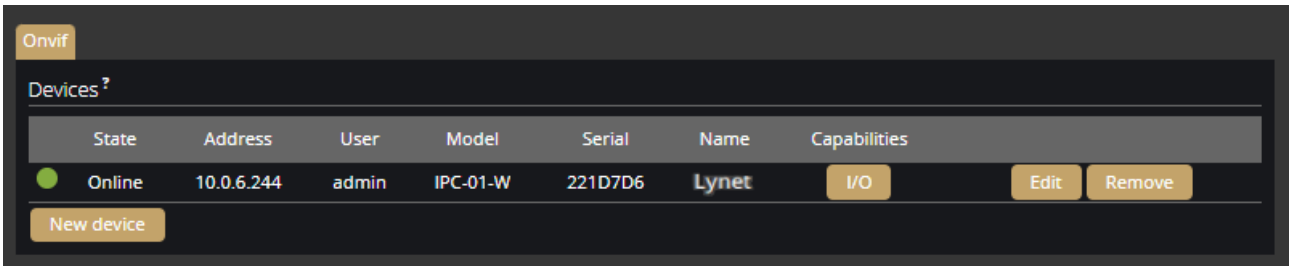
In the **Email** tab, you can specify the email settings for sending messages. The following parameters can be adjusted after clicking on the **[Enabled]** button:

- **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.



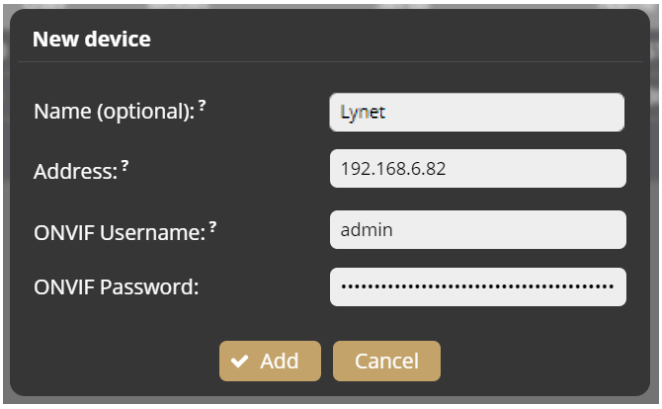
7.9. SYSTEM / EXTERNAL / ONVIF

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.



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.



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. SYSTEM / EXTERNAL / MQTT

You can manage the MQTT communication settings in the External menu.

Client ID: The name of the device can be entered.

Broker URL: Url to a MQTT Broker service with the following formats:

mqtt://<address>:<port>[/path]*[?query] - TCP

mqtts://<address>:<port>[/path]*[?query] - Secure TCP

ws://<address>:<port>[/path]*[?query] - Websocket

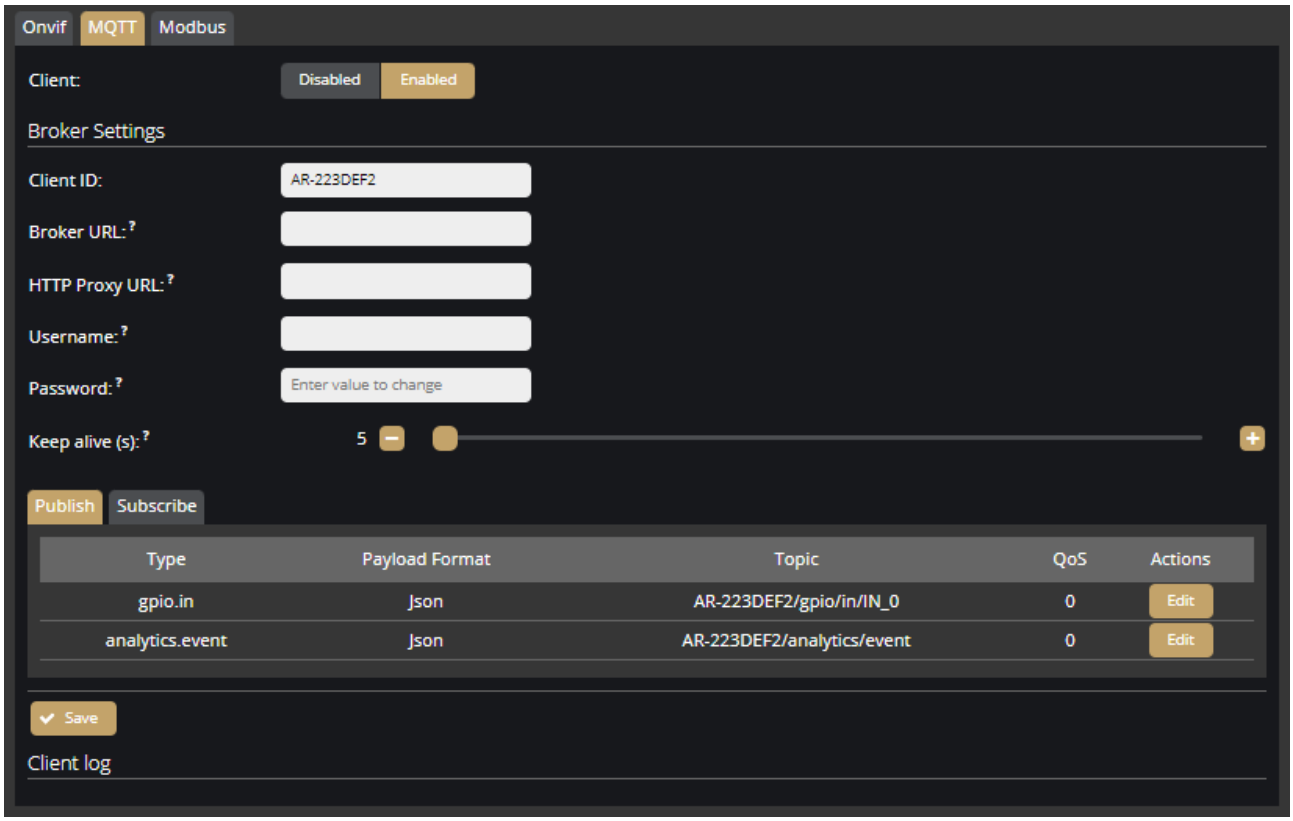
wss://<address>:<port>[/path]*[?query] - Secure Websocket

HTTP Proxy URL: Url to a HTTP Proxy Server for transfering MQTT communication.

Username: Username for the MQTT Broker service.

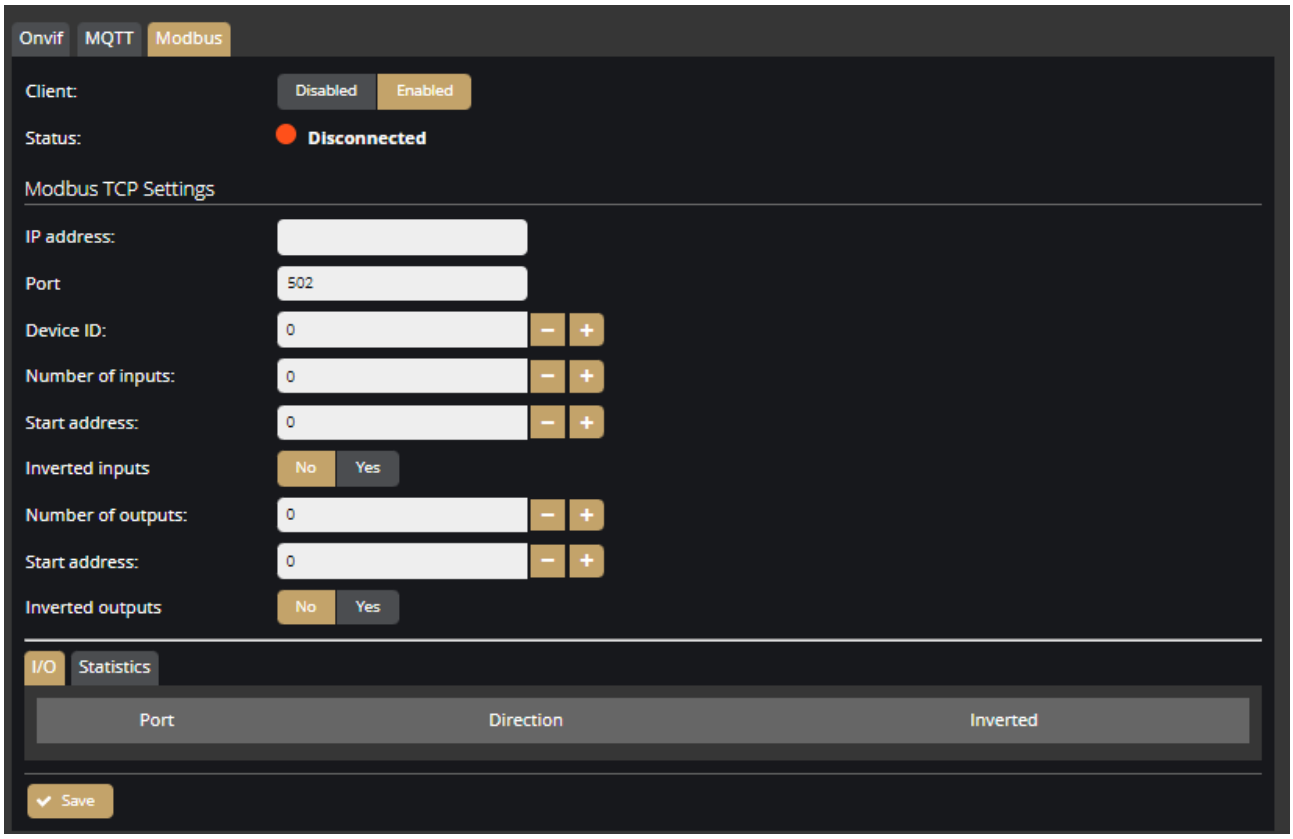
Password: Password for the MQTT Broker service.

Keep alive (s): MQTT keepalive message frequency in seconds.



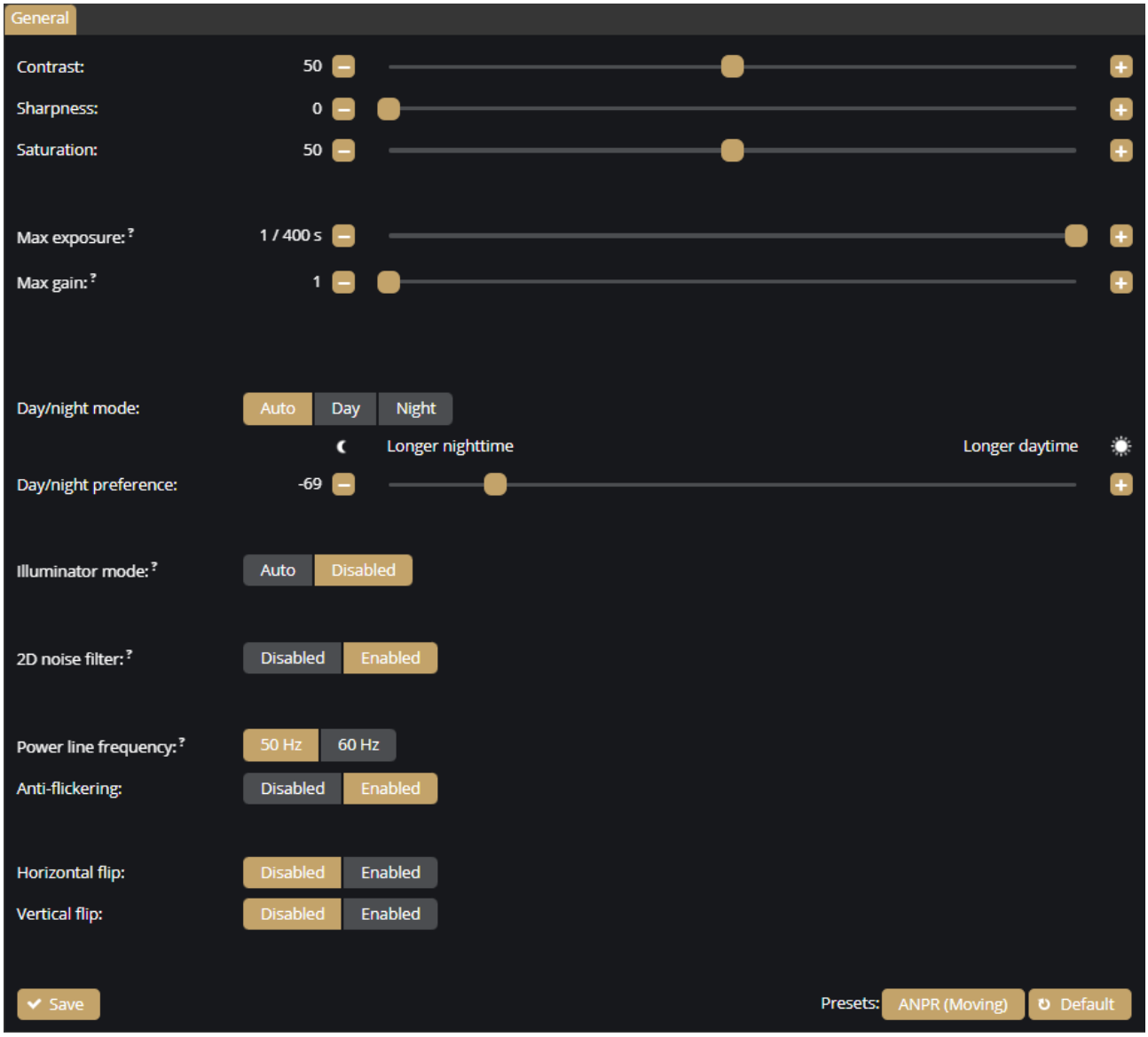
7.11. SYSTEM / EXTERNAL / MODBUS

In the External menu you can manage devices connected on Modbus protocol.



7.12. MEDIA / IMAGE

On this configuration interface, you can optimize the image quality.



Here, the following elements can be fine-tuned:

- **Contrast:** Brightness difference between the brightest and darkest points of the image.
- **Sharpness:** It makes the details sharper or more blurred.
- **Saturation:** It defines the saturation of colors in the image.
- **Max exposure:** The camera cannot expose longer than the specified value.
- **Max gain:** The digital gain cannot be greater than the specified value.
- **Day/night mode:** Day and night mode can be chosen manually or automatically.
- **Day/night preference**: A preferred setting of day or night mode can be defined, which is applied in the case of automatic shifting.

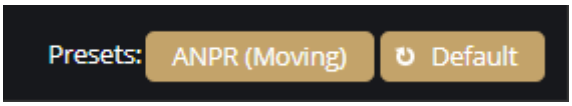
- **Illuminator mode:** The infra LED can be operated automatically or can be disabled.
- **Illuminator power:** You can adjust the brightness of the infra LED.
- **2D noise filter:** Noisy pixels are filtered based on the environment of a pixel.
- **Power line frequency:** The power line frequency of the region where the camera is installed. It is used to eliminate vertical flickering in artificially lit areas.
- **Anti-flickering:** Reduces flickering caused by light sources that flicker due to the mains frequency
- **Horizontal flip:** Flips the image horizontally when enabled.
- **Vertical flip:** Flips the image vertically when enabled.

Presets

You can select preset values which are calibrated to different situations. These elements are not necessarily ideal settings for all cases; they are to be considered as guidelines. They generally show acceptable image settings for certain scenarios.

These preset values are the following:

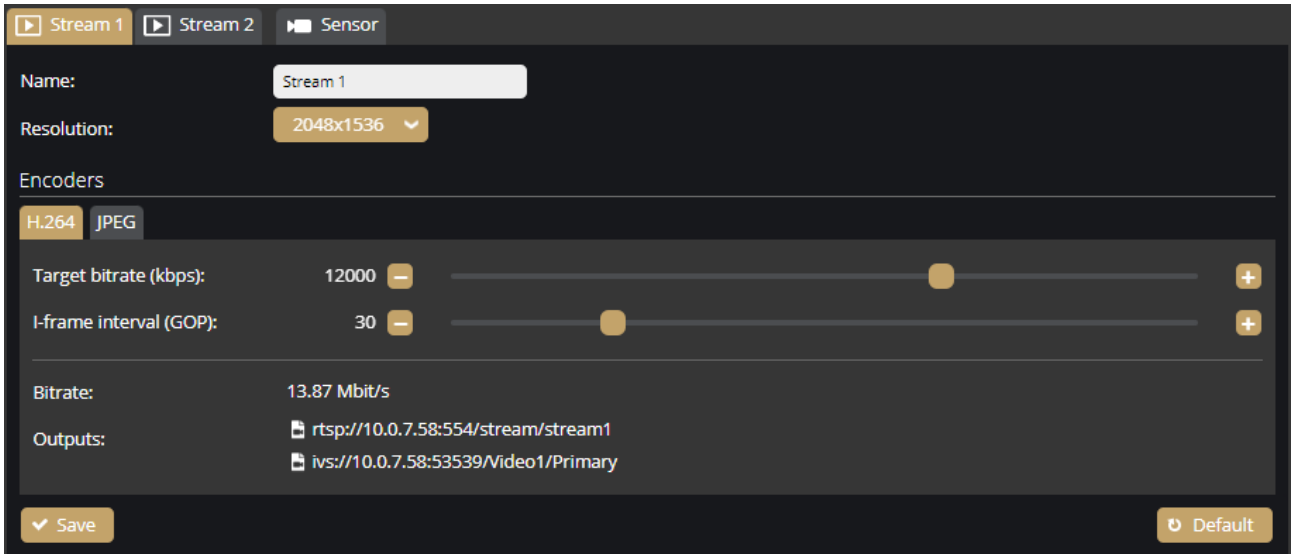
- **ANPR (Moving)**: Ideal for free-flow, slow traffic license plate reading
- **Default**: General setting for basic motion detection



Preset buttons show if current settings match a preset.

7.13. MEDIA / VIDEO

When clicking on the **Video** menu item, the functions related to the video encoders are displayed. Above these, the live stream of the camera remains visible.



On this interface, the settings (name, resolution, frame rate, etc.) of **Stream 1** and **Stream 2** can be performed separately.

Encoders

- **Target bitrate (kbps):** The bandwidth size can be adjusted.
- **I-frame interval (GOP):** The density of the I frames can be specified.
- **Quality:** The quality of the images can be specified.

