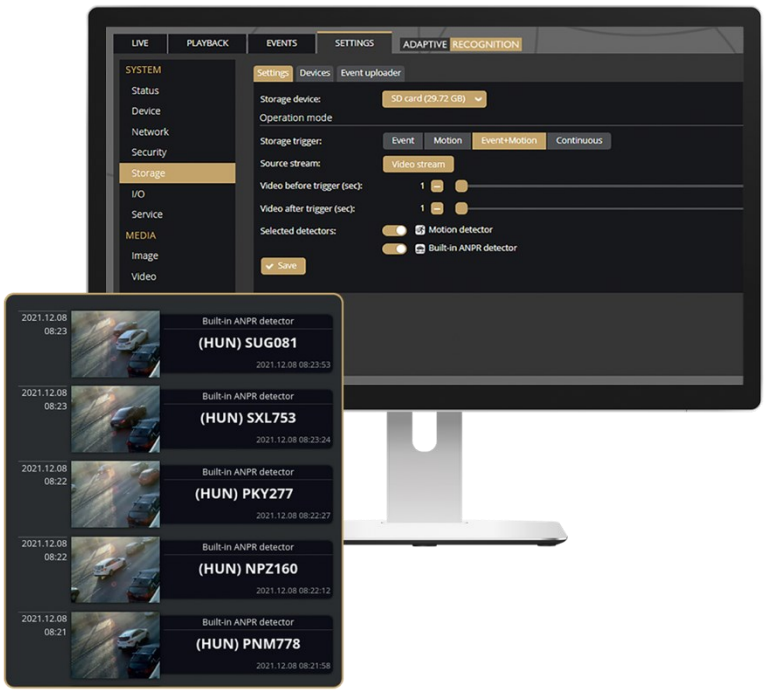
ADAPTIVE **RECOGNITION**

Carmen® Nano

User Manual



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

RELEASE DATE
31.03.2025.

Carmen® Nano

USER MANUAL

with Installation Checklist and other practical tips

Document version: 2025.03.31.
Firmware version: 1.6.0.45

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

1. OVERVIEW

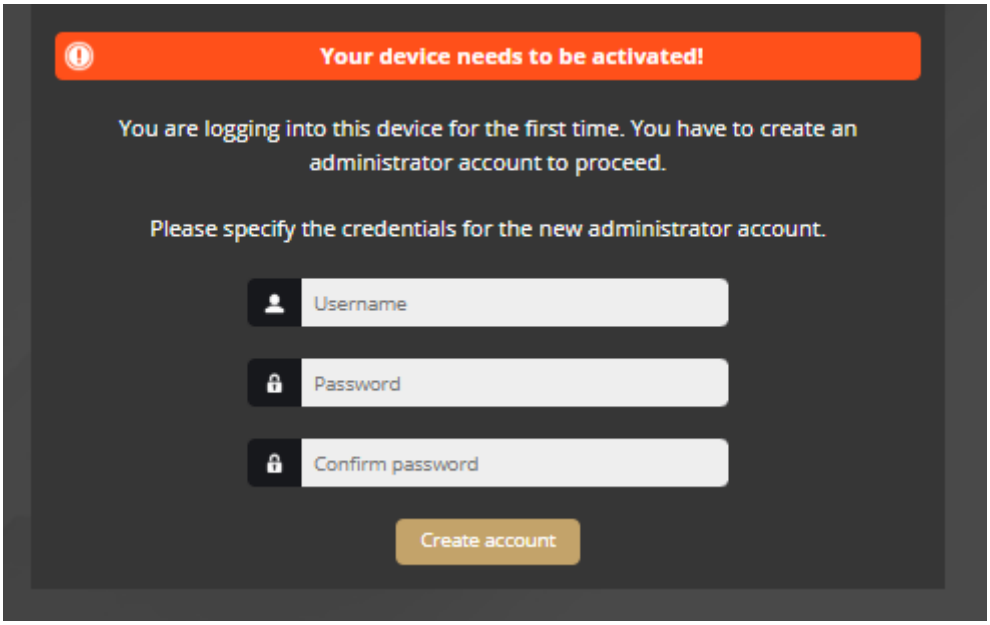
The Carmen Nano has its own web interface through which you can access the settings, the **LIVE**, the **PLAYBACK**, the **EVENTS** and the **SETTINGS** interfaces.