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

Stream1 / H.264 stream: rtsp://LYNET_IP:554/stream/stream1
Stream1 / JPEG image: http://LYNET_IP/image/stream1

Stream2 / H.264 stream: rtsp://LYNET_IP:554/stream/stream2
Stream2 / MJPEG stream: http://LYNET_IP/stream/stream2
Stream2 / MJPEG stream: rtsp://LYNET_IP:554/stream/stream2/mjpeg
Stream2 / JPEG image: http://LYNET_IP/image/stream2

Authentication (username and password) is required to use the streams. The stream links with authentication are listed below:

Stream1 / H.264 stream: rtsp://USERNAME:PASSWORD@LYNET_IP:554/stream/stream1
Stream1 / JPEG image: http://USERNAME:PASSWORD@LYNET_IP/image/stream1

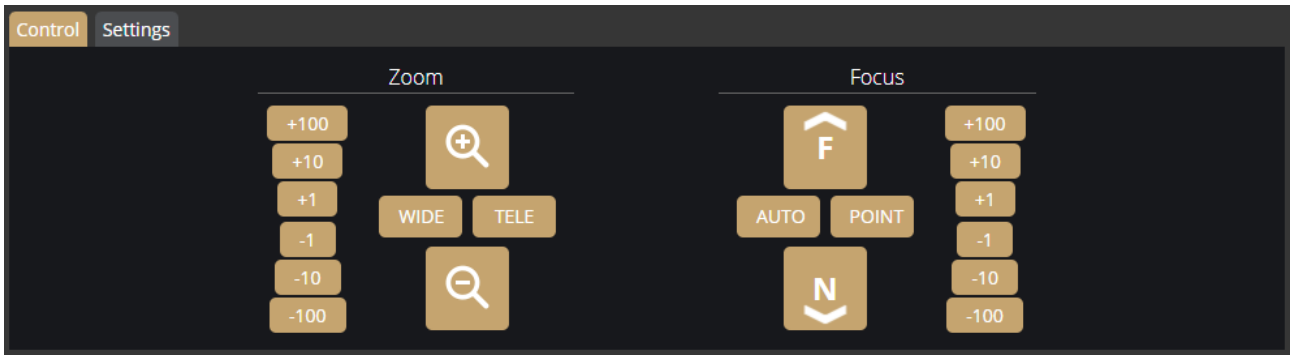
Stream2 / H.264 stream: rtsp://USERNAME:PASSWORD@LYNET_IP:554/stream/stream2
Stream2 / MJPEG stream: http://USERNAME:PASSWORD@LYNET_IP/stream/stream2
Stream2 / MJPEG stream: rtsp://USERNAME:PASSWORD@LYNET_IP:554/stream/stream2/mjpeg
Stream2 / JPEG image: http://USERNAME:PASSWORD@LYNET_IP/image/stream2

Sensor

Under the **Sensor** tab, you can adjust the aspect ratio to 4:3 or 16:9.

7.14. MEDIA / LENS

On this interface, you can fine-tune the lens of the device: the **Focus** and the **Zoom** can be customized by using the corresponding buttons. See also ***{Zoom & Focus}***.



WIDE / TELE buttons: the lens can be adjusted to the end position using these buttons.
Press and hold the **+** (magnifying glass with + sign) and **-** (magnifying glass with - sign) buttons to decrease or increase the camera's angle of view. The speed of adjustment increases proportionally to the time the buttons are held down.
Press and hold the **F** (Far) and **N** (Near) buttons to set the focus to Far and Near, respectively. The speed of adjustment increases proportionally to the time the buttons are held down.
The **+100/+10/+1/-1/-10/-100** buttons move the lens in the given direction by the given number of units.

The camera possesses autofocus. The autofocus function is enabled by default, but it can be disabled on the **Settings** tab with the **Focus on zoom** option. If it is enabled, the camera starts to look for the correct focus value by itself to achieve the sharpest image possible. This process begins simultaneously with the zooming.
In the manual, disabled state, the autofocus can be initiated with the **[AUTO]** button in the **Focus** section.

With the **[POINT]** button you can select a specific point of the image what you want to be the sharpest region in the picture.

Settings / Lens correction: This setting can correct lens distortion occurring at specific zoom settings.

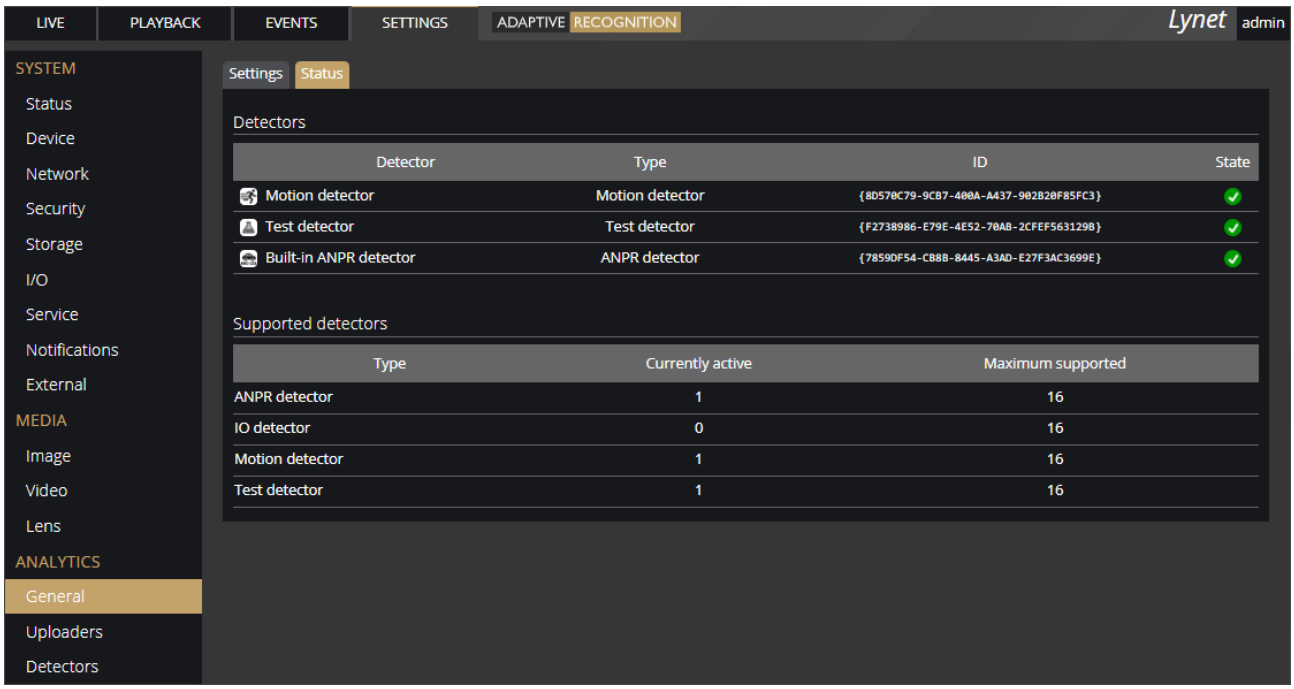
7.15. ANALYTICS / GENERAL

Settings

You can use the Settings tab to set the resolution and quality of the images – specified in Stream1 or Stream2 **{*MEDIA / Video*}** – to be assigned to events when uploading via API, GDS, HTTP or FTP.

Status

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



7.16. ANALYTICS / DETECTORS

You can add, modify or delete the camera's detectors in this window.
The list of detectors differs by camera type except for Motion engine, which can be found on all types.

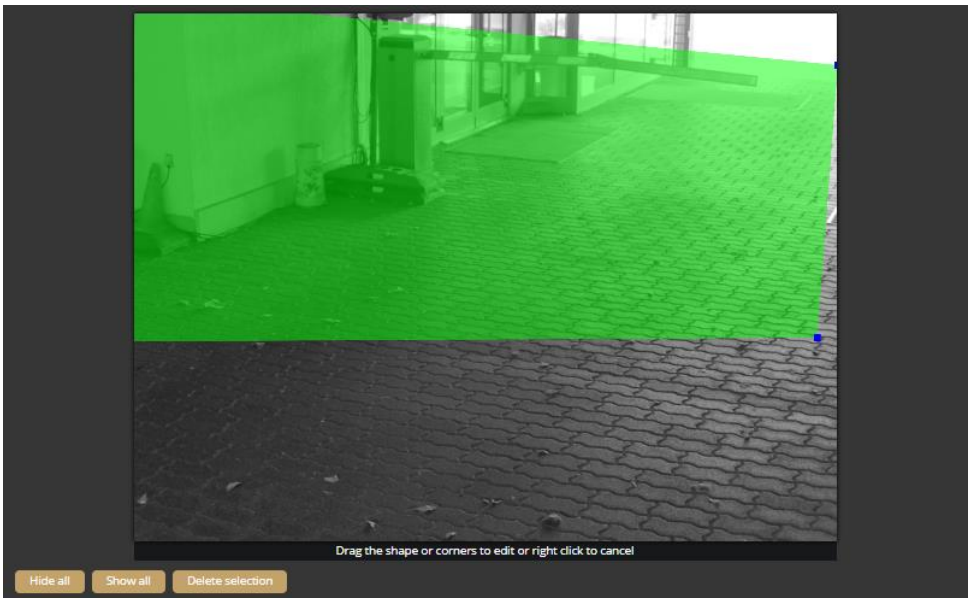
7.16.1. MOTION ENGINE AND GENERAL USE OF MASKS

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

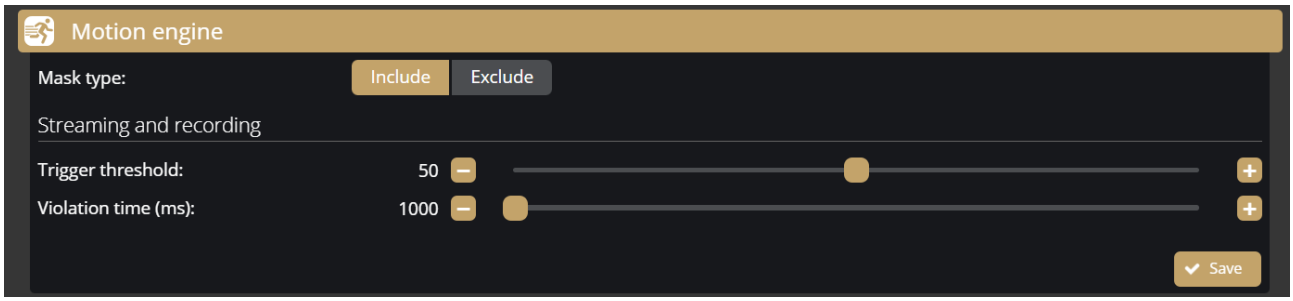
> **Note**
>
> The Motion engine does not induce events; it is responsible for the setting of the motion-based recordings. As such, it can be found on all camera types.

If you click on the Motion 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.

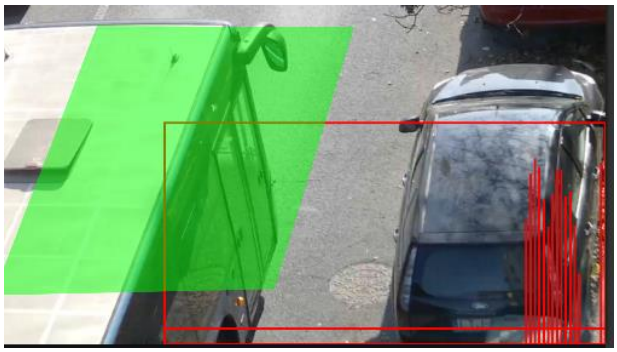
The mask can be modified by clicking on the green area.



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



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}***.

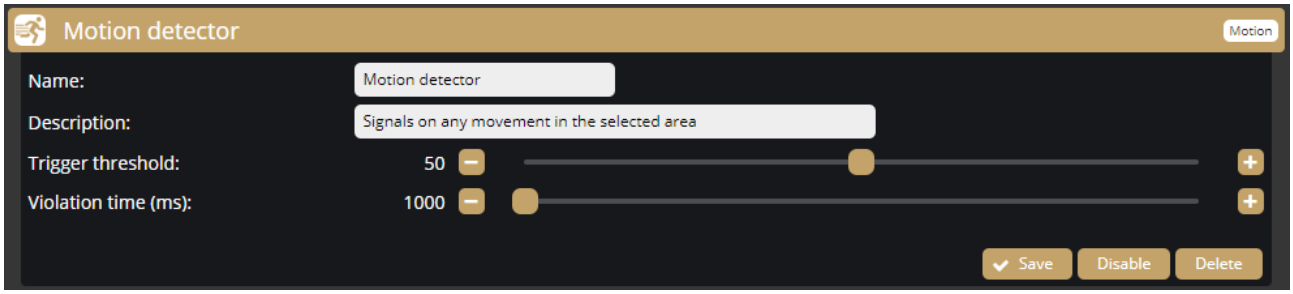


Violation time: you can set how long the movement must persist for the engine to become active.

7.16.2. 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**: you can set how long the movement must persist for the detector to become active.



ANPR Engine and ANPR Detector

The ANPR Engine and the ANPR Detector(s) jointly perform the reading of license plates.

Important!

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

7.16.3. ANPR ENGINE

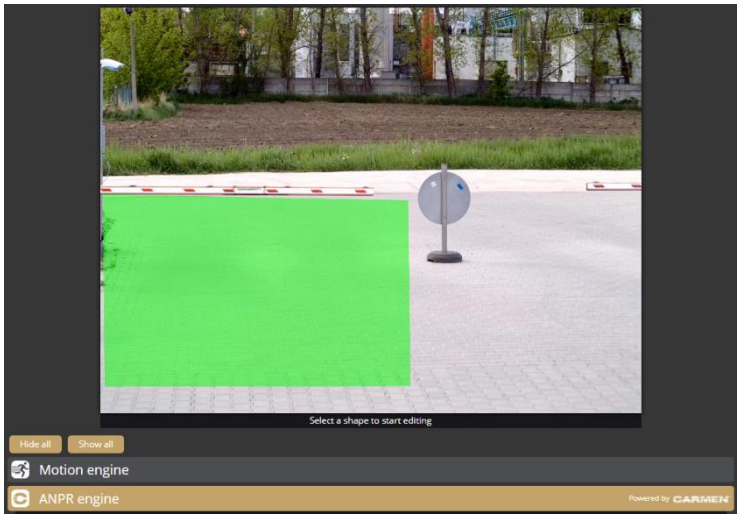
You can choose between the Onboard ANPR Engine running on the camera or the Carmen Cloud service for license plate recognition.

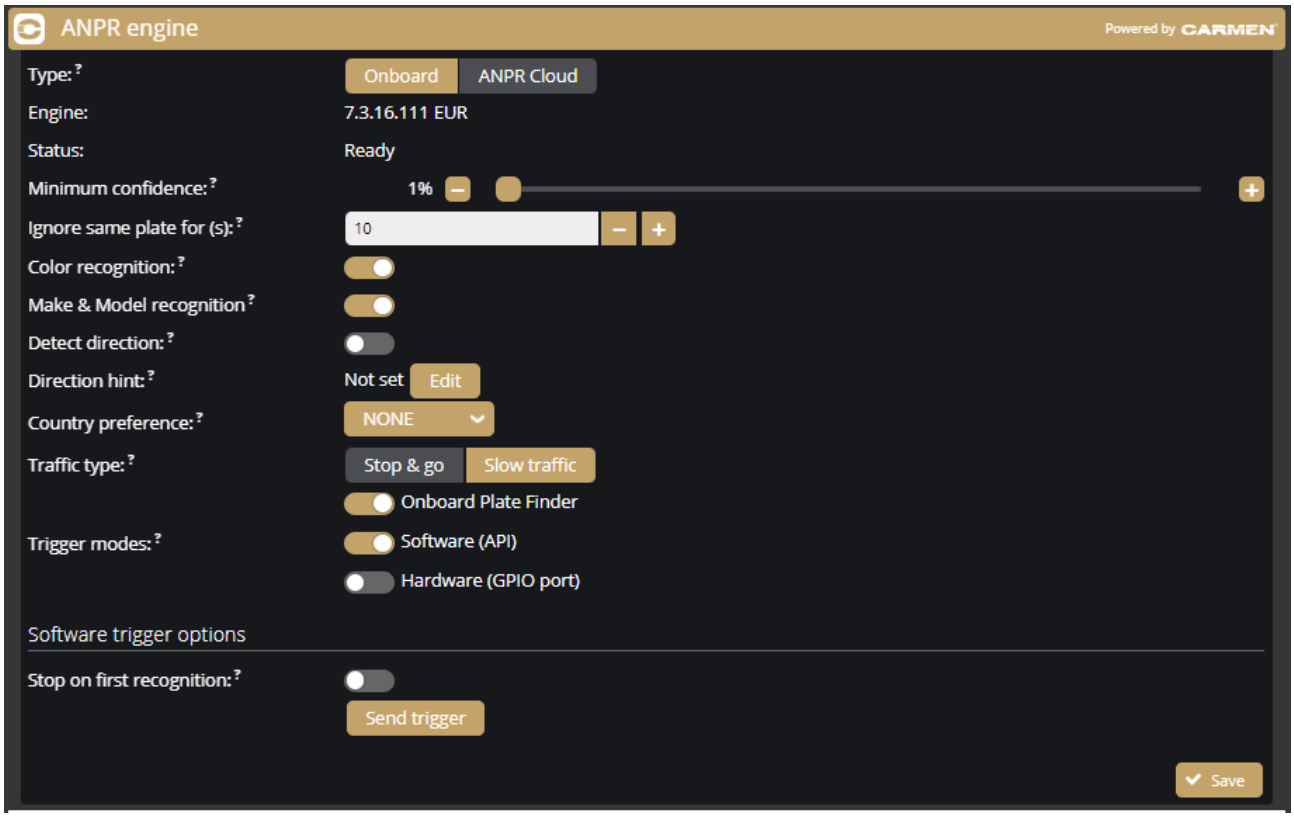
You can create a mask on the image, after which license plates will be detected only in the selected area. It is recommended to define the area where license plates are expected to appear on the image with the proper character size, and from the proper angle of view (the license plate should be visible from as straight direction as possible, and should not be distorted or slanted).

The expected area of license plates can be plotted on the image with a polygon that contains a given number of vertices.

Important!

Although the ANPR engine recognizes the license plates in the image, to make it work properly, the area around the license plate must be visible (front or rear of the vehicle).





In the case of **Onboard ANPR**, you can adjust the primary functions of the ANPR engine. The parameters that can be set vary depending on the engine:

- **Engine:** It shows the current ANPR engine version and region.
- **Minimum confidence:** Use the slider to define what reading confidence percentage should trigger an event. A low value potentially results in more reading errors. A high value potentially leads to missed reading cases. Therefore, values between **50-70%** are recommended. Check the **ANPR engine status** OSD layer to see the detected license plate numbers and their confidence values and then set the desired minimum confidence level based on the results. See also ***{Overlay}***.
- **Ignore same plate for (s):** Set a time limit for previously-read license plates. This value determines a waiting time in seconds before the same license plate is read again.
- **Country preference:** The selected region is prioritized and read with a higher confidence value by the license plate reading system. In comparison, license plates from other regions are managed with a lower confidence value.
- **Traffic type:** Selecting the proper traffic scenario increases the precision of detections.
    - **Stop & go:** Vehicles stop in front of the device for identification then leave. Recommended for gate entry with boom barrier.

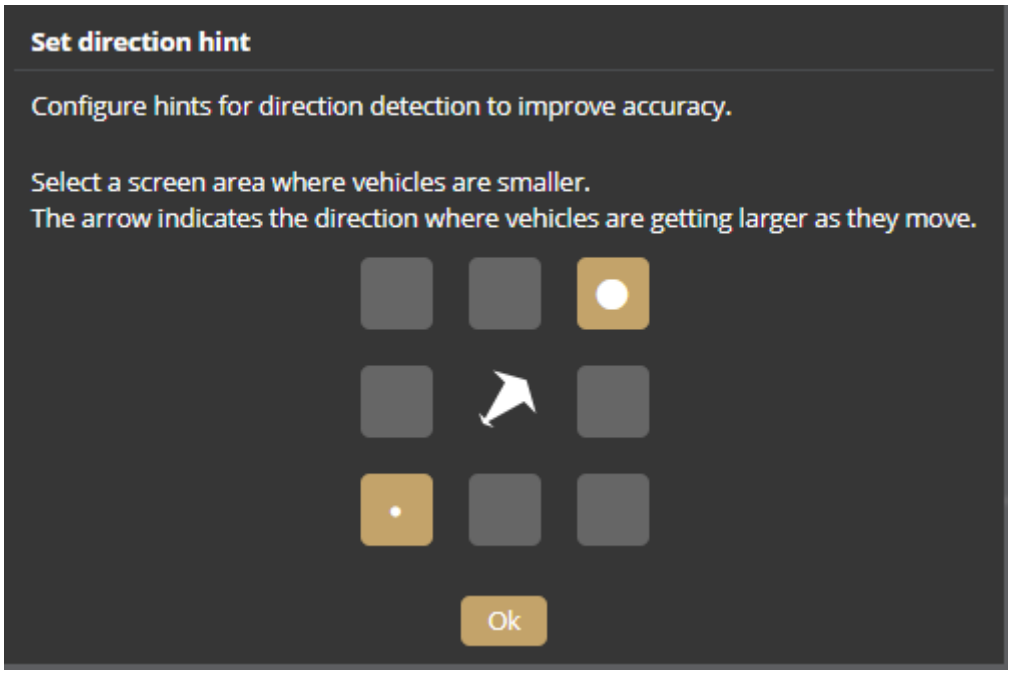
- **Slow traffic:** Vehicle traffic in urban areas.