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

Accessing the web interface:

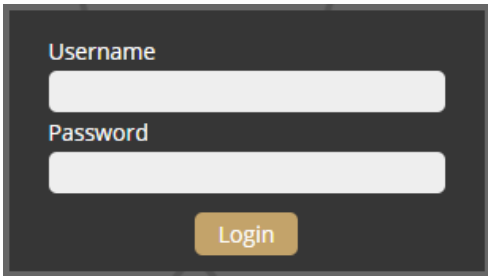
1. Start a browser and enter the device IP address into the address bar of the browser.

2. You are logging into this device for the first time. You have to create an administrator account to proceed.
   Please specify the credentials for the new administrator account.



   If the device has firmware earlier than firmware version 1.6, this step is skipped. You will also be welcomed at the login screen.
   The default user account is the following:
   - **Username**: admin
   - **Password**: admin

3. Type the username and the password on the displayed login interface and click on [Login].



4. After signing in, the web interface of the camera is displayed on which the required settings can be performed.

2. AR DEVICE TOOL

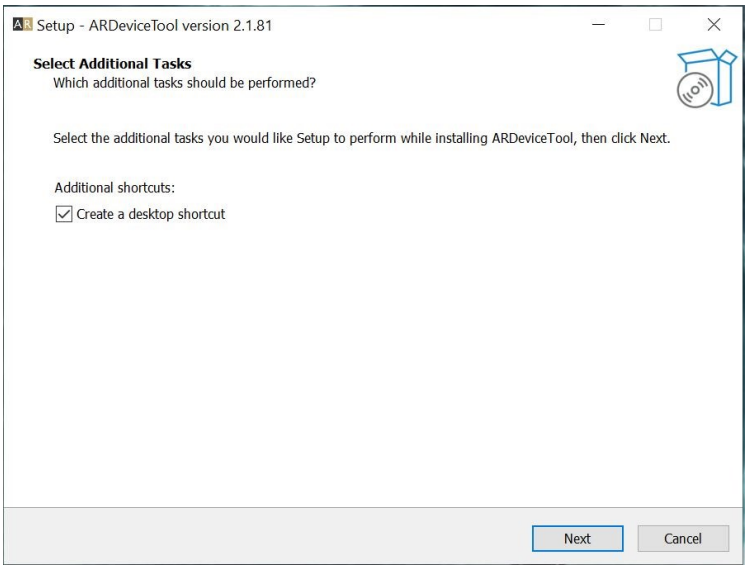
With the AR Device Tool, you can discover Einar or Visus cameras, Carmen Box; Carmen Nano or Enforce Box devices on the local network. You can upload Firmware, License and Engine files to these cameras/devices.

The AR Device Tool also finds all AR cameras (Vidar, MicroCAM, SmartCAM) via ONVIF protocol.

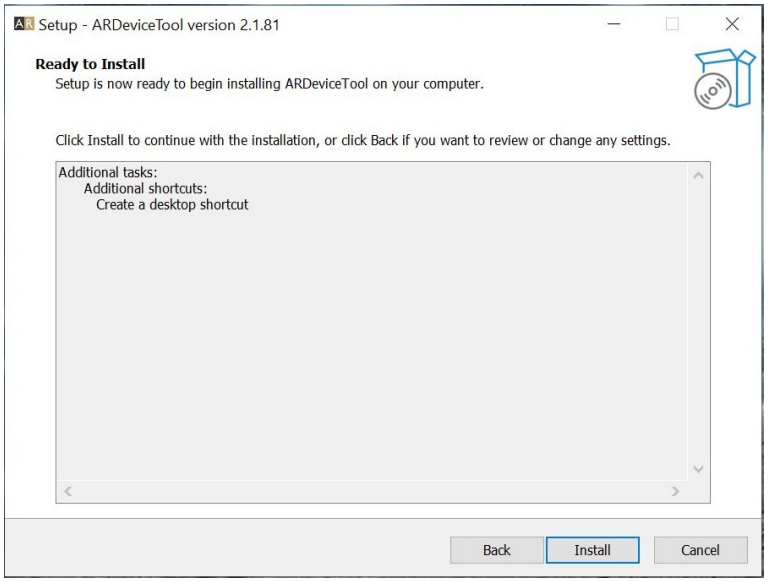
Download the program here: AR DeviceTool.

2.1. AR DEVICE TOOL SETUP

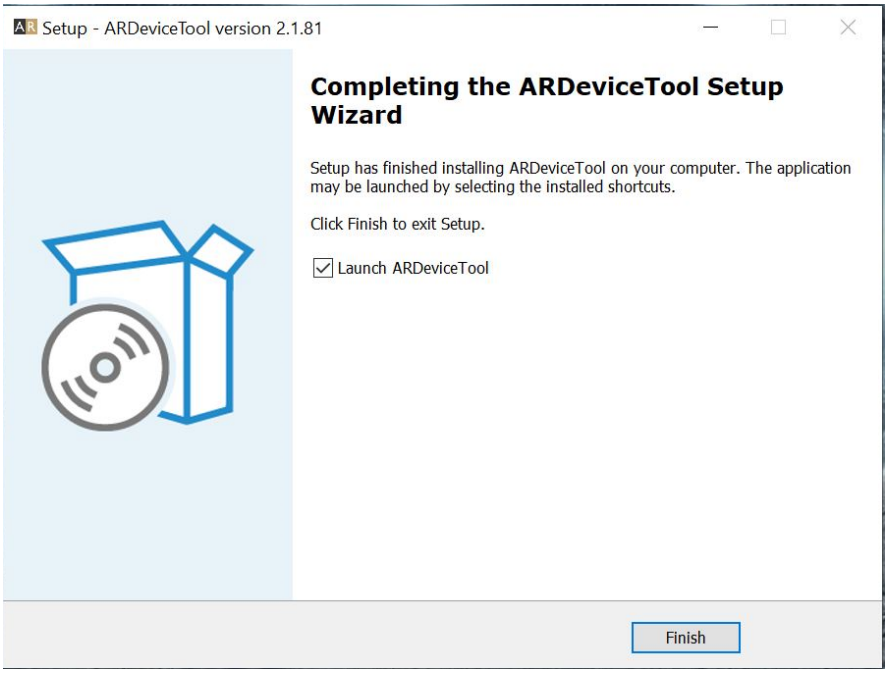
After downloading, start the installation file. Select the desktop shortcut you want to use and click Next.



The next step is to start the installation by clicking on the Install button.



Select if you want to start ARDeviceTool and click Finish.