- **Color recognition:** When color recognition is active, the device will attempt to identify the plate and/or vehicle colors.
- **Make & Model recognition:** When make and model recognition is active, the device will attempt to identify the make, model and color of the vehicle.
- **Detect direction:** Attempt to detect direction of the vehicle movement by license plate.

Important!

Direction detection is only performed if the same license plate number has been read at least twice by the camera and the height of the license plate number character has changed by at least 2 pixels in the image.

- **Direction hint:** Click on the edit button. Configure hints for direction detection to improve accuracy. Select a screen area where vehicles are smaller. The arrow indicates the direction where vehicles are getting larger as they move.

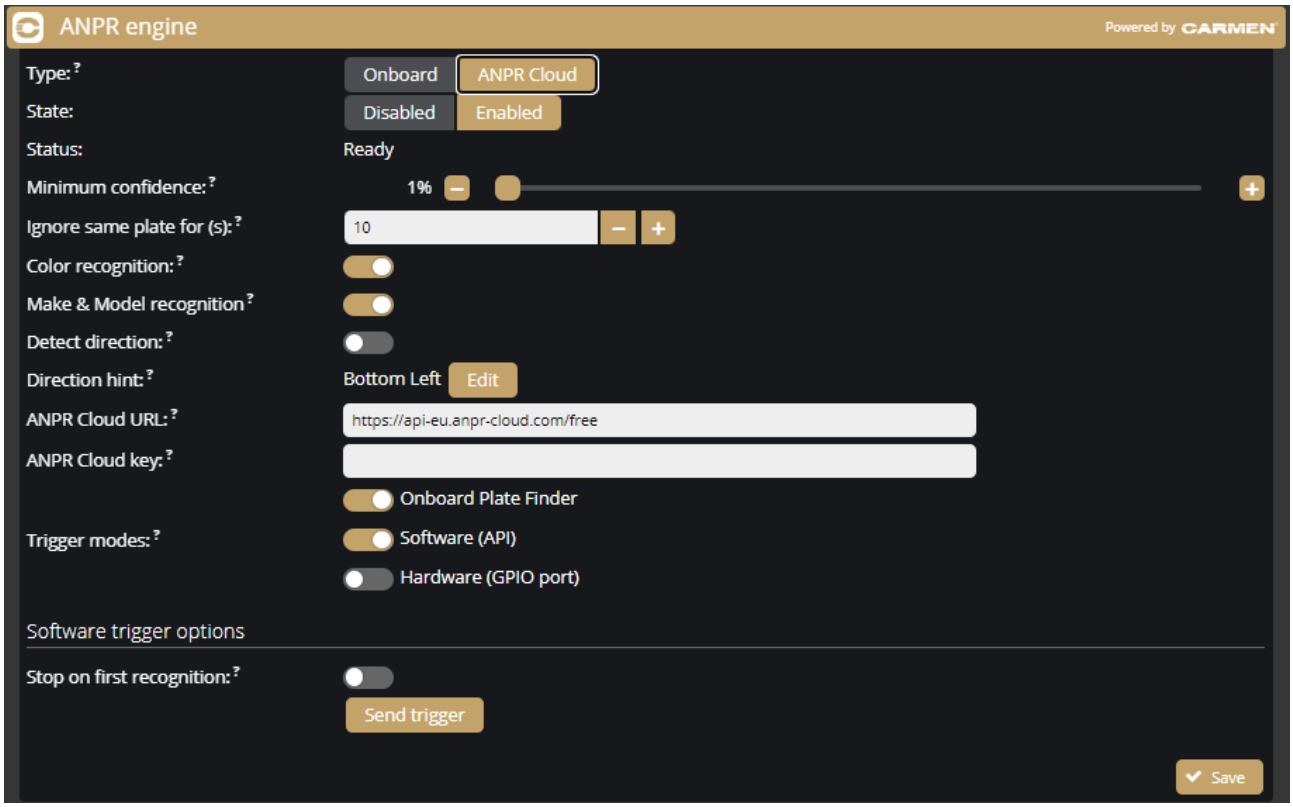


In the case of **Carmen Cloud**, the license plate recognition is not processed in the camera. Instead, images with license plates are selected in the camera and sent to the engine in the cloud for license plate recognition. A stable internet connection and a Carmen Cloud subscription are required for the license plate recognition to work in the cloud. For more information about Carmen Cloud, visit https://adaptiverecognition.com/anpr-cloud/.

You need to adjust the following parameters. The parameters that can be set vary depending on the engine:

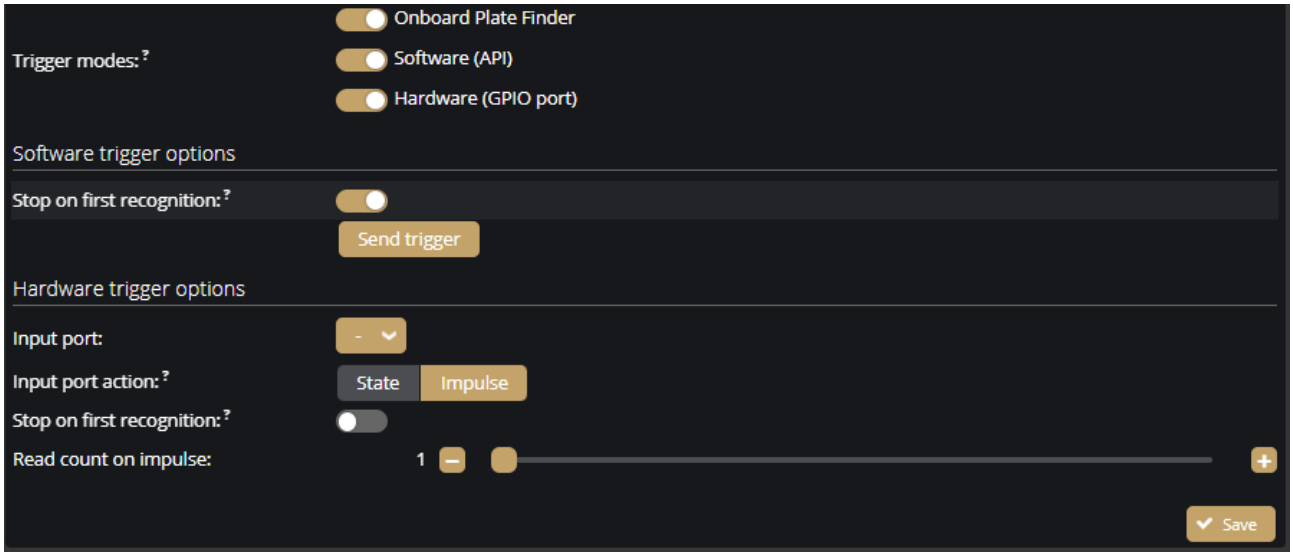
- **State:** You can enable or disable the ANPR engine to send license plate images to ANPR Cloud.
- **Minimum confidence:** Use the slider to define what reading confidence percentage should trigger an event. A low value potentially results in more reading errors. A high value potentially leads to missed reading cases. Therefore, values between **50-70%** are recommended. Check the **ANPR engine status** OSD layer to see the detected license plate numbers and their confidence values and set the desired minimum confidence level based on the results. See also ***{Overlay}***.
- **Ignore same plate for (s):** A time limit can be set for previously-read license plates. This value determines a waiting time in seconds before the same license plate is read again.
- **Color recognition:** When color recognition is active, the device will attempt to identify the plate and/or vehicle colors.
- **Make & Model recognition:** When make and model recognition is active, the device will attempt to identify the make, model and color of the vehicle.
- **Detect direction:** Attempt to detect direction of the vehicle movement by license plate. Enabling this option license plates will only be recognized after 2 successful recognitions.
- **Detect direction:** Attempt to detect direction of the vehicle movement by license plate.
- **Direction hint:** Click on the edit butto ) and set the road type to improve direction detection. (It becomes visible after the Detect direction option is activated.)
- **ANPR Cloud URL:** Paste the URL you received when subscribing to the field.
- **ANPR Cloud key:** Enter the individual key of the subscription.

The **Status** field displays information about the operation of Carmen Cloud, e.g., successful/failed connection, upload status, expired credits, etc.



Trigger modes

By default, the engine uses the on-board license plate finder to search for possible license plate locations before trying to detect license plates. This behaviour can be changed to use external triggers by configuring the **Trigger Mode** option. These modes can be used alternatively.



Available trigger modes are:

- **On-board Plate Finder:** Engine is triggered automatically by the on-board license plate finder
- **Software (API):** Engine can only be triggered using an API call (Analytics/TriggerEngine). See also the API documentation ***{System / Status}***
- **Hardware (Input port):** Engine is triggered by a configured GPIO input port. See also ***{System / IO}***

  Input port action: If **State** is selected the camera continuously processes images and searches license plates while the input port is active.

  Read count on impulse: When using hardware trigger in Impulse mode, the engine reads license plates until the specified count is reached.

  Interrupt on recognition: When this option is selected, the camera aborts further readings after recognizing successfully the first license plate.

7.16.4. ANPR DETECTOR

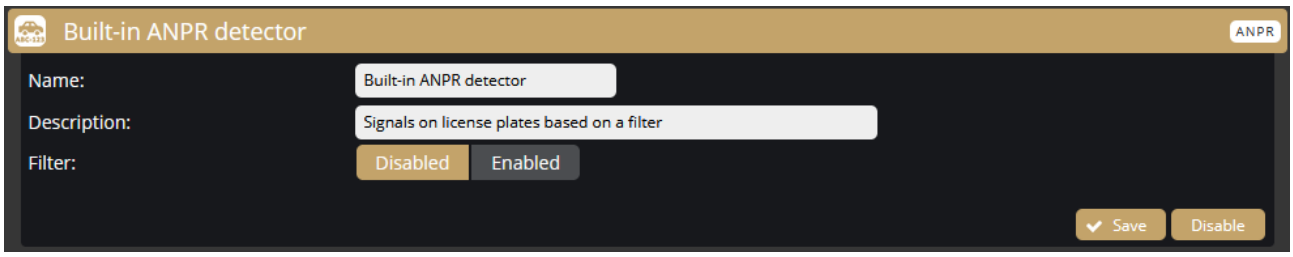
ANPR detector(s) is responsible for creating events from the results of the ANPR Engine.
A factory-set and non-erasable detector, the Built-in ANPR detector, is always present in the camera and generates the events of the license plate numbers detected by the ANPR Engine as soon as the camera is switched on.

Important!

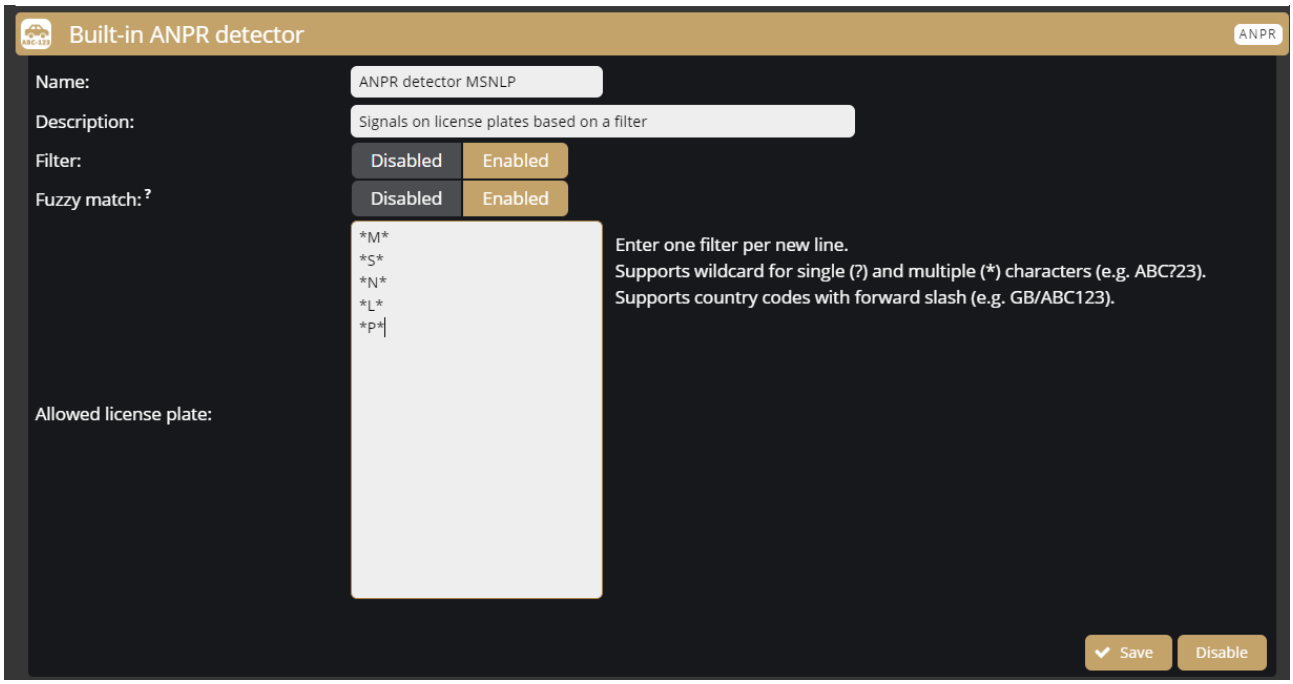
The ANPR detector must be defined separately. The primary functions of the license plate recognition cannot be set under the ANPR detector section, but on the interface of the ANPR engine.

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

- **Name:** The name of the detector can be entered.
- **Description:** To add a brief description to the detector.
- **Filter:** The system can be set only to read certain license plates.
- **Fuzzy match:** Allows matching to characters that look similar for example:
  - O ↔ 0 (Latin letter O and Latin digit zero)
  - ৭ ↔ 9 (Bengali digit seven and Latin digit nine)
  - Ү ↔ Y (Cyrillic letter Ue and Latin letter Y)



You can create additional ANPR detectors in addition to the Built-in ANPR detector. For example, by enabling filters and adding license plates to them, you can activate different detectors when certain license plates are detected, thus affecting the storage of the given group, the detector's manageability and also to control the output port. In the example below, the „ANPR detector MSNLP" is triggered if a license plate containing M, S, N, L or P characters are detected.



If you want to open a barrier when the above characters are detected, you have to enable the output port with the above detector in System / I/O menu at Trigger configuration section ***{System / IO}***.

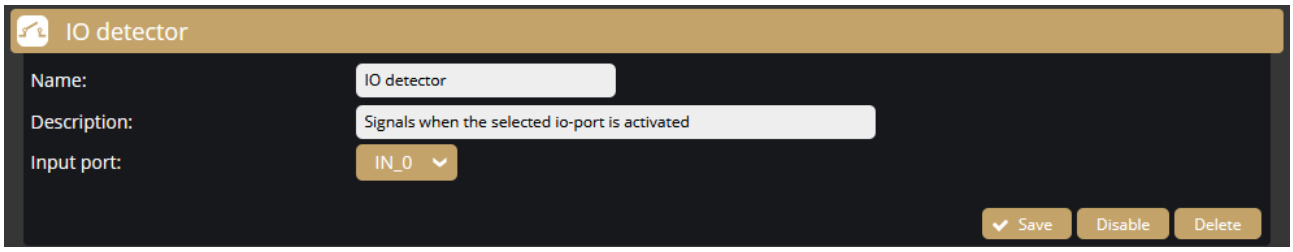
The individual masks defined in the different detectors can be used to determine where the recognised license plate appeared. In the example below, a separate detector generates the license plates detected in the left lane and the license plates detected in the middle lane. The ANPR Engine mask must have a common intersection with both detectors. This can be checked by activating the corresponding OSD layer(s).

7.16.5. IO DETECTOR

The IO detector detects a signal at the camera contact input and, thus, an event is generated when the input is activated. Note that no license plate recognition is performed when the input is activated.

You can adjust the following on the **IO detector** interface:

- **Name:** To enter the name of the detector.
- **Description:** A brief description can be added to the detector.
- **Input port:** The input receiving the signal must be selected. Currently, "IN_0" is the only available input on the camera.

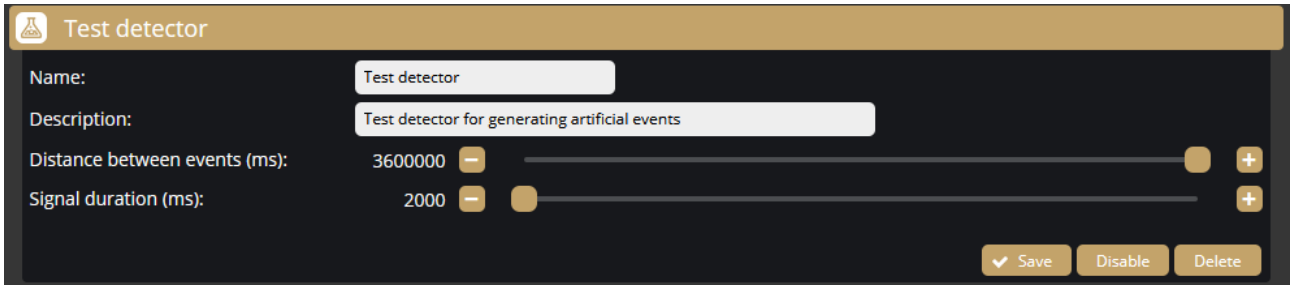


7.16.6. TEST DETECTOR

The Test detector can be used to test the camera'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 camera produces a test event at the set intervals.
- **Signal duration:** To adjust the signal length.



7.17. ANALYTICS / UPLOADERS

Setting the GDS upload

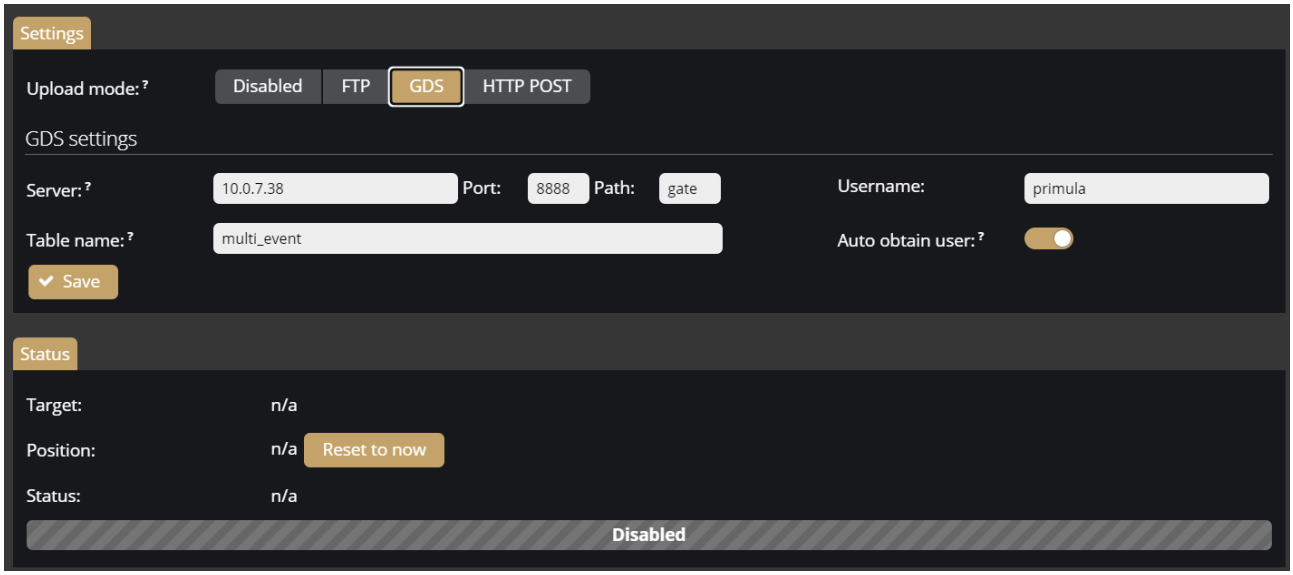
Enable the upload to the GDS (Globessey Data Server) at **Upload mode**, then enter the required data to set the GDS server. The microSD card is required for GDS upload to work. The uploader only sends data from previously stored content.

The following fields should be defined:

- **Server:** Address (IPv4) of the GDS server
- **Port:** Access port of the GDS server
- **Path:** Access path within the GDS 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.

With the **Reset to now** button you can reset the uploader progress to the current date. Events 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.

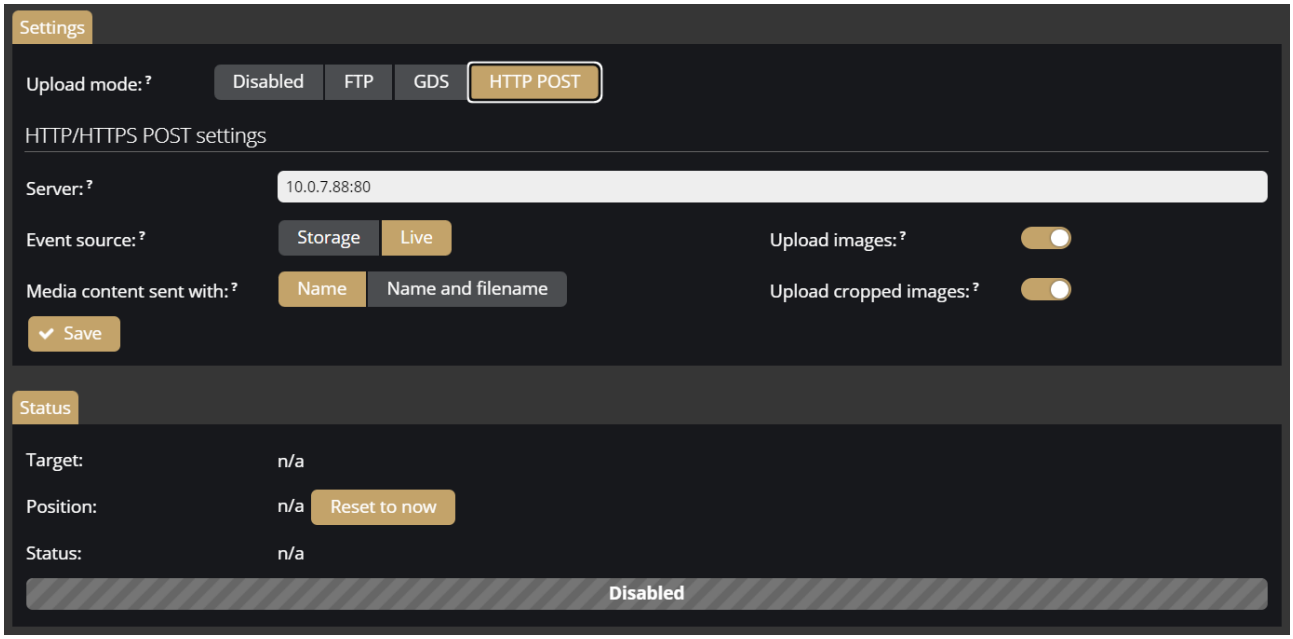


Setting the HTTP(S) upload

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

With the **Reset to now** button you can reset the uploader progress to the current date. Events 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.



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

Important!

Media attachment may be uploaded with only a Name or both a Name and filename files. The correct option depends on the capabilities of the used web service.

Event source:

Selecting the proper event source depend on the usecase.

Storage requires a storage device with configured on-board storage and uploads are delayed to some extent. This type can continue operation after a server or even device outage.

Live operates without any additional requirement and uploads are sent as soon as possible but in case of an outage or longer network failure events will be lost.

Note

When the device uploads an event packet it will only consider the upload a success if the server responds with HTTP status code 2xx. Under any other circumstances the upload is a failure and the device will retry at most 3 times.

If the server responds with HTTP status code 503 or 504 the device will retry indefinitely.

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

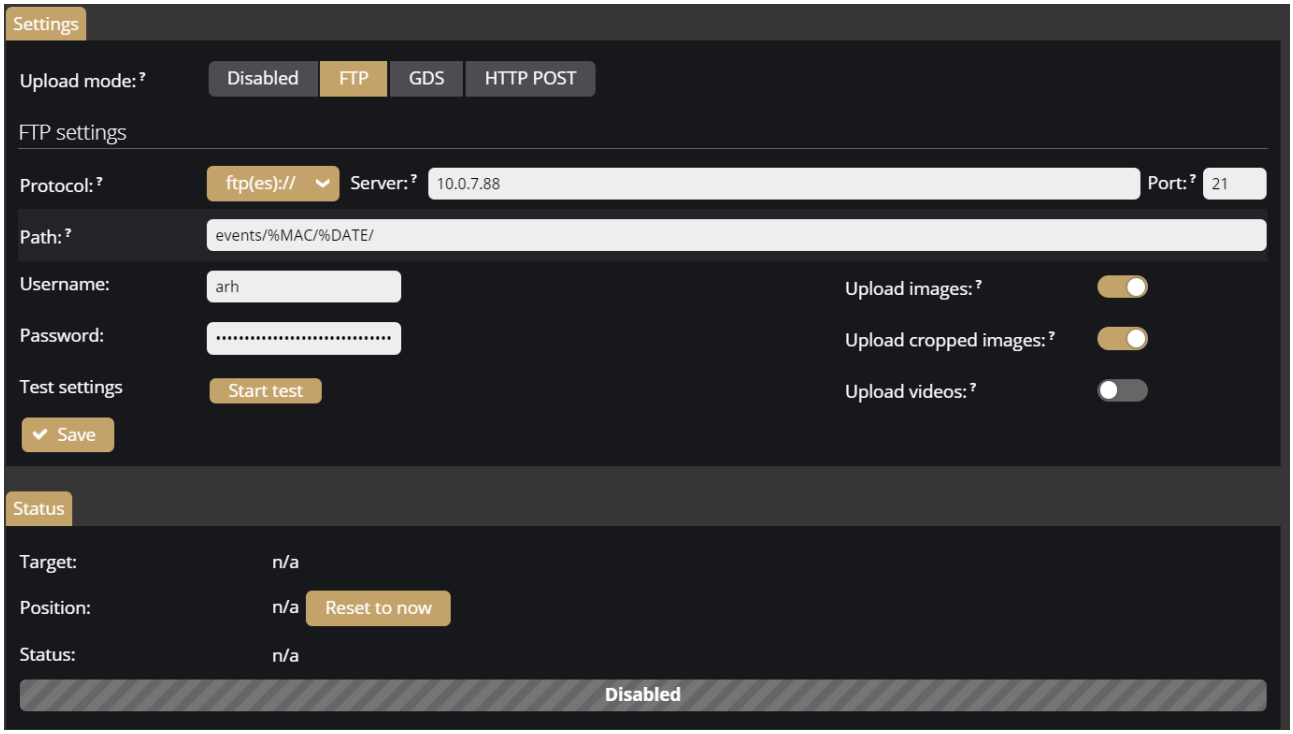
Setting the FTP upload

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

The following fields should be defined:

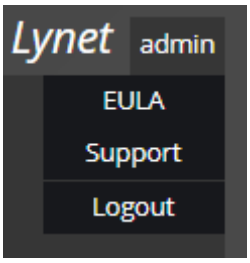
- **Protocol:** the services that are supported by the uploader (FTP(ES), FTPS, SFTP) can be selected
- **Server:** IP address (IPv4) or hostname of the FTP server
- **Port:** the service's port where it listens to requests
- **Path:** Path on the server, where the events will be stored. This can be configured with "%MAC" and "%DATE" variables, that will be exchanged to real values. Example: "events/%MAC/%DATE/" --> "events/001234/2022-01-01/".
- **Username/password:** Username and password required for the identification

With the **[Start test]** button you can test the connection between the camera and the FTP server.
With the **Reset to now** button you can reset the uploader progress to the current date. Events 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.



8. SUPPORT

The Support page can be accessed by clicking on the **Support button** above the Logout button.



Remote assistance

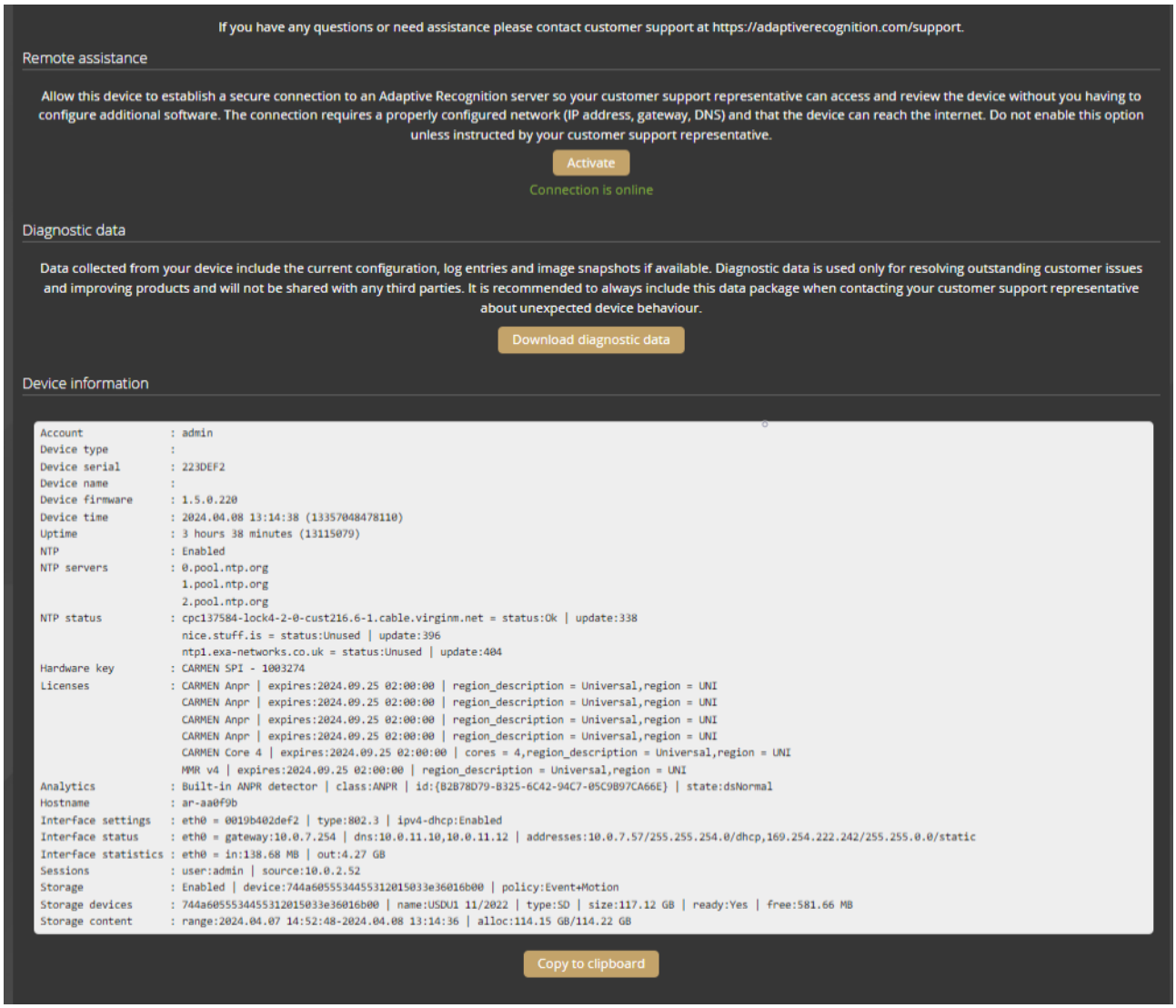
Allow this device to establish a secure connection to an Adaptive Recognition server so your customer support representative can access and review the device without you having to configure additional software. The connection requires a properly configured network (IP address, gateway, DNS) and that the device can reach the internet. Do not enable this option unless instructed by your customer support representative.

Please also open port 51820 UDP on your network to enable the connection.

The **Download diagnostic data** button allows you to download the diagnostic data (log file).

Diagnostic data: Data collected from your device include the current configuration, log entries and image snapshots if available. Diagnostic data is used only for resolving outstanding customer issues and improving products and will not be shared with any third parties. It is highly recommended to always include this data package when contacting your customer support representative about unexpected device behaviour.

Click the **Copy to clipboard** button to easily copy the most important data of the device.



Important!

If you announce a device-related issue in our support system, please copy and paste the above device information into the ticket description and attach the log file.

9. HOW TO USE LYNET CAMERA

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

9.1. CAMERA INSTALLATION

1. Mount the camera as follows (Please also refer to the Lynet Installation Guide:
    a. If you want to use the camera's input and output, mount the **GPIO cable kit** to the camera.
    b. For communication with the camera, use at least **Cat5** or **Cat6** outdoor shielded cable.
    c. Use a **PoE+** rated PoE power adapter or a switch output capable of PoE+.
    d. **Mount/secure the camera firmly** in its intended location.
    e. The maximum angle of rotation and/or tilt of the camera should not exceed **25–30 degrees**.
    f. The optical axis of the camera should **face the license plate** of the oncoming or departing vehicle. The camera's optical axis shall be as small an angle as possible with the axis of travel of passing vehicles.
    g. The camera should be **at least one meter above the headlights** of vehicles. If possible, it should not be accessible without an assistive device. If the camera is mounted completely unprotected outdoors, it is recommended to mount it higher to minimize the risk of water falling on the front of the camera when raining.
    h. In the image, **the license plate should be horizontal**. In extreme cases, it may be necessary to turn the mounting bracket.
    i. Adjust the camera lens angle of view so that the characters on the license plate are **at least 25 pixels high**. Consider whether you want to recognize license plates with small character sizes. For example, the characters on Italian license plates are smaller than those of countries geographically further north. The easiest way to check the character size of a license plate is to display the "ANPR Status" OSD layer in Live View and look at the "**Avg. char height:**" to see the character size ***{Overlay}***.
    j. Use **real, moving vehicle(s)** to test the settings; it's not enough to just show a plain license plate in front of the camera.
    k. Adjust the camera's direction and angle of view to make the license plate visible for **at least one, but preferably two seconds** with the right character size.

2. **Find the camera on the network**, then access the camera's web interface
    a. From the MAC address of the camera, you can calculate the **IP address that is always available**. It is always in the format **169.254.aa.bb**, where the number aa/bb is the decimal value of the last two pairs of numbers in the MAC address. However, if a DHCP server is available on the network, the camera will also get an IP address from the DHCP server. You can access the camera from both IP addresses.
    b. Optionally, you can use the **AR Device Tool** to locate the camera on your local network. ***{AR Device Tool}***.
    c. You can access the camera's web interface 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 **ANPR Engine and License Key** to your camera ***{System / Device}*** according to the email received when you purchased the camera. Check the ANPR Engine quarterly to ensure that the camera is always running with the latest version of the engine. 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 („License key: CARMEN SPI – xxxxxx") and your licence corresponds to the desired region (push the „ANPR License button" and the popup window shows the region and the validity time of the license).
    b. Check that the **camera has the latest firmware**. You can download the latest FW from ATSS. Upload it in ***{System / Device}*** menu.
    c. **Check/set the camera time *{System / Device}***. Use at least a local NTP server.
    d. If you haven't already done so, adjust the camera **angle of view** and take into consideration the character sizes as well ***{see point (1) in this chapter}***.
    e. With the above steps and the default camera settings, you will get **license plate reading results**. These events will be listed in the Live interface below the live image.
    f. After changing any settings (eg.: the field of view, position, direction of the camera, resolution, etc.) **restart** the camera to reset internal statistics.

4. Fine-tuning

   Besides checking everything in the Settings menu, there are a few settings that, in addition to the above, are particularly recommended to check/fine-tune to achieve above-average license plate recognition accuracy.

   a. Set the **Image settings parameters** to as perfect as possible for ANPR. Take into account the location you are observing and, most importantly, make sure the number plates are legible. Check the good visibility of the license plates for a few days after the camera installation, during the day, night and sunrise/sunset times of the day. Repeat the process later to adjust for the different seasonal changes.

   b. Check the **resolution/FOV of the camera**. The pixel size of the license plate characters' has to be in the mentioned range (**min. 25 pixels**). You may need to **increase the bandwidth** of the video stream if the characters cannot be seen well ***{Media / Video}***.

   c. Check the settings of the **ANPR engine *{ANPR engine}***. Lowering the minimum confidence parameter will increase the chance of misreading license plates but will also reduce the probability of missing an unrecognized plate.

   d. Set the **country** where the camera is installed so that in the case of a type/country code engine, you increase the priority of the license plate numbers of that country ***{ANPR engine}***.

   e. Draw the area on the picture where you expect to see number plates using a mask. This will increase the possibility of speeding up the number plate recognition, and cars parked/passing by in an undesirable place will not trigger an event. ***{ANPR engine}***.

   f. During the setup, testing and checking of the operation, it is recommended to display the necessary OSD layer(s) to get technical feedback about the camera's internal operation, changes of image parameters, etc. We suggest displaying the **General/ANPR Engine Status** and **General/Image properties** layers during camera setup ***{Overlay}***. Also, display these captions during playback and reduce the playback speed to facilitate more accurate evaluation.

9.2. USING LYNET CAMERA

Below you will find some possibilities for creating access control with Lynet and some other options for using the camera. A suitable combination of the following can also be implemented.

9.2.1. STANDALONE OPERATION WITH ONBOARD ANPR

The camera itself manages an access point. It activates its output when it recognizes the license plates registered in the camera to give an opening command, e.g., to a barrier. Lynet stores events, which can be retrieved, downloaded and exported via the camera's own web interface. In addition to local storage on an SD card, it is also possible to configure an optional http, FTP or GDS server for event upload.

The necessary devices and the most important, specific settings are as follows:

- GPIO cable kit
- Carmen ANPR Engine and License in Lynet camera
- ANPR detector with filter enabled. (Registering license plate numbers in "Allowed license plate" field)
- A properly configured HTTP, FTP or GDS server (if required).