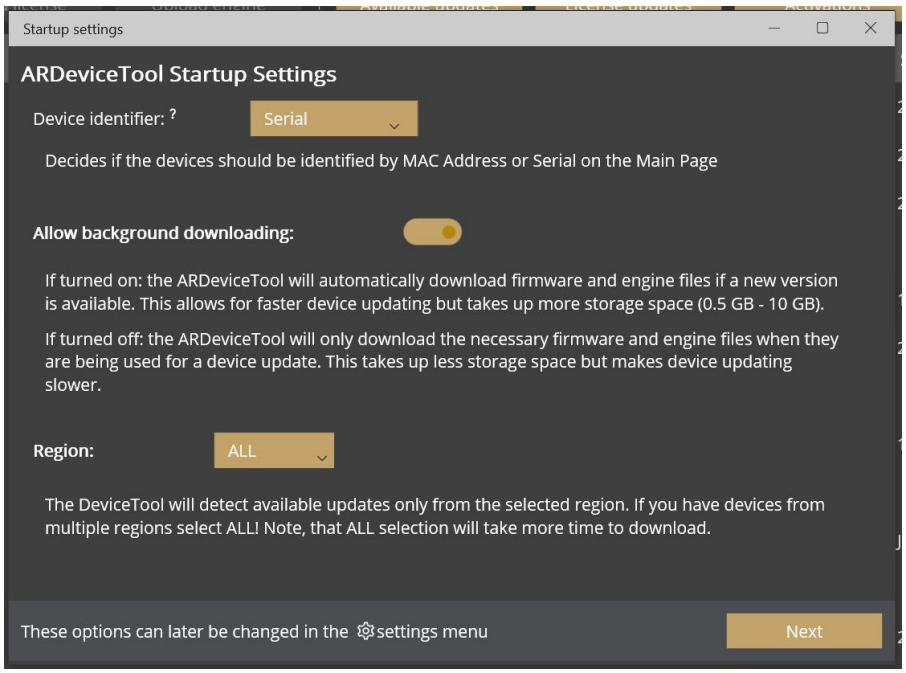


After installation, enable the Device Tool to access networks in the firewall settings. DeviceTool communicates with cameras via HTTP (default port 80).

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

2.2. AR DEVICE TOOL STARTUP SETTINGS

When you start AR DeviceTool for the first time, you can make the initial settings.



Device Identifier: Decides is the devices should be identified by MAC Address or Serial on the Main page.

Allow background:

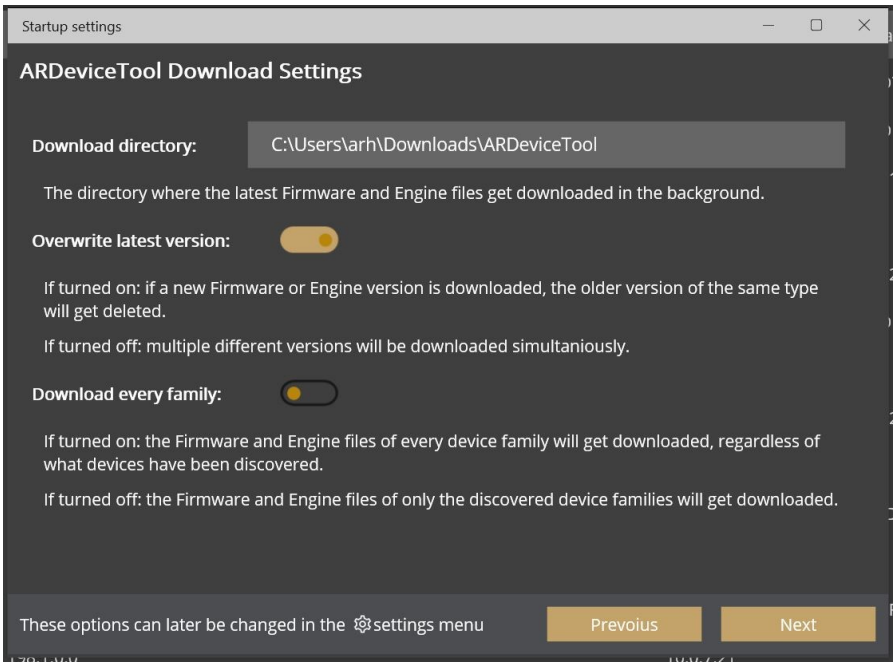
If turned on: the AR DeviceTool will automatically download firmware and engine files if a new version is available. This allows for faster device updating but takes up more storage space (0.5 – 10 GB)

If turned off: the AR DeviceTool will only download the necessary firmware and engine files when they are being used for a device update. This takes up less storage space but makes device updating slower.

Region: The DeviceTool will detect available updates only from the selected region. If you have devices from multiple regions select ALL! Note, that ALL selection will take more time download.

When you have made the settings, click Next.

These options can later be changed in the settings menu.