9.2.2. STANDALONE OPERATION WITH CARMEN CLOUD

The camera itself manages an access point. It activates its output when it recognizes the number plates registered in the camera to give an opening command, e.g., to a barrier. Lynet stores events, which can be retrieved, downloaded and exported via the camera's own web interface. In addition to local storage on an SD card, it is also possible to configure an optional HTTP, FTP or GDS server for event upload. The number plates are detected in the camera, but the number plates are recognized in Carmen Cloud.

The necessary devices and the most important, specific settings are as follows:

- Stable internet connection
- Lynet camera without ANPR Engine and License
- Carmen Cloud subscription
- ANPR detector with filter enabled. (Registering license plate numbers in "Allowed license plate" field)
- A properly configured HTTP, FTP or GDS server (if required).

9.2.3. INTEGRATED OPERATION WITH 3RD PARTY SOFTWARE

By mounting the camera at an access point, the camera uses its own API to allow the system integrator to connect the camera to the system integrator's own software. If required, the I/O can be used on the camera, or the system integrator can use its own software IO hardware.
The following data is available:

- live video stream,
- still image,
- query the input port status
- output port control
- current event data
- stored events data
- event video, event image, license plate coordinate.

The necessary devices and the most important, specific settings are as follows:

- Stable internet connection (if necessary)
- GPIO cable kit (if necessary)
- Onboard ANPR or Carmen Cloud subscription
- A properly configured HTTP, FTP or GDS server (if required).

More useful information on integration can be found on the following website:
https://github.com/adaptiverecognition/harbard-sdk

API DOCUMENTATION

10. GETTING STARTED

10.1. INTRODUCTION

This document is the API specification of the Lynet devices.

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 futurewithout notice

10.1.1. 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 http://en.wikipedia.org/wiki/HTTP_body_data)
EXCEPTION	A response given by the device when an error occured

10.2. AUTHENTICATION

Accessing resources on the device requires an authenticated session.

10.2.1. 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.

10.2.2. 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.

10.2.3. 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

{}
```

10.2.4. 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 isprovided for easy access while experimenting with APIs

10.3. EXECUTING COMMANDS

10.3.1. 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 theCOOKIE or GET variable named sid

10.3.2. INPUT/OUPUT 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

Adaptive Recognition **America** Adaptive Recognition **Nordic** REQUESTINFO@ADAPTIVERECOGNITION.COM
Adaptive Recognition **Hungary** Adaptive Recognition **Singapore** WWW.ADAPTIVERECOGNITION.COM

10.3.3. 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."
        }
}
```

10.4. DATA TYPES

The JSON format allows transfer of serveral 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 bitnumbers. 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 ahexadecimal 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.

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

9.5 COMMAND OPTIONS

Certain structures' parameters are limited to numeric ranges or a list of possible values. These possible valuesare 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 **OptionNumericRange** 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	Funtionality	Module descriptor
Analytics	Detectors and events	**ModuleAnalytics**
IO	External I/O ports	**ModuleIO**
Media	Audio and video streams	**ModuleMedia**

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

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 detectorsto 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 **Analytics/GetAnprEngine**)
iTracking engine	Marks and tracks moving objects (see **Analytics/GetTracker**)
Motion engine	Performs motion detection on the whole image

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

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 thedevice configuration the following detectors may be available:

Detector	Reference
AlarmInput port monitor DetectorIO	DetectorConfigurationIO EventIO
AlarmDetectorTest Detector for API testing	DetectorConfigurationTest EventTest
For ANPR devices only:	
AlarmDetectorANPR License plate detection	DetectorConfigurationANPR EventANPR
For Enforcement devices only:	
AlarmDetectorEmergencyLane Emergency lane violation	DetectorConfigurationEmergencyLane EventEmergencyLane
AlarmDetectorForbiddenZone Forbidden zone violation	DetectorConfigurationForbiddenZone EventForbiddenZone
AlarmDetectorLane Lane movement	DetectorConfigurationLane EventLane
AlarmDetectorRedStop Traffic light violation	DetectorConfigurationRedStop EventRedStop
AlarmDetectorStoppedObject Prohibited stop detection	DetectorConfigurationStoppedObject EventStoppedObject
AlarmDetectorStopViolation Stop sign violation	DetectorConfigurationStopViolation EventStopViolation
AlarmDetectorTrafficLine General line crossing	DetectorConfigurationTrafficLine EventTrafficLine
AlarmDetectorUTurn Illegal U-turn detection	DetectorConfigurationUTurn EventUTurn
AlarmDetectorWhiteLineViolation White line violation	DetectorConfigurationWhiteLineViolation EventWhiteLineViolation
AlarmDetectorWrongTurn Illegal turn violation	DetectorConfigurationWrongTurn EventWrongTurn
AlarmDetectorWrongWay Wrong-way driving detection	DetectorConfigurationWrongWay EventWrongWay

10.2. GEOMETRY

Some detectors and engines require some form of 2D configuration where polygons and lines define how theimages 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	GeometryLineSegment	Straight line with two points defining the start and endof the line
Segmented line	GeometryLine	Segmented line with at least one segment, each consisting of a start and end point
Ordered segmented lines	GeometryLineGroups	Groups of segmented lines where an order of groups isformed using indicies
Rectangle	GeometryRectangle	Rectangle where each side is parallel to the x or y axisof the image
Polygons	GeometryPolygons	List of polygons. A polygon has at least 3 points and anarbitary 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 Device buffers events	Event loss on slow connection No image or video content

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

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

Stored event query is a similiar 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 Advanced filtering	Requires storage device 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.

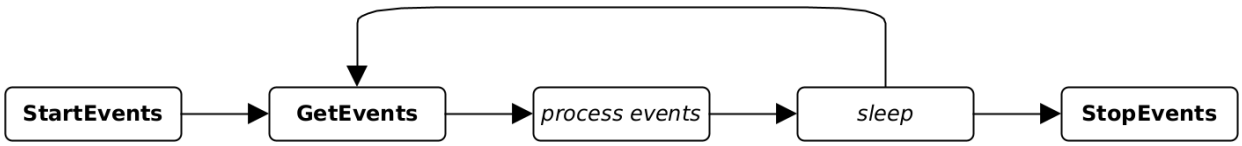
Pros	Cons
Event image and video available Event region of interest image available Compatible with most HTTP server implementation	Requires storage device Significant latency

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:

X-Event-Index	Index incrementing by one for each event. A gap in the indicies means the device wasunable to send a packet probably due to slow connection and buffer limitations and dropped the event
X-Timestamp	Posix UTC timestamp of the event in milliseconds

Images are sent with the multipart Content-Type of **image/jpeg** or text/plain depending on the **image** GET parameter. Additional headers include:

X-Image-Index	Index incrementing by one for each image. A gap in the indicies means the device wasunable to send a packet probably due to slow connection and buffer limitations and dropped the image
X-Frame-Id	ID of sensor frame from which this JPEG was encoded
X-Frame-Timestamp	Monotonic timestamp of the image in milliseconds that is independent of the wallclock and is not affected by clock changes
X-Frame-Width	Image width
X-Frame-Height	Image height
X-Timestamp	Posix UTC timestamp of the image in milliseconds
X-Keep-Alive	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

How images are inserted into the stream can be changed with the image GET parameter.

- image=0: images are disabled
- image=1: (default) images are sent as binary stream with **image/jpeg** Content-Type
- image=2: images are sent as base64 encoded text stream with **text/plain** Content-Type

Using the URL below the client will receive base64 encoded images with the events:

```
http://DEVICE_IP/live/events?image=2
```

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 networkequipment 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 transferwas 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:
noneX-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 theevent.

11.5.3. REQUEST FORMAT

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

Field name	MIME type	Count	Description
event_timestamp	text/plain	1	Field contains the posix UTC timestamp of the event inmilliseconds
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 **Event**)

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 aswell:

```
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/1.0
Accept: */*
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
```


Adaptive Recognition **America** Adaptive Recognition **Nordic**
Adaptive Recognition **Hungary** Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

```
{
        "DetectorVersion" : 131072,
        "DetectorID" : "{7D0829EA-E8FD-7546-92C7-3528E6216CBB}",
        "DetectorClassID" : 1968398405,
        "DetectorClass" : "AlarmDetectorANPR",
        "EventTime" : "13276206506436",
        "State" : "dsNormal",
        "EventCode" : 114,
        "EventInfo" : {
                        "Text" : "ABC123",
                        "Confidence" : 0.81999999284744262695,
                        "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
```


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

12.1.3. KEEPALIVE

During quiet periods the device may not transmit any data for a significant amount of time. Many networkequipment 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 transferwas 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",
}

--IPCamIOStreamBoundary
```

13.FUNCTIONS

13.1 CATEGORY:ANALYTICS

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

Methods

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

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

13.1.1 ANALYTICS/CREATEDETECTOR

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

Specification

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

13.1.2 ANALYTICS/DELETEALLDETECTORS

Deletes all detector instances except built-in detectors

Specification

User level	ADMINISTRATOR
Request data	*none*
Response data	*none*
Exceptions	*none*

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	DetectorRequest
Response data	*none*
Exceptions	**DetectorNotFoundException**: The specified detector does not exist. **AccessDeniedException**: 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	**DetectorRequest**
Response data	*none*
Exceptions	**DetectorNotFoundException**: 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	DetectorRequest
Response data	*none*
Exceptions	**DetectorNotFoundException**: 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	*none*
Response data	AnprEngineConfiguration
Exceptions	*none*

Adaptive Recognition **America** Adaptive Recognition **Nordic**
Adaptive Recognition **Hungary** Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

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	*none*
Response data	**AnprEngineConfiguration**
Exceptions	*none*

13.1.8 ANALYTICS/GETANPRENGINESTATE

Get the current state of the ANPR engine

Specification

User level	ADMINISTRATOR
Request data	none
Response data	**AnprEngineState**
Exceptions	none

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

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	**DetectorRequest**
Response data	**Detector**
Exceptions	**DetectorNotFoundException**: 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	DetectorClassRequest
Response data	Detector
Exceptions	*none*

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	DetectorRequest
Response data	DetectorState
Exceptions	**DetectorNotFoundException**: The specified detector does not exist.

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

13.1.12 ANALYTICS/GETDETECTORS

Get the active detector instances on this device

Specification

User level	USER
Request data	*none*
Response data	**DetectorList**
Exceptions	*none*

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	*none*
Response data	**BufferedEvents**
Exceptions	**StreamNotStartedException**: 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	*none*
Response data	**SupportedDetectors**
Exceptions	*none*

13.1.15 ANALYTICS/GETTRACKER

Get the current configuration of the tracker

See also: **Analytics/GetTrackerDefaults**, **Analytics/SetTracker**

Specification

User level	ADMINISTRATOR
Request data	*none*
Response data	**TrackerConfiguration**
Exceptions	*none*

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	*none*
Response data	**TrackerConfiguration**
Exceptions	*none*

13.1.17 ANALYTICS/SETANPRENGINE

Change the configuration of the ANPR engine

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

Specification

User level	ADMINISTRATOR
Request data	**AnprEngineConfiguration**
Response data	*none*
Exceptions	*none*

Adaptive Recognition **America** Adaptive Recognition **Nordic** REQUESTINFO@ADAPTIVERECOGNITION.COM
Adaptive Recognition **Hungary** Adaptive Recognition **Singapore** WWW.ADAPTIVERECOGNITION.COM

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	**Detector**
Response data	*none*
Exceptions	**DetectorNotFoundException**: 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	**TrackerConfiguration**
Response data	*none*
Exceptions	*none*

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

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	**BufferedEventsRequest**
Response data	*none*
Exceptions	*none*

13.1.21 ANALYTICS/STOPEVENTS

Stop the event buffering for the calling session

Specification

User level	USER
Request data	*none*
Response data	*none*
Exceptions	*none*

13.1.22 ANALYTICS/TRIGGERENGINE

Manually trigger an analytics engine

Specification

User level	USER
Request data	**AnalyticsEngineTrigger**
Response data	**AnalyticsEngineTriggerResponse**
Exceptions	**InvalidTriggerException:** 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
Storage/GetEvents	Perform a query on the stored events
Storage/GetStatistics	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	**StorageEventsRequest**
Response data	**StorageEvents**
Exceptions	**EventsNotFoundException**: 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	*none*
Response data	**StorageStatistics**
Exceptions	*none*

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 recommneded to use the **System/ GetDevice** method to get general information about it.

Methods

Method	Description
System/ClearSecurityHistory	Release the block on all clients that are currently banned
System/FactoryReset	Factory reset the settings and reboot
System/GetApiVersion	Get the version of the JSON API
System/GetDevice	Get general information about the device
System/DetLocationSettings	Get the location settings of the device
System/GetSecurityHistory	List the active session and blocked clients
System/GetSecuritySettings	Get the security settings
System/Reboot	Start the reboot of the device
System/RunTest	Testing method for checking JSON API
System/SetDevice	Change the name and description of the device
System/SetLocationSettings	Set the location settings of the device
System/SetSecuritySettings	Change the security settings
Date & time	
System/GetNtpSettings	Get the NTP settings
System/GetTime	Get the current timestamp
System/GetTimezone	Get the current timezone
System/GetTimezones	List the timezones that are available on the device
System/SetNtpSettings	Change the NTP settings
System/SetTime	Change the current timestamp
System/SetTimezone	Change the current timezone
I/O	
System/GetGpioSettings	Get the available digital inputs and outputs on this device
System/GetGpioStates	Get the last known state of available digital inputs and

	outputs on thisdevice
System/SetGpioInputSettings	Change the configuration of a digital input port
System/SetGpioOutput	Change the state of a digital output port
System/SetGpioOutputSettings	Change the configuration of a digital output port
System/TriggerGpioOutput	Send an impulse to a digital output port
Users	
System/AddUser	Add a new user account
System/DeleteUser	Remove a user account
System/GetCurrentUser	Get the user of the current session
System/GetUsers	List all users accounts on the device. The password field is present but willnot contain any information.
System/ModifyUser	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	**User**
Response data	*none*
Exceptions	**UserValueException**: An invalid parameter was sent **UserExistsException**: 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	*none*
Response data	*none*
Exceptions	*none*

13.3.3 SYSTEM/DELETEUSER

Remove a user account

See also: System/AddUser, System/GetCurrentUser, System/ModifyUser

Specification

User level	ADMINISTRATOR
Request data	**UserId**
Response data	*none*
Exceptions	**DeleteSelfException**: A user cannot remove its own account **UserNotExistsException**: 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	*none*
Response data	*none*
Exceptions	*none*

13.3.5 SYSTEM/GETAPIVERSION

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	*none*
Response data	**ApiVersion**
Exceptions	*none*

13.3.6 SYSTEM/GETCURRENTUSER

Get the user of the current session

> **See also:** **System/AddUser**, **System/DeleteUser**, **System/ModifyUser**

Specification

User level	USER
Request data	*none*
Response data	**UserInfo**
Exceptions	*none*

13.3.7 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

13.3.8 SYSTEM/GETGPIOSETTINGS

Get the available digital inputs and outputs on this device

Specification

User level	USER
Request data	*none*
Response data	GpioSettings
Exceptions	*none*

13.3.9 SYSTEM/GETGPIOSTATES

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

Specification

User level	USER
Request data	*none*
Response data	**GpioStates**
Exceptions	*none*

13.3.10 SYSTEM/GETLOCATIONSETTINGS

Query how the device obtains its locational data.

> **See also:** System/SetLocationSettings

Specification

User level	USER
Request data	*none*
Response data	**LocationSettings**
Exceptions	*none*

13.3.11 SYSTEM/GETNTPSETTINGS

Get the NTP settings

> **See also:** System/SetNtpSettings

Specification

User level	USER
Request data	*none*
Response data	**NtpSettings**
Exceptions	*none*

13.3.12 SYSTEM/GETSECURITYHISTORY

List the active session and blocked clients

Specification

User level	USER
Request data	*none*
Response data	**SecurityHistory**
Exceptions	*none*

13.3.13 SYSTEM/GETSECURITYSETTINGS

Get the security settings of the device tha controls allowed authentication attemps 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 attemps - regardless of the used credentials - will be ignored until the block expires.

See also: System/SetSecuritySettings

Specification

User level	USER
Request data	*none*
Response data	**SecuritySettings**
Exceptions	*none*

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

13.3.14 SYSTEM/GETTIME

Get the current timestamp

> **See also:** System/SetTime

Specification

User level	USER
Request data	*none*
Response data	**TimeSettings**
Exceptions	*none*

13.3.15 SYSTEM/GETTIMEZONE

Get the current timezone

> **See also:** System/SetTimezone

Specification

User level	USER
Request data	*none*
Response data	**TimezoneSettings**
Exceptions	*none*

13.3.16 SYSTEM/GETTIMEZONES

List the timezones that are available on the device

Specification

User level	USER
Request data	*none*
Response data	**TimezoneList**
Exceptions	*none*

13.3.17 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	*none*
Response data	**Users**
Exceptions	*none*

13.3.18 SYSTEM/MODIFYUSER

Modify the properties of a user account

> **see also:** **System/AddUser**, **System/DeleteUser**, **System/GetCurrentUser**

Specification

User level	ADMINISTRATOR
Request data	**User**
Response data	*none*
Exceptions	**UserValueException**: An invalid parameter was sent **ModifySelfException**: A user cannot modify its own role **UserNotExistsException**: Tried to modify a non-existing user account

13.3.19 SYSTEM/REBOOT

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

Specification

User level	ADMINISTRATOR
Request data	**RebootSettings**
Response data	*none*
Exceptions	*none*

13.3.20 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	**TestInput**
Response data	**TestOutput**
Exceptions	**TestException**: This is an exception thrown when the **ThrowException** of the input is setto true

13.3.21 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	*none*
Response data	**SystemSettings**
Exceptions	*none*

13.3.22 SYSTEM/SETGPIOINPUTSETTINGS

Change the configuration of a digital input port

See also: System/SetGpioOutput, System/SetGpioOutputSettings, System/TriggerGpioOutput

Specification

User level	ADMINISTRATOR
Request data	**GpioInputPort**
Response data	*none*
Exceptions	*none*

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

13.3.23 SYSTEM/SETGPIOOUTPUT

Change the state of a digital output port

> **See also:** System/SetGpioInputSettings, System/SetGpioOutputSettings, System/TriggerGpioOutput

Specification

User level	OPERATOR
Request data	**GpioOutputPortState**
Response data	*none*
Exceptions	*none*

13.3.24 SYSTEM/SETGPIOOUTPUTSETTINGS

Change the configuration of a digital output port

> **See also:** System/SetGpioInputSettings, System/SetGpioOutput, System/TriggerGpioOutput

Specification

User level	ADMINISTRATOR
Request data	**GpioOutputPort**
Response data	*none*
Exceptions	*none*

13.3.25 SYSTEM/SETLOCATIONSETTINGS

Change the way the device obtains its locational data.

See also: System/GetLocationSettings

Specification

User level	ADMINISTRATOR
Request data	*none*
Response data	**LocationSettings**
Exceptions	*none*

13.3.26 SYSTEM/SETNTPSETTINGS

Change the NTP settings

See also: System/GetNtpSettings

Specification

User level	ADMINISTRATOR
Request data	**NtpSettings**
Response data	*none*
Exceptions	*none*

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

13.3.27 SYSTEM/SETSECURITYSETTINGS

Change the security settings

See also: System/GetSecuritySettings

Specification

User level	ADMINISTRATOR
Request data	**SecuritySettings**
Response data	*none*
Exceptions	*none*

13.3.28 SYSTEM/SETTIME

Change the current timestamp

See also: System/GetTime

Specification

User level	ADMINISTRATOR
Request data	**TimeSettings**
Response data	*none*
Exceptions	*none*

13.3.29 SYSTEM/SETTIMEZONE

Change the current timezone

See also: System/GetTimezone

Specification

User level	ADMINISTRATOR
Request data	**TimezoneSettings**
Response data	*none*
Exceptions	*none*

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

13.3.30 SYSTEM/TRIGGERGPIOOUTPUT

Send an impulse to a digital output port

See also: System/SetGpioInputSettings, System/SetGpioOutput, System/SetGpioOutputSettings

Specification

User level	OPERATOR
Request data	**GpioPortId**
Response data	*none*
Exceptions	*none*

14. DATA STUCTURES

14.1. 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": "..."
}
```

14.2. 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	User defined string that will be attached to triggered events

Pseudo code

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

14.3 ANALYTICSENGINETRIGGERRESPONSE

Properties of a manual engine trigger.

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)
Timestamp	int32	Timestamp when the trigger was received by the device

Pseudo code

```
{
    "Count": ...,
    "Target": "...",
    "Timestamp":.....
}
```

14.4 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
- **Tracker:** Engine is triggered automatically by the on-board object tracker
- **Software:** Engine can be triggered using the **Analytics/TriggerEngine** call
- **Hardware:** Engine is triggered by a configured GPIO input port

The **TextFormat** option controls how the text of recognised license plates will be outputted into the event.

- **Type1:** No separator characters are included
- **Type2:** All separator characters are included as spaces
- **Type3:** All separator characters are included as is

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

- **Impulse:** Activation of the input port triggers the engine to make **ReadCount** number of successful reads
- **State:** The engine continouously 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	Prefered country code
Direction	Bool	Set to enable direction recognition on license plates
DirectionHint	string	Additional hint for direction detection
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
Mask	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
Speed	Bool	Set to enable speed estimation using license plate positions
SpeedHint		Advanced settings that improve speed estimatio
Height	Double	Vertical distance in meters between the device and the road
TextFormat	String	Outputted format of license plate text
TriggerMode	String	(deprecated) Source of triggers that activates the ANPR engine. This setting is overwritten if TriggerModes is specified aswell.
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": ...,
            "DirectionHint": "...",
            "HardwareTriggerSettings":
            {
                  "InterruptOnRecognition": ...,
                  "Port": "...",
                  "ReadCount": ...,
                  "TriggerMode": "..."
            },
            "InterruptOnRecognition": ...,
            "MMR": ...,
            "Masks":
            {
                   "0": [ ..., ..., ... ],
                   "1": [ ..., ..., ... ]
            },
            "RecognitionMode": "...",
            "SoftwareTriggerSettings":
            {
            "InterruptOnRecognition": ... },
            "Speed": ...,
            "SpeedHint":
            {
                    "Height": ...
            },
            "TextFormat": "...",
            "TriggerMode": "...",
            "TriggerModes":
            {
                    "0": "...",
                    "1": "..."
            },
            "Type": "...",
            "ValidInTimeWindow": ...
     }
}
```

14.5 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": "..."
    }
}
```

14.6 APIVERSION

JSON API information

> **See also:** System/GetVersion

Structure

Parameter	Type	Description
Version	int32	Current version of the JSON API

Pseudo code

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


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.7 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

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

Structure

Parameter	Type	Description
DiscardedEvents	int32	Number of events discarded since the start of buffering
EventList	Event	List of events
Config	DetectorConfiguration	Configuration of the detector when this event was created
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	[em]unused[/em]
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.
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
Device		
Description	string	User-specified description
Location		User-specified location
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees
Network		
Host	String	Hostname or IP address of the location transmitter
Port	Int32	Port of the location transmitter service on the server

Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties
Name	String	User-specified name
Serial	String	Unique device serial number
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 DetectorState)

Pseudo code

```
{
    "DiscardedEvents": ...,
    "EventList":
    {
        "Config":
        {
            "BuiltIn": ...,
            "Class": "...",
            "Description": "...",
            "DetectorClassID": ...,
            "DetectorID": "{...}",
            "DisplayName": "...",
            "Enabled": ...,
            "FpsLimit": ...,
            "RestoreDelayMs": ...,
            "Version": ...,
            "ViolationTimeMs": ...
        },
        "DetectorClassID": ...,
        "DetectorEventType": "...",
        "DetectorID": "{...}",
        "DetectorVersion": ...,
        "Device":
        {
            "Description": "...",
            "Location":
            {
                "Config":
                {
                  "Manual":
                  {
                    "Position":
                    {
                        "Latitude" :...,
                        "Longitude":...
                     }
                  },
                  "Network":
                  {
                      "Host": "...",
                      "Port": ...,
                      "Protocol": "..."
                   }
                },
                "Mode": "..."
            }
            "Name": "...",
            "Serial": "..."
```

```
            }
            "EventCode": ...,
            "EventID": "{...}",
            "EventTime": ...,
            "EventTriggerTime": ...,
            "State": "..."
        }
}
```

14.8 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": "{...}"
	}
}
```


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.9 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	DetectorConfiguration	Contains further configuration options specific to the detectortype
BuiltIn	bool	Automatically created detectors are marked built-in andcannot 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-facinginterfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set tozero for no limit.
RestoreDelayMs	int64	*unused*
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an eventis 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 datafrom the device.

Adaptive Recognition **America** Adaptive Recognition **Nordic**
Adaptive Recognition **Hungary** Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

Pseudo code

```
{
	"Config":
	{
		"BuiltIn": ...,
		"Class": "...",
		"Description": "...",
		"DetectorClassID": ...,
		"DetectorID": "{...}",
		"DisplayName": "...",
		"Enabled": ...,
		"FpsLimit": ...,
		"RestoreDelayMs": ...,
		"Version": ...,
		"ViolationTimeMs": ...
	},
	"DetectorID": "{...}","
}
```

14.10 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": "..."
}
```


Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.11 DETECTORCONFIGURATION

Inherited by: DetectorConfigurationANPR, DetectorConfigurationIO, DetectorConfigurationTest

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	*unused*
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an event is emitted. Not alldetectors may use this field.

Pseudo code

```
{
    "BuiltIn": ...,
    "Class": "...",
    "Description": "...",
    "DetectorClassID": ...,
    "DetectorID": "{...}",
    "DisplayName": "...",
    "Enabled": ...,
    "FpsLimit": ...,
    "RestoreDelayMs": ...,
    "Version": ...,
    "ViolationTimeMs": ...
}
```

14.12 DETECTORCONFIGURATIONANPR

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
AllowEmptyText	bool	Enable signalling when a license plate has no text
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
Direction	List/ string	List of directions to signal for
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
Filter	string	New-line separated list of license plates to signal for
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
FuzzySearch	bool	Enable fuzzy matching when using license plate filters
RestoreDelayMs	int64	*unused*
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an event is emitted. Not alldetectors may use this field.
Whitelist	bool	Enable filter usage

Pseudo code

```
{
    "AllowEmtyText": ...,
    "BuiltIn": ...,
    "Class": "...",
    "Description": "...",
    "DetectorClassID": ...,
    "DetectorID": "{...}",
    "Direction":
    {
        "0": "...",
        "1": "..."
    }
    "DisplayName": "...",
    "Enabled": ...,
    "Filter": "...",
    "FpsLimit": ...,
    "FuzzySearch": ...,
    "RestoreDelayMs": ...,
    "Version": ...,
    "ViolationTimeMs": ...,
    "Whitelist": ...
}
```

14.13 DETECTORCONFIGURATIONEMERGENCYLANE

Structure

Parameter	Type	Description
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Center	bool	Set to true to operate using an object's center point instead of all corners
Class	string	Detector type name
Confidence	int8	Minimum allowed object confidence when [strong]ConfidenceEnabled[/strong] is set to true
ConfidenceEnabled	bool	Set to true to use a confidence treshold for object monitoring
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.
Masks	List/Array/int16	Mask defining the working area of the detector (see GeometryPolygons)
ObjectTypes	List/string	List of object types that are monitored or empty list for all types
RestoreDelayMs	int64	[em]unused[/em]
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": ...,
	"Masks":
	{
		"0": [ ..., ..., ... ],
		"1": [ ..., ..., ... ]
	},
	"ObjectTypes":
	{
		"0": "...",
		"1": "..."
	},
	"RestoreDelayMs": ...,
	"Version": ...,
	"ViolationTimeMs": ...
}
```


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.14 DETECTORCONFIGURATIONFORBIDDENZONE

Structure

Parameter	Type	Description
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Center	bool	Set to true to operate using an object's center point instead of all corners
Class	string	Detector type name
Confidence	int8	Minimum allowed object confidence when [strong]ConfidenceEnabled[/strong] is set to true
ConfidenceEnabled	bool	Set to true to use a confidence treshold for object monitoring
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.
Masks	List/Array/int16	Mask defining the working area of the detector (see GeometryPolygons)
ObjectTypes	List/string	List of object types that are monitored or empty list for all types
RestoreDelayMs	int64	[em]unused[/em]
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.

Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

Pseudo code

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


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.15 DETECTORCONFIGURATIONIO

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
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.
InputPort	string	Name of the input port to monitor
RestoreDelayMs	int64	[em]unused[/em]
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": ...,
	"DetectorID": "{...}",
	"DisplayName": "...",
	"Enabled": ...,
	"FpsLimit": ...,
	"InputPort": "...",
	"RestoreDelayMs": ...,
	 "Version": ...,
	 "ViolationTimeMs": ...
}
```

14.16 DETECTORCONFIGURATIONLANE

Structure

Parameter	Type	Description
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Center	bool	Set to true to operate using an object's center point instead of all corners
Class	string	Detector type name
Confidence	int8	Minimum allowed object confidence when [strong]ConfidenceEnabled[/strong] is set to true
ConfidenceEnabled	bool	Set to true to use a confidence treshold for object monitoring
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.
Masks	List/Array/int16	Mask defining the working area of the detector (see GeometryPolygons)
ObjectTypes	List/string	List of object types that are monitored or empty list for all types
RestoreDelayMs	int64	[em]unused[/em]
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": ...,
	"Masks":
	{
		"0": [ ..., ..., ... ],
		"1": [ ..., ..., ... ]
	},
	"ObjectTypes":
	{
		"0": "...",
		"1": "..."
	},
	"RestoreDelayMs": ...,
	"Version": ...,
	"ViolationTimeMs": ...
}
```


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.17 DETECTORCONFIGURATIONREDSTOP

Detector monitors for objects that cross [strong]Lines[/strong] and leave the area through ExitLines after the light turns red and GracePeriodhad elapsed. The TrafficLighttype can be configured to be RogColumn (vertical road traffic light), RrwRailRoad (triangular railroad light) or RrwRailRoad2 (horizontal railroad light).

Structure

Parameter	Type	Description
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Center	bool	Set to true to operate using an object's center point instead of all corners
Class	string	Detector type name
Confidence	int8	Minimum allowed object confidence when [strong]ConfidenceEnabled[/strong] is set to true
ConfidenceEnabled	bool	Set to true to use a confidence treshold for object monitoring
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
Direction	string	[em]unused[/em]
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
ExitLines	IndexedTrackingDetectorLines	List of segments defining the exit line of the detector
Id	int8	Index of the line
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
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
GracePeriod	int64	The grace period in milliseconds after the a light turns red where crossing is still allowed
Lines	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
ObjectTypes	List/string	List of object types that are monitored or empty list for all types
RestoreDelayMs	int64	[em]unused[/em]
TrafficLight		
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.

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

Pseudo code

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

14.18 DETECTORCONFIGURATIONSTOPVIOLATION

Structure

Parameter	Type	Description
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Center	bool	Set to true to operate using an object's center point instead of all corners
Class	string	Detector type name
Confidence	int8	Minimum allowed object confidence when [strong]ConfidenceEnabled[/strong] is set to true
ConfidenceEnabled	bool	Set to true to use a confidence treshold for object monitoring
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
Direction	string	[em]unused[/em]
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.
GracePeriod	int64	The grace period in milliseconds after the a light turns red where crossing is still allowed
Lines	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
ObjectTypes	List/string	List of object types that are monitored or empty list for all types
RestoreDelayMs	int64	[em]unused[/em]
TrafficLight		
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": "{...}",
	"Direction": "...",
	"DisplayName": "...",
	"Enabled": ...,
	"FpsLimit": ...,
	"GracePeriod": ...,
	"Lines":
	{
		"X0": ...,
		"X1": ...,
		"Y0": ...,
		"Y1": ...
	},
	"ObjectTypes":
	{
		"0": "...",
		"1": "..."
	},
	"RestoreDelayMs": ...,
	"TrafficLight": ,
	"Version": ...,
	"ViolationTimeMs": ...
}
```

14.19 DETECTORCONFIGURATIONSTOPPEDOBJECT

Structure

Parameter	Type	Description
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Center	bool	Set to true to operate using an object's center point instead of all corners
Class	string	Detector type name
Confidence	int8	Minimum allowed object confidence when [strong]ConfidenceEnabled[/strong] is set to true
ConfidenceEnabled	bool	Set to true to use a confidence treshold for object monitoring
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.
Masks	List/Array/int16	Mask defining the working area of the detector (see GeometryPolygons)
ObjectTypes	List/string	List of object types that are monitored or empty list for all types
RestoreDelayMs	int64	[em]unused[/em]
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": ...,
	"Masks":
	{
		"0": [ ..., ..., ... ],
		"1": [ ..., ..., ... ]
	},
	"ObjectTypes":
	{
		"0": "...",
		"1": "..."
	},
	"RestoreDelayMs": ...,
	"Version": ...,
	"ViolationTimeMs": ...
}
```


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.20 DETECTORCONFIGURATIONTEST

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 emiting 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
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.
Interval	int64	Duration of normal state in milliseconds
RestoreDelayMs	int64	*unused*
Timeout	int64	Duration of signalling state in milliseconds
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an event is emitted. Not alldetectors may use this field.

Pseudo code

```
{
      "BuiltIn": ...,
      "Class": "...",
      "Description": "...",
      "DetectorClassID": ...,
      "DetectorID": "{...}",
      "DisplayName": "...",
      "Enabled": ...,
      "FpsLimit": ...,
      "Interval": ...,
      "RestoreDelayMs": ...,
      "Timeout": ...,
       "Version": ...,
      "ViolationTimeMs": ...
}
```


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.21 DETECTORCONFIGURATIONTRAFFICLINE

Structure

Parameter	Type	Description
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Center	bool	Set to true to operate using an object's center point instead of all corners
Class	string	Detector type name
Confidence	int8	Minimum allowed object confidence when [strong]ConfidenceEnabled[/strong] is set to true
ConfidenceEnabled	bool	Set to true to use a confidence treshold for object monitoring
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
Direction	string	[em]unused[/em]
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.
GracePeriod	int64	The grace period in milliseconds after the a light turns red where crossing is still allowed
Lines	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
ObjectTypes	List/string	List of object types that are monitored or empty list for all types
RestoreDelayMs	int64	[em]unused[/em]
TrafficLight		
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": "{...}",
	"Direction": "...",
	"DisplayName": "...",
	"Enabled": ...,
	"FpsLimit": ...,
	"GracePeriod": ...,
	"Lines":
	{
		"X0": ...,
		"X1": ...,
		"Y0": ...,
		"Y1": ...
	},
	"ObjectTypes":
	{
		"0": "...",
		"1": "..."
	},
	"RestoreDelayMs": ...,
	"TrafficLight": ,
	"Version": ...,
	"ViolationTimeMs": ...
}
```

14.22 DETECTORCONFIGURATIONUTURN

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

Structure

Parameter	Type	Description
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Center	bool	Set to true to operate using an object's center point instead of all corners
Class	string	Detector type name
Confidence	int8	Minimum allowed object confidence when [strong]ConfidenceEnabled[/strong] is set to true
ConfidenceEnabled	bool	Set to true to use a confidence treshold for object monitoring
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
Direction	string	[em]unused[/em]
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.
GracePeriod	int64	The grace period in milliseconds after the a light turns red where crossing is still allowed
Lines	GeometryLineSegment	List of segments defining the entry line for the detector (see GeometryLine)
Masks	List/Array/int16	List of masks. Each mask is a list of coordinates where odd and even indicies are x and y coordinates of a corner in the polygon (x0, y0, x1, y1, ...).
ObjectTypes	List/string	List of object types that are monitored or empty list for all types
RestoreDelayMs	int64	[em]unused[/em]
TrafficLight		
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": "{...}",
	"Direction": "...",
	"DisplayName": "...",
	"Enabled": ...,
	"FpsLimit": ...,
	"Lines":
	{
		"Masks":
		{
			"0": [ ..., ..., ... ],
			"1": [ ..., ..., ... ]
		}
	},
	"ObjectTypes":
	{
		"0": "...",
		"1": "..."
	},
	"RestoreDelayMs": ...,
	"Version": ...,
	"ViolationTimeMs": ...
}
```

14.23 DETECTORCONFIGURATIONWHITELINEVIOLATION

Structure

Parameter	Type	Description
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Center	bool	Set to true to operate using an object's center point instead of all corners
Class	string	Detector type name
Confidence	int8	Minimum allowed object confidence when [strong]ConfidenceEnabled[/strong] is set to true
ConfidenceEnabled	bool	Set to true to use a confidence treshold for object monitoring
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
Direction	string	[em]unused[/em]
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.
GracePeriod	int64	The grace period in milliseconds after the a light turns red where crossing is still allowed
Lines	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
ObjectTypes	List/string	List of object types that are monitored or empty list for all types
RestoreDelayMs	int64	[em]unused[/em]
TrafficLight		
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.

Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

Pseudo code

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


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.24 DETECTORCONFIGURATIONWRONGTURN

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

Structure

Parameter	Type	Description
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Center	bool	Set to true to operate using an object's center point instead of all corners
Class	string	Detector type name
Confidence	int8	Minimum allowed object confidence when [strong]ConfidenceEnabled[/strong] is set to true
ConfidenceEnabled	bool	Set to true to use a confidence treshold for object monitoring
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
Direction	string	[em]unused[/em]
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.
LineGroups	GeometryLineGroup	Mask defining the working area of the detector (see GeometryLineGroups)
Lines	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
SequenceNumber	int32	Numeric id of this group for ordering
ObjectTypes	List/string	List of object types that are monitored or empty list for all types
RestoreDelayMs	int64	[em]unused[/em]
TrafficLight		
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": ...,
	"LineGroups":
	{
		"Lines":
		{
			"X0": ...,
			"X1": ...,
			"Y0": ...,
			"Y1": ...
		},
		"SequenceNumber": ...
	},
	"ObjectTypes":
	{
		"0": "...",
		"1": "..."
	},
	"RestoreDelayMs": ...,
	"Version": ...,
	"ViolationTimeMs": ...
}
```


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.25 DETECTORCONFIGURATIONWRONGWAY

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 [em]Angle=90[/em] and [em]AngleRange=10[/em] 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
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Center	bool	Set to true to operate using an object's center point instead of all corners
Class	string	Detector type name
Confidence	int8	Minimum allowed object confidence when [strong]ConfidenceEnabled[/strong] is set to true
ConfidenceEnabled	bool	Set to true to use a confidence treshold for object monitoring
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.
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 GeometryPolygons)
ObjectTypes	List/string	List of object types that are monitored or empty list for all types
RestoreDelayMs	int64	[em]unused[/em]
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

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

14.26 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": "{...}"
}
```

14.27 DETECTORCREATECONFIGURATION

Structure

Parameter	Type	Description
Description	string	User-specified description
Location		User-specified location
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees
Network		
Host	String	Hostname or IP address of the location transmitter
Port	Int32	Port of the location transmitter service on the server
Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties
Name	String	User-specified name
Serial	String	Unique device serial number

Pseudo code

```
{
                "Description": "...",
                "Location":
                {
                    "Config":
                    {
                      "Manual":
                      {
                        "Position":
                        {
                            "Latitude" :...,
                            "Longitude":...
                         }
                      },
                      "Network":
                      {
                          "Host": "...",
                          "Port": ...,
                          "Protocol": "..."
                       }
                    },
                    "Mode": "..."
                }
                "Name": "...",
                "Serial": "..."
}
```

14.28 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 delete 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": ...,
    "DetectorID": "{...}",
    "DisplayName": "...",
    "State": "...",
    "Version": ...
}
```

14.29 DETECTORLIST

> **See also:** Analytics/GetDetectors

Structure

Parameter	Type	Description
Detectors	List/**DetectorInfo**	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":
      {
            "BuiltIn": ...,
            "Description": "...",
            "DetectorClass": "...",
            "DetectorClassID": ...,"DetectorID":
            "{...}",
            "DisplayName": "...",
            "State": "...",
            "Version": ...
       }
}
```

14.30 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": "{...}"
}
```


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.31 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 loadingconfiguration
2	dsError	Detector is in an erroneous state and cannot operate
3	dsUnableToOperate	The current device environment does not allow normal operation ofdetector. 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. Detectoroperation 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": ...
}
```

14.32 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": ...
}
```


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.33 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 onlyserve to mark the end of a detected occurence.