Download directory: The directory where the latest firmware and engine files det downloaded in the background.

Overwrite latest version:

If turned on: if a new firmware or engine version is downloaded, the older version of the same type will get deleted.

If turned off: multiple different versions will be downloaded simultaniously.

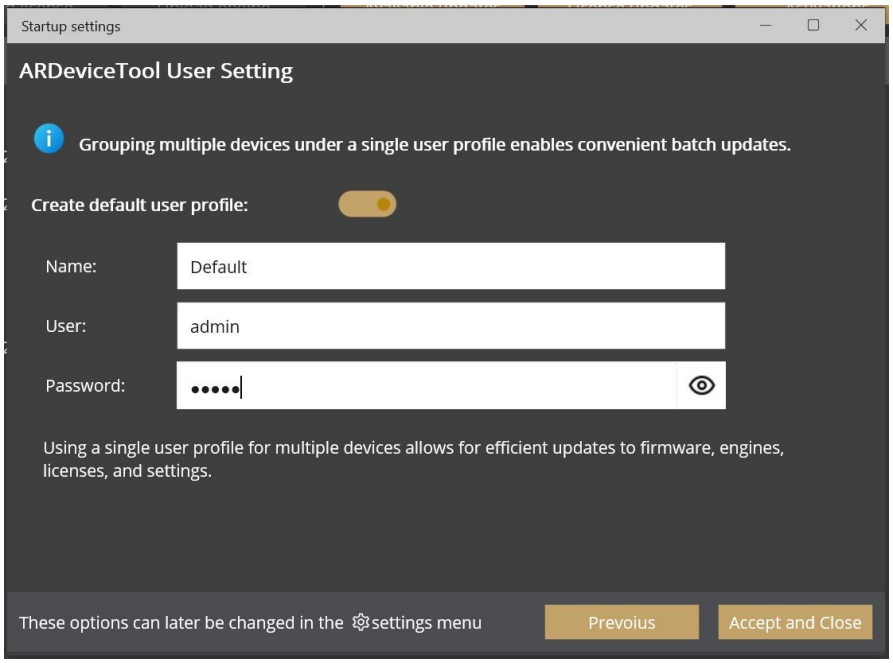
Download every family:

If turned on: the firmware and engine files of every device family will get downloaded, regardless of what devices have been discovered.

If turned off: the firmware and engine files of only discovered device families will get downloaded.

When you have made the settings, click Next.

These options can later be changed in the settings menu.



Create default user profile: Fill in the Name, User and Password fields.

Using a single user profile for multiple devices allows for efficient updates to firmware, engine, license, and settings.

When you have made the settings, click Accept and Close.

These options can later be changed in the settings menu.