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 initialized 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.

Events have two timestamps that may have different values based on detector operation. EventTime is the timestamp when the detector event was created because all required conditions were met. EventTriggerTime may be an earlier timestamp that points to the exact moment the interest of the event happened. For example with tracking detectors where a line is monitored an object crosses a line but it is not yet validated or categorized. The moment of crossing is saved as the original trigger time and when later the object is validated an event is emitted with the current timestamp but with an earlier trigger timestamp.

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

Structure

Parameter	Type	Description
Config	DetectorConfiguration	Configuration of the detector when this event was created
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	[em]unused[/em]
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.
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
Device	DetectorDeviceInfo	Information about the device where the detector operates
Description	string	User-specified description
Location	LocationSettings	User-specified location
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees
Network		
Host	String	Hostname or IP address of the location transmitter
Port	Int32	Port of the location transmitter service on the server

Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties
Name	String	User-specified name
Serial	String	Unique device serial number
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 DetectorState)

Pseudo code

```
{
    "Config":
    {
        "BuiltIn": ...,
        "Class": "...",
        "Description": "...",
        "DetectorClassID": ...,
        "DetectorID": "{...}",
        "DisplayName": "...",
        "Enabled": ...,
        "FpsLimit": ...,
        "RestoreDelayMs": ...,
        "Version": ...,
        "ViolationTimeMs": ...
    },
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "Device":
    {
            "Description": "...",
            "Location":
            {
                "Config":
                {
                   "Manual":
                   {
                      "Position":
                      {
                          "Latitude" :...,
                          "Longitude":...
                       }
                   },
                   "Network":
                   {
                       "Host": "...",
                       "Port": ...,
                       "Protocol": "..."
                    }
                },
                "Mode": "..."
            }
            "Name": "...",
           "Serial": "..."
    }
    "EventCode": ...,
    "EventID": "{...}",
    "EventTime": ...,
    "EventTriggerTime": ...,
    "State": "..."
}
```

14.34 EVENTANPR

License plate detection event. The EventCode associated with this event is 114.

Structure

Parameter	Type	Description
Config	DetectorConfiguration	Configuration of the detector when this event was created
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	[em]unused[/em]
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.
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
Device	DetectorDeviceInfo	Information about the device where the detector operates
Description	string	User-specified description
Location	LocationSettings	User-specified location
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees

Network		
Host	String	Hostname or IP address of the location transmitter
Port	Int32	Port of the location transmitter service on the server
Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties
Name	String	User-specified name
Serial	String	Unique device serial number
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventInfo	EventANPRLicensePlate	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 [strong]Approaching[/strong], [strong]Moving away[/strong] or [strong]Unknown[/strong].
MMR		Make and model recognition results
Category	string	Vehicle category
CategoryConfidence	double	Confidence of the category recognition
Color	string double	Color of vehicle in #RRGGBB format
ColorConfidence		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
Timestamp	string	Timestamp of when the trigger was activated
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 DetectorState)

Pseudo code

```
{
    "Config":
    {
        "BuiltIn": ...,
        "Class": "...",
        "Description": "...",
        "DetectorClassID": ...,
        "DetectorID": "{...}",
        "DisplayName": "...",
        "Enabled": ...,
        "FpsLimit": ...,
        "RestoreDelayMs": ...,
        "Version": ...,
        "ViolationTimeMs": ...
    },
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "Device":
    {
            "Description": "...",
            "Location":
            {
                "Config":
                {
                  "Manual":
                  {
                    "Position":
                    {
                        "Latitude" :...,
                        "Longitude":...
                     }
                  },
                  "Network":
                  {
                      "Host": "...",
                      "Port": ...,
                      "Protocol": "..."
                   }
                },
                "Mode": "..."
            }
            "Name": "...",
          "Serial": "..."
    }
    "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": "..."
}
```

14.35 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 [strong]Approaching[/strong], [strong]Moving away[/strong] or [strong]Unknown[/strong].
MMR		Make and model recognition results
Category	string	Vehicle category
CategoryConfidence	double	Confidence of the category recognition
Color	string double	Color of vehicle in #RRGGBB format
ColorConfidence		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
Timestamp	string	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": "...",
       }

}
```

14.36 EVENTEMERGENCYLANE

Structure

Parameter	Type	Description
Config	DetectorConfiguration	Configuration of the detector when this event was created
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	[em]unused[/em]
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.
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
Device	DetectorDeviceInfo	Information about the device where the detector operates
Description	string	User-specified description
Location	LocationSettings	User-specified location
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees
Network		
Host	String	Hostname or IP address of the location transmitter

Port	Int32	Port of the location transmitter service on the server
Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties
Name	String	User-specified name
Serial	String	Unique device serial number
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventInfo	TrackedObjectInfo	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
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 DetectorState)

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

Pseudo code

```
{
    "Config":
    {
        "BuiltIn": ...,
        "Class": "...",
        "Description": "...",
        "DetectorClassID": ...,
        "DetectorID": "{...}",
        "DisplayName": "...",
        "Enabled": ...,
        "FpsLimit": ...,
        "RestoreDelayMs": ...,
        "Version": ..., "ViolationTimeMs":
        ...
    },
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "Device":
    {
            "Description": "...",
            "Location":
            {
                "Config":
                {
                  "Manual":
                  {
                    "Position":
                    {
                        "Latitude" :...,
                        "Longitude":...
                     }
                  },
                  "Network":
                  {
                      "Host": "...",
                      "Port": ...,
                      "Protocol": "..."
                   }
                },
                "Mode": "..."
            }
            "Name": "...",
          "Serial": "..."
    }
    "EventCode": ...,
```

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

14.37 EVENTFORBIDDENZONE

Structure

Parameter	Type	Description
Config	DetectorConfiguration	Configuration of the detector when this event was created
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	[em]unused[/em]
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.
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
Device	DetectorDeviceInfo	Information about the device where the detector operates
Description	string	User-specified description
Location	LocationSettings	User-specified location
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees
Network		
Host	String	Hostname or IP address of the location transmitter

Port	Int32	Port of the location transmitter service on the server
Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties
Name	String	User-specified name
Serial	String	Unique device serial number
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventInfo	TrackedObjectInfo	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
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 DetectorState)

Pseudo code

```
{
    "Config":
    {
        "BuiltIn": ...,
        "Class": "...",
        "Description": "...",
        "DetectorClassID": ...,
        "DetectorID": "{...}",
        "DisplayName": "...",
        "Enabled": ...,
        "FpsLimit": ...,
        "RestoreDelayMs": ...,
        "Version": ..., "ViolationTimeMs":
        ...
    },
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "Device":
    {
            "Description": "...",
            "Location":
            {
                "Config":
                {
                  "Manual":
                  {
                    "Position":
                    {
                        "Latitude" :...,
                        "Longitude":...
                     }
                  },
                  "Network":
                  {
                      "Host": "...",
                      "Port": ...,
                      "Protocol": "..."
                   }
                },
                "Mode": "..."
            }
            "Name": "...",
          "Serial": "..."
    }
    "EventCode": ...,
```

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


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.38 EVENTIO

Input port activation event

Structure

Parameter	Type	Description
Config	DetectorConfiguration	Configuration of the detector when this event was created
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	[em]unused[/em]
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.
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
Device	DetectorDeviceInfo	Information about the device where the detector operates
Description	string	User-specified description
Location	LocationSettings	User-specified location
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees

Longitude	double	Longitude coordinate in decimal degrees
Network		
Host	String	Hostname or IP address of the location transmitter
Port	Int32	Port of the location transmitter service on the server
Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties
Name	String	User-specified name
Serial	String	Unique device serial number
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 DetectorState)

Pseudo code

```
{
    "Config":
    {
        "BuiltIn": ...,
        "Class": "...",
        "Description": "...",
        "DetectorClassID": ...,
        "DetectorID": "{...}",
        "DisplayName": "...",
        "Enabled": ...,
        "FpsLimit": ...,
        "RestoreDelayMs": ...,
        "Version": ...,
        "ViolationTimeMs": ...
    },
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "Device":
    {
            "Description": "...",
            "Location":
            {
                "Config":
                {
                  "Manual":
                  {
                     "Position":
                     {
                         "Latitude" :...,
                         "Longitude":...
                      }
                  },
                  "Network":
                  {
                      "Host": "...",
                      "Port": ...,
                      "Protocol": "..."
                   }
                },
                "Mode": "..."
            }
            "Name": "...",
           "Serial": "..."
    }
    "EventCode": ...,
    "EventID": "{...}",
    "EventTime": ...,
    "EventTriggerTime": ...,
    "State": "..."
}
```

14.39 EVENTLANE

Structure

Parameter	Type	Description
Config	DetectorConfiguration	Configuration of the detector when this event was created
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	[em]unused[/em]
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.
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
Device	DetectorDeviceInfo	Information about the device where the detector operates
Description	string	User-specified description
Location	LocationSettings	User-specified location
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees
Network		
Host	String	Hostname or IP address of the location transmitter

Port	Int32	Port of the location transmitter service on the server
Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties
Name	String	User-specified name
Serial	String	Unique device serial number
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventInfo	TrackedObjectInfo	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
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 DetectorState)

Pseudo code

```
{
    "Config":
    {
        "BuiltIn": ...,
        "Class": "...",
        "Description": "...",
        "DetectorClassID": ...,
        "DetectorID": "{...}",
        "DisplayName": "...",
        "Enabled": ...,
        "FpsLimit": ...,
        "RestoreDelayMs": ...,
        "Version": ..., "ViolationTimeMs":
        ...
    },
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "Device":
    {
            "Description": "...",
            "Location":
            {
                "Config":
                {
                  "Manual":
                  {
                    "Position":
                    {
                        "Latitude" :...,
                        "Longitude":...
                     }
                  },
                  "Network":
                  {
                      "Host": "...",
                      "Port": ...,
                      "Protocol": "..."
                   }
                },
                "Mode": "..."
            }
            "Name": "...",
          "Serial": "..."
    }
    "EventCode": ...,
```

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

14.40 EVENTREDSTOP

Structure

Parameter	Type	Description
Config	DetectorConfiguration	Configuration of the detector when this event was created
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	[em]unused[/em]
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.
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
Device	DetectorDeviceInfo	Information about the device where the detector operates
Description	string	User-specified description
Location	LocationSettings	User-specified location
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees

Network		
Host	String	Hostname or IP address of the location transmitter
Port	Int32	Port of the location transmitter service on the server
Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties
Name	String	User-specified name
Serial	String	Unique device serial number
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventInfo	RedStopViolationInfo	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
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
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
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 DetectorState)

Pseudo code

```
{
    "Config":
    {
        "BuiltIn": ...,
        "Class": "...",
        "Description": "...",
        "DetectorClassID": ...,
        "DetectorID": "{...}",
        "DisplayName": "...",
        "Enabled": ...,
        "FpsLimit": ...,
        "RestoreDelayMs": ...,
        "Version": ..., "ViolationTimeMs":
        ...
    },
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "Device":
    {
            "Description": "...",
            "Location":
            {
                "Config":
                {
                  "Manual":
                  {
                    "Position":
                    {
                        "Latitude" :...,
                        "Longitude":...
                     }
                  },
                  "Network":
                  {
                      "Host": "...",
                      "Port": ...,
                      "Protocol": "..."
                   }
                },
                "Mode": "..."
            }
            "Name": "...",
          "Serial": "..."
    }
    "EventCode": ...,
```

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

14.41 EVENTSTOPVIOLATION

Structure

Parameter	Type	Description
Config	DetectorConfiguration	Configuration of the detector when this event was created
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	[em]unused[/em]
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.
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
Device	DetectorDeviceInfo	Information about the device where the detector operates
Description	string	User-specified description
Location	LocationSettings	User-specified location
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees

Network		
Host	String	Hostname or IP address of the location transmitter
Port	Int32	Port of the location transmitter service on the server
Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties
Name	String	User-specified name
Serial	String	Unique device serial number
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventInfo	TrackedObjectInfo	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
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 DetectorState)

Pseudo code

```
{
    "Config":
    {
        "BuiltIn": ...,
        "Class": "...",
        "Description": "...",
        "DetectorClassID": ...,
        "DetectorID": "{...}",
        "DisplayName": "...",
        "Enabled": ...,
        "FpsLimit": ...,
        "RestoreDelayMs": ...,
        "Version": ..., "ViolationTimeMs":
        ...
    },
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "Device":
    {
            "Description": "...",
            "Location":
            {
                "Config":
                {
                  "Manual":
                  {
                    "Position":
                    {
                        "Latitude" :...,
                        "Longitude":...
                     }
                  },
                  "Network":
                  {
                      "Host": "...",
                      "Port": ...,
                      "Protocol": "..."
                   }
                },
                "Mode": "..."
            }
            "Name": "...",
          "Serial": "..."
    }
    "EventCode": ...,
```

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

14.42 EVENTSTOPPEDOBJECT

Structure

Parameter	Type	Description
Config	DetectorConfiguration	Configuration of the detector when this event was created
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	[em]unused[/em]
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.
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
Device	DetectorDeviceInfo	Information about the device where the detector operates
Description	string	User-specified description
Location	LocationSettings	User-specified location
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees

Network		
Host	String	Hostname or IP address of the location transmitter
Port	Int32	Port of the location transmitter service on the server
Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties
Name	String	User-specified name
Serial	String	Unique device serial number
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventInfo	TrackedObjectInfo	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
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 DetectorState)

Pseudo code

```
{
    "Config":
    {
        "BuiltIn": ...,
        "Class": "...",
        "Description": "...",
        "DetectorClassID": ...,
        "DetectorID": "{...}",
        "DisplayName": "...",
        "Enabled": ...,
        "FpsLimit": ...,
        "RestoreDelayMs": ...,
        "Version": ..., "ViolationTimeMs":
        ...
    },
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "Device":
    {
            "Description": "...",
            "Location":
            {
                "Config":
                {
                  "Manual":
                  {
                    "Position":
                    {
                        "Latitude" :...,
                        "Longitude":...
                     }
                  },
                  "Network":
                  {
                      "Host": "...",
                      "Port": ...,
                      "Protocol": "..."
                   }
                },
                "Mode": "..."
            }
            "Name": "...",
          "Serial": "..."
    }
    "EventCode": ...,
```

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

14.43 EVENTTEST

Basic test event

Structure

Parameter	Type	Description
Config	DetectorConfiguration	Configuration of the detector when this event was created
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	[em]unused[/em]
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.
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
Device	DetectorDeviceInfo	Information about the device where the detector operates
Description	string	User-specified description
Location	LocationSettings	User-specified location
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees
Network		

Host	String	Hostname or IP address of the location transmitter
Port	Int32	Port of the location transmitter service on the server
Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties
Name	String	User-specified name
Serial	String	Unique device serial number
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
Index	int64	A numeric counter that increments when the detector emitted an event of any type
State	string	State of the detector after the event was emitted (see DetectorState)

Pseudo code

```
{
    "Config":
    {
        "BuiltIn": ...,
        "Class": "...",
        "Description": "...",
        "DetectorClassID": ...,
        "DetectorID": "{...}",
        "DisplayName": "...",
        "Enabled": ...,
        "FpsLimit": ...,
        "RestoreDelayMs": ...,
        "Version": ...,
        "ViolationTimeMs": ...
    },
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "Device":
    {
            "Description": "...",
            "Location":
            {
                "Config":
                {
                  "Manual":
                  {
                     "Position":
                     {
                         "Latitude" :...,
                         "Longitude":...
                      }
                  },
                  "Network":
                  {
                      "Host": "...",
                      "Port": ...,
                      "Protocol": "..."
                   }
                },
                "Mode": "..."
            }
            "Name": "...",
           "Serial": "..."
    }
    "EventCode": ...,
    "EventID": "{...}",
    "EventTime": ...,
    "EventTriggerTime": ...,
    "Index": ...,
    "State": "..."
}
```

14.44 EVENTTRAFFICLINE

Structure

Parameter	Type	Description
Config	DetectorConfiguration	Configuration of the detector when this event was created
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	[em]unused[/em]
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.
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
Device	DetectorDeviceInfo	Information about the device where the detector operates
Description	string	User-specified description
Location	LocationSettings	User-specified location
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees
Network		
Host	String	Hostname or IP address of the location transmitter

Port	Int32	Port of the location transmitter service on the server
Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties
Name	String	User-specified name
Serial	String	Unique device serial number
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventInfo	TrackedObjectInfo	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
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 DetectorState)

Pseudo code

```
{
    "Config":
    {
        "BuiltIn": ...,
        "Class": "...",
        "Description": "...",
        "DetectorClassID": ...,
        "DetectorID": "{...}",
        "DisplayName": "...",
        "Enabled": ...,
        "FpsLimit": ...,
        "RestoreDelayMs": ...,
        "Version": ...,
        "ViolationTimeMs": ...
    },
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...
    "Device":
    {
            "Description": "...",
            "Location":
            {
                "Config":
                {
                  "Manual":
                  {
                    "Position":
                    {
                        "Latitude" :...,
                        "Longitude":...
                     }
                  },
                  "Network":
                  {
                      "Host": "...",
                      "Port": ...,
                      "Protocol": "..."
                   }
                },
                "Mode": "..."
            }
            "Name": "...",
          "Serial": "..."
    }
    "EventCode": ...,
```

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

14.45 EVENTUTURN

Structure

Parameter	Type	Description
Config	DetectorConfiguration	Configuration of the detector when this event was created
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	[em]unused[/em]
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.
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
Device	DetectorDeviceInfo	Information about the device where the detector operates
Description	string	User-specified description
Location	LocationSettings	User-specified location
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees

Network		
Host	String	Hostname or IP address of the location transmitter
Port	Int32	Port of the location transmitter service on the server
Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties
Name	String	User-specified name
Serial	String	Unique device serial number
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventInfo	TrackedObjectInfo	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
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 DetectorState)

Pseudo code