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

2.3. FINDING CAMERAS

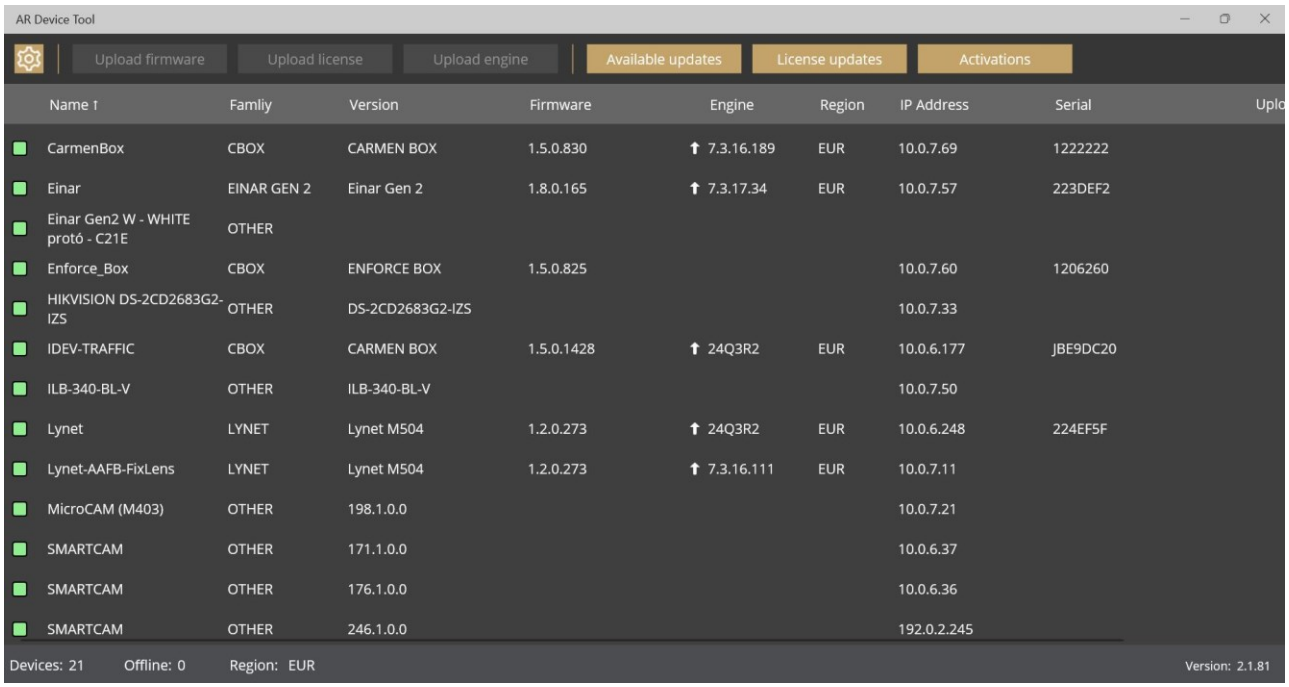
The program lists the AR devices/cameras detected on the local network if the devices/cameras are in the same network segment as the computer. The device/camera name, product family name, type, firmware version, IP address, MAC address and brief information about the current upload process will be displayed.

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, an up arrow will appear in the firmware/engine column.

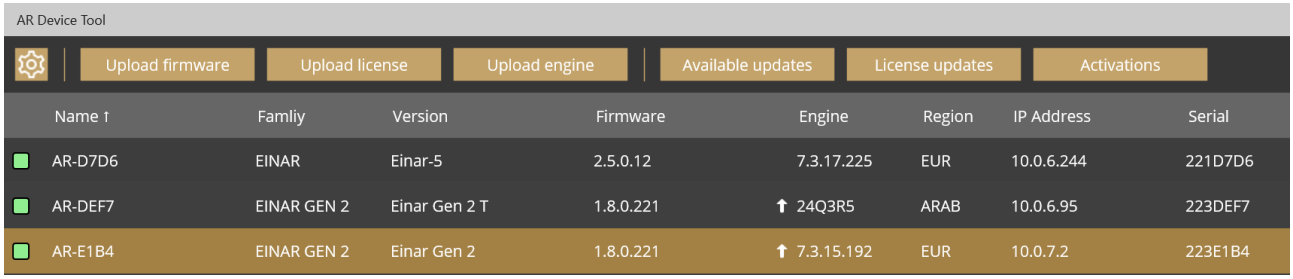
Double-click on the selected device/camera to open its web interface in the default browser.



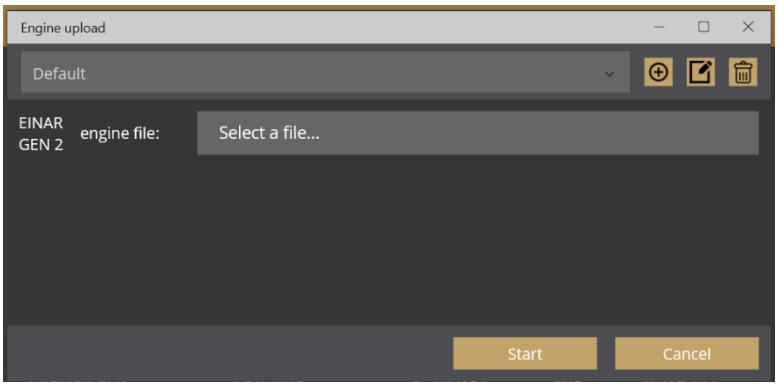
For Vidar, MicroCAM, SmartCam cameras only Name, Version and IP Address data can be detected by AR DeviceTool, these cameras can not be upgraded using AR DeviceTool.