```
{
    "Config":
    {
        "BuiltIn": ...,
        "Class": "...",
        "Description": "...",
        "DetectorClassID": ...,
        "DetectorID": "{...}",
        "DisplayName": "...",
        "Enabled": ...,
        "FpsLimit": ...,
        "RestoreDelayMs": ...,
        "Version": ...,
        "ViolationTimeMs": ...
    },
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "Device":
    {
            "Description": "...",
            "Location":
            {
                "Config":
                {
                  "Manual":
                  {
                    "Position":
                    {
                        "Latitude" :...,
                        "Longitude":...
                     }
                  },
                  "Network":
                  {
                      "Host": "...",
                      "Port": ...,
                      "Protocol": "..."
                   }
                },
                "Mode": "..."
            }
            "Name": "...",
          "Serial": "..."
    }
    "EventCode": ...,
```

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

14.46 EVENTWHITELINEVIOLATION

Structure

Parameter	Type	Description
Config	DetectorConfiguration	Configuration of the detector when this event was created
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	[em]unused[/em]
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.
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
Device	DetectorDeviceInfo	Information about the device where the detector operates
Description	string	User-specified description
Location	LocationSettings	User-specified location
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees

Network		
Host	String	Hostname or IP address of the location transmitter
Port	Int32	Port of the location transmitter service on the server
Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties
Name	String	User-specified name
Serial	String	Unique device serial number
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventInfo	TrackedObjectInfo	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
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 DetectorState)

Pseudo code

```
{
    "Config":
    {
        "BuiltIn": ...,
        "Class": "...",
        "Description": "...",
        "DetectorClassID": ...,
        "DetectorID": "{...}",
        "DisplayName": "...",
        "Enabled": ...,
        "FpsLimit": ...,
        "RestoreDelayMs": ...,
        "Version": ...,
        "ViolationTimeMs": ...
    },
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "Device":
    {
            "Description": "...",
            "Location":
            {
                "Config":
                {
                  "Manual":
                  {
                    "Position":
                    {
                        "Latitude" :...,
                        "Longitude":...
                     }
                  },
                  "Network":
                  {
                      "Host": "...",
                      "Port": ...,
                      "Protocol": "..."
                   }
                },
                "Mode": "..."
            }
            "Name": "...",
          "Serial": "..."
    }
    "EventCode": ...,
```

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

14.47 EVENTWRONGTURN

Structure

Parameter	Type	Description
Config	DetectorConfiguration	Configuration of the detector when this event was created
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.
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
Device	DetectorDeviceInfo	Information about the device where the detector operates
Description	string	User-specified description
Location	LocationSettings	User-specified location
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees
Network		
Host	String	Hostname or IP address of the location transmitter

Port	Int32	Port of the location transmitter service on the server
Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties
Name	String	User-specified name
Serial	String	Unique device serial number
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventInfo	TrackedObjectInfo	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
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 DetectorState)

Pseudo code

```
{
    "Config":
    {
        "BuiltIn": ...,
        "Class": "...",
        "Description": "...",
        "DetectorClassID": ...,
        "DetectorID": "{...}",
        "DisplayName": "...",
        "Enabled": ...,
        "FpsLimit": ...,
        "RestoreDelayMs": ...,
        "Version": ...,
        "ViolationTimeMs": ...
    },
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "Device":
    {
            "Description": "...",
            "Location":
            {
                "Config":
                {
                  "Manual":
                  {
                    "Position":
                    {
                        "Latitude" :...,
                        "Longitude":...
                     }
                  },
                  "Network":
                  {
                      "Host": "...",
                      "Port": ...,
                      "Protocol": "..."
                   }
                },
                "Mode": "..."
            }
            "Name": "...",
          "Serial": "..."
    }
    "EventCode": ...,
```

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

14.48 EVENTWRONGWAY

Structure

Parameter	Type	Description
Config	DetectorConfiguration	Configuration of the detector when this event was created
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	[em]unused[/em]
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.
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
Device	DetectorDeviceInfo	Information about the device where the detector operates
Description	string	User-specified description
Location	LocationSettings	User-specified location
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees
Network		
Host	String	Hostname or IP address of the location transmitter

Port	Int32	Port of the location transmitter service on the server
Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties
Name	String	User-specified name
Serial	String	Unique device serial number
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventInfo	TrackedObjectInfo	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
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 DetectorState)

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

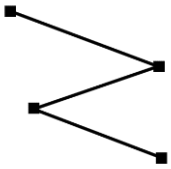
Pseudo code

```
{
    "Config":
    {
        "BuiltIn": ...,
        "Class": "...",
        "Description": "...",
        "DetectorClassID": ...,
        "DetectorID": "{...}",
        "DisplayName": "...",
        "Enabled": ...,
        "FpsLimit": ...,
        "RestoreDelayMs": ...,
        "Version": ...,
        "ViolationTimeMs": ...
    },
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "Device":
    {
            "Description": "...",
            "Location":
            {
                "Config":
                {
                  "Manual":
                  {
                    "Position":
                    {
                        "Latitude" :...,
                        "Longitude":...
                     }
                  },
                  "Network":
                  {
                      "Host": "...",
                      "Port": ...,
                      "Protocol": "..."
                   }
                },
                "Mode": "..."
            }
            "Name": "...",
          "Serial": "..."
    }
    "EventCode": ...,
```

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

14.49 GEOMETRYLINE

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



Structure

Parameter	Type	Description
Lines	List/**GeometryLineSegment**	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

Pseudo code

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


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.50 GEOMETRYLINEGROUP

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

Structure

Parameter	Type	Description
Lines	List/**GeometryLineSegment**	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":
    {
            "X0": ...,
            "X1": ...,
            "Y0": ...,
            "Y1": ...
        }
    },
    "SequenceNumber": ...
}
```

14.51 GEOMETRYLINEGROUPS

Groups of segmented lines where an order of groups is formed using indicies

Structure

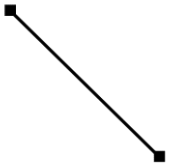
Parameter	Type	Description
LineGroups	List/**GeometryLineGroup**	List of line groups
Lines	List/**GeometryLineSegment**	List of line segments
X0	int32	X coordinate of the start point
X1	int32	X coordinate of the end point
Y0	int32	Y coordinate of the start point
Y1	int32	Y coordinate of the end point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

Pseudo code

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

14.52 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

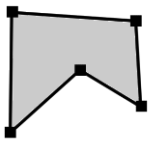
Pseudo code

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


Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.53 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 indicies arex and y coordinates of a corner in the polygon (x0, y0, x1, y1, ...).

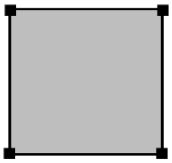
Pseudo code

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


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.54 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": ...
}
```


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.55 GPIOINPUTPORT

Settings of a digital input port

> **See also:** System/SetGpioInputSettings

Structure

Parameter	Type	Description
Inherited from GpioPort:		
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": "..."
}
```

14.56 GPIOOUTPUTPORT

Settings of a digital output port

> **See also:** System/SetGpioOutputSettings

Structure

Parameter	Type	Description
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
OutputMode	string	Output signal form. Only the "Impulse" mode is supported.
Port	string	Unique identifier of a digital input/output port

Pseudo code

```
{
	"ActiveState": ...,
	"ActiveTime": ...,
	"DetectorList":
	{
		"0": "{...}",
		"1": "{...}"
	},
	"OutputMode": "...",
	"Port": "..."
}
```

14.57 GPIOOUTPUTPORTSTATE

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
Port	string	Unique identifier of a digital input/output port

Pseudo code

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


Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.58 GPIOPORT

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)
Port	string	Unique identifier of a digital input/output port

Pseudo code

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


Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.59 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": "..."
}
```


Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.60 GPIOPORTSTATE

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
Port	string	Unique identifier of a digital input/output port

Pseudo code

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


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.61 GPIOPORTSTATECHANGE

Structure

Parameter	Type	Description
Type	string	Value of "Input" or "Output" indicating the port type
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": "..."
}
```


Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.62 GPIOSETTINGS

Settings of all digital input/output ports

See also: System/GetGpioSettings

Structure

Parameter	Type	Description
Inputs	Map/GpioInputPort	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/GpioOutputPort	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 withan event
OutputMode	string	Output signal form. Only the "Impulse" mode is supported.

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

Pseudo code

```
{
    "Inputs":
    {
        "ActiveState": ...,
        "Port": "..."
    },
    "Outputs":
    {
        "ActiveState": ...,
        "ActiveTime": ...,
        "DetectorList":
        {
            "0": "{...}",
            "1": "{...}"
        },
        "OutputMode": "...",
        "Port": "..."
    }
}
```

14.63 GPIOSTATES

Last known state of all digital input/output ports

> **See also:** System/GetGpioStates

Structure

Parameter	Type	Description
Inputs	GpioPortState	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 changedto this state
Outputs	GpioPortState	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 changedto this state

Pseudo code

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

14.64 INDEXEDTRACKINGDETECTORLINES

Structure

Parameter	Type	Description
Id	int8	Index of the line
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": ...
}
```


Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.65 LOCATIONSETTINGS

With **Mode** the device can switch its location data source.

In **Manual** mode the device uses the position saved in the **Manual** configuration.

In **Network** mode the device connects to the configured location service and periodically queries the current location. Protocol must be either TCP or UDP.

Structure

Parameter	Type	Description
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees
Network		
Host	String	Hostname or IP address of the location transmitter
Port	Int32	Port of the location transmitter service on the server
Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties

Pseudo code

```
{
    "Config":
    {
        "Manual":
        {
            "Position":
            {
                "Latitude": ...,
                "Longitude": ...
             }
         }
        "Network":
        {
            "Host": "...",
            "Port": ...,
            "Protocol": "..."
         }
    }
    "Mode": "..."
}
```

14.66 MODULEANALYTICS

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

Tracker	Supports the iTracking tracker engine (see **Analytics/GetTracker**)
TrafficDetectors	Supports traffic focused detectors
CarmenEngine	Supports CARMEN license plate recognition (see **Analytics/GetAnprEngine**)

Structure

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

Pseudo code

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


Adaptive Recognition **America** Adaptive Recognition **Hungary** Adaptive Recognition **Nordic** Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.67 MODULEIO

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": "..."
	}
}
```


Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.68 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": ...
}
```

14.69 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": "..."
	}
}
```


Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.70 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": ...
}
```


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.71 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": "...",
	"Values":
	{
		"0": "...",
		"1": "..."
	}
}
```

14.72 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": "..."
}
```


Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.73 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
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
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": "..."
}
```

14.74 SECURITYHISTORY

List of security related information like blocked sources and active sessions

> **See also:** System/GetSecurityHistory

Structure

Parameter	Type	Description
BlockedSources	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/**ActiveSession**	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":
    {
        "LastSeen": ...,
        "Source": "...",
        "User": "..."
    }
}
```

14.75 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 attemps before a source isblocked
SourceBlockDuration	int64	Block length in milliseconds

Pseudo code

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


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.76 STORAGEEVENTS

Result of a stored event query. The parameters of the original query are returned with **StartTime** and EndTimemodified 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 the the previously returned **EndTime**.

> **See also:** Storage/GetEvents

Structure

Parameter	Type	Description
EndTime	int64	Wall clock timestamp in milliseconds of the end of the search range
EventList	Event	List of events that match the search criteria
Config	DetectorConfiguration	Configuration of the detector when this event was created
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	[em]unused[/em]
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.
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
Device	DetectorDeviceInfo	Information about the device where the detector operates
Description	string	User-specified description
Location	LocationSettings	User-specified location
Config		
Manual		
Position		
Latitude	Double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees

Network		
Host	String	Hostname or IP address of the location transmitter
Port	Int32	Port of the location transmitter service on the server
Protocol	String	Protocol to use when connecting to the service
Mode	String	General system properties
Name	String	User-specified name
Serial	String	Unique device serial number
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 DetectorState)
Filter	StorageEventsRequestFilter	(optional) Additional filter parameters
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.
ID	guid	(optional) Unique ID of the detector to search for
StartTime	int64	Wall clock timestamp in milliseconds of the beggining of the search range
Status	string	Final status of the query

Pseudo code

```
{
    "EndTime": ...,
    "EventList":
    {
    "Config":
    {
        "BuiltIn": ...,
        "Class": "...",
        "Description": "...",
        "DetectorClassID": ...,
        "DetectorID": "{...}",
        "DisplayName": "...",
        "Enabled": ...,
        "FpsLimit": ...,
        "RestoreDelayMs": ...,
        "Version": ...,
        "ViolationTimeMs": ...
    },
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "Device":
    {
            "Description": "...",
            "Location":
            {
                "Config":
                {
                  "Manual":
                  {
                    "Position":
                    {
                        "Latitude" :...,
                        "Longitude":...
                     }
                  },
                  "Network":
                  {
                      "Host": "...",
                      "Port": ...,
                      "Protocol": "..."
                   }
                },
                "Mode": "..."
            }
            "Name": "...",
```

```
                "Serial": "..."
        }
        "EventCode": ...,
        "EventID": "{...}",
        "EventInfo":
        "EventTime": ...,
        "EventTriggerTime": ...,
        "State": "..."
        },
        "Filter":
        {
                "FuzzySearch": ...,
                "Params": "...",
                "Pattern": "..."
        },
        "ID": "{...}",
        "StartTime": ...,
        "Status": "..."
}
```

14.77 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 thesearch range
Filter	StorageEventsRequestFilter	(optional) Additional filter parameters
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 matchany 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 beggining of thesearch range

Pseudo code

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

14.78 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 similiarmatches 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": "..."
}
```

14.79 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 storagedevice
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 storagedevice
Total	int64	Total number of bytes available on the used storage device

Pseudo code

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


Adaptive Recognition **America** Adaptive Recognition **Nordic**
Adaptive Recognition **Hungary** Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.80 SUPPORTEDDETECTORS

> **See also:** Analytics/GetSupportedDetectors

Structure

Parameter	Type	Description
DetectorTypes	List/DetectorTypeInfo	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":
    {
        "DetectorClass": "...",
        "InstanceCount": ...,
        "InstanceLimit": ...,
        "Version": ...
    }
}
```


Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.81 SYSTEMSETTINGS

> **Inherited by:** SystemSettingsResponse
>
> **See also:** System/GetDevice, System/SetDevice

Structure

Parameter	Type	Description
Description	string	User-specified description
Name	string	User-specified name

Pseudo code

```
{
    "Description": "...",
    "Name": "..."
}
```


Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.82 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 asProductName.
ProductName	string	Name of the product design
ProductSubclass	string	Subclass of the lineup identifying a specific use-case
RequiredFirmwareVersion	string	Minimum firmware version in x.x.x.x format that this device acceptswhen a new firmware is uploaded
Serial	string	Unique device serial number

Pseudo code

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

14.83 SYSTEMSETTINGSMODULE

Inherited by: ModuleAnalytics, ModuleIO, ModuleMedia

Structure

Parameter	Type	Description

Pseudo code

```
{

}
```


Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.84 SYSTEMSETTINGSRESPONSE

See also: System/GetDevice

Structure

Parameter	Type	Description
Description	string	User-specified description
Device	SystemSettingsDevice	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 similiarfeatures
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 specificuse-case
RequiredFirmwareVersion	string	Minimum firmware version in x.x.x.x format that this device accepts when a new firmwareis uploaded
Serial	string	Unique device serial number
InstanceId	int64	Unique ID that changes every time the systemrestarts
Modules	SystemSettingsModule	List of module specific entries that describeeach module's capabilities
Name	string	User-specified name
Uptime	int64	Elapsed milliseconds since the system started

Pseudo code

```
{
	"Description": "...",
	"Device":
	{
		"Description": "...",
		"FirmwareVersion": "...",
		"ProductClass": "...",
		"ProductDisplayName": "...",
		"ProductName": "...",
		"ProductSubclass": "...",
		"RequiredFirmwareVersion": "...",
		"Serial": "..."
	},
	"InstanceId": ...,
	"Modules":
	{

	},
	"Name": "...",
	"Uptime": ...
}
```

14.85 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 **System/RunTest** will return if no exceptions arethrown
ThrowException	bool	If this field is set to true the response to **System/RunTest** will be an exception

Pseudo code

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


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.86 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": "..."
}
```

14.87 TIMESETTINGS

Device time settings

See also: System/GetTime, System/SetTime

Structure

Parameter	Type	Description
Datetime	string	Timestamp in the selected timezone as a formatted string (YYYY-MM-DD hh:mm:ss.sss)
Timestamp	int64	Current wall clock timestamp on the device (UTC)
Timezone		
Code	string	Abbreviated name of the timezone if available
Name	string	Name of the selected timezone
Offset	int64	Current offset of time from UTC in seconds

Pseudo code

```
{
    "Datetime": "...",
    "Timestamp": ...,
    "Timezone": ...
    {
        "Code": "...",
        "Name": "...",
        "Offset": ...
    }
}
```

14.88 TIMEZONELIST

List of supported timezone configurations

Inherited by: System/GetTimezones

Structure

Parameter	Type	Description
Timezones	List/string	List of timezone names available on the device

Pseudo code

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


Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.89 TIMEZONESETTINGS

Device timezone settings

Inherited by: System/GetTimezone, System/SetTimezone

Structure

Parameter	Type	Description
DayLightEnabled	bool	This property is unused and deprecated
Timezone	string	v

Pseudo code

```
{
    "DayLightEnabled": ...,
    "Timezone": "..."
}
```

14.90 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": "..."
}
```

14.91 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

Pseudo code

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


Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.92 TRACKINGDETECTORCONFIGURATION

Inherited by: DetectorConfigurationEmergencyLane, DetectorConfigurationForbiddenZone, DetectorConfigurationLane, DetectorConfigurationRedStop, DetectorConfigurationStopViolation, DetectorConfigurationStoppedObject, DetectorConfigurationTrafficLine, DetectorConfigurationUTurn, DetectorConfigurationWhiteLineViolation, DetectorConfigurationWrongTurn, DetectorConfigurationWrongWay

Structure

Parameter	Type	Description
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Center	bool	Set to true to operate using an object's center point instead of all corners
Class	string	Detector type name
Confidence	int8	Minimum allowed object confidence when [strong]ConfidenceEnabled[/strong] is set to true
ConfidenceEnabled	bool	Set to true to use a confidence treshold for object monitoring
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.
ObjectTypes	List/string	List of object types that are monitored or empty list for all types
RestoreDelayMs	int64	[em]unused[/em]
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": ...
}
```

14.93 USER

All user account information

See also: System/AddUser, System/ModifyUser

Structure

Parameter	Type	Description
Password	string	User password (write only)
	string	User name
Role	string	User role

Pseudo code

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

14.94 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": "..."
}
```

14.95 USERINFO

User account information

> **Inherited by:** User
>
> **See also:** System/AddUser, System/GetCurrentUser, System/ModifyUser

Structure

Parameter	Type	Description
Role	string	User role
Name	string	User name

Pseudo code

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


Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

14.96 USERS

Contains information about all user accounts available on the device

> **See also:** System/GetUsers

Structure

Parameter	Type	Description
Users	User	List of user accounts
Name	string	User name
Role	string	User role
Password	string	User password (write only)

Pseudo code

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


Adaptive Recognition America
Adaptive Recognition Hungary
Adaptive Recognition Nordic
Adaptive Recognition Singapore
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

CONTACT INFORMATION

Headquarters:
Adaptive Recognition, Hungary Inc.
Alkotás utca 41 HU
1123 Budapest Hungary
Web: adaptiverecognition.com

Service Address:
Adaptive Recognition, Hungary Inc.
Ipari Park HRSZ1113/1 HU
2074 Perbál Hungary
Web: 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) 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)!

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.