2.4. FIRMWARE, LICENSE, AND ENGINE UPLOAD - MANUALLY

In addition to find devices/cameras, you can also use the AR 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.



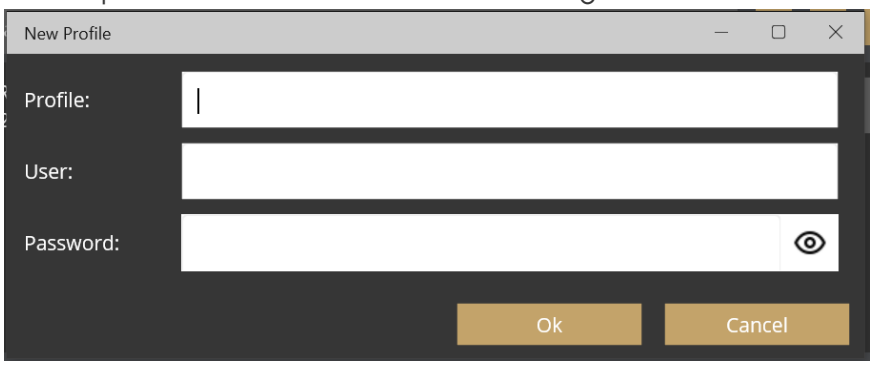
Select the device(s)/camera(s) you want to update and press the **[Upload firmware]**, **[Upload License]** or **[Upload engine]** buttons that become active.
In the window that appears, enter the username and password to access the camera, select the file you want to upload and click **[Start]** to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



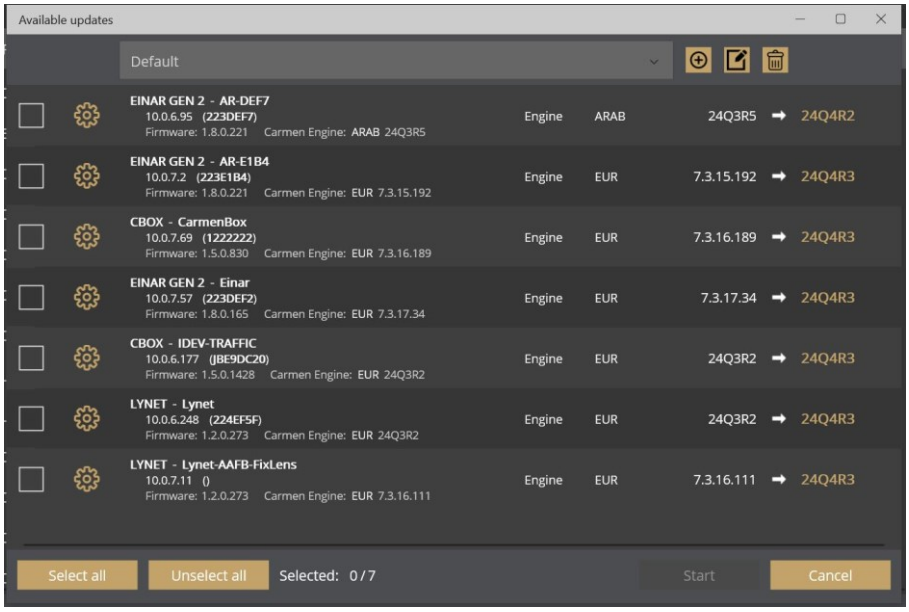
If you want to save the username and password to access the device(s)/camera(s), you can create user profiles. This way, you don't have to enter credentials before each upload.

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.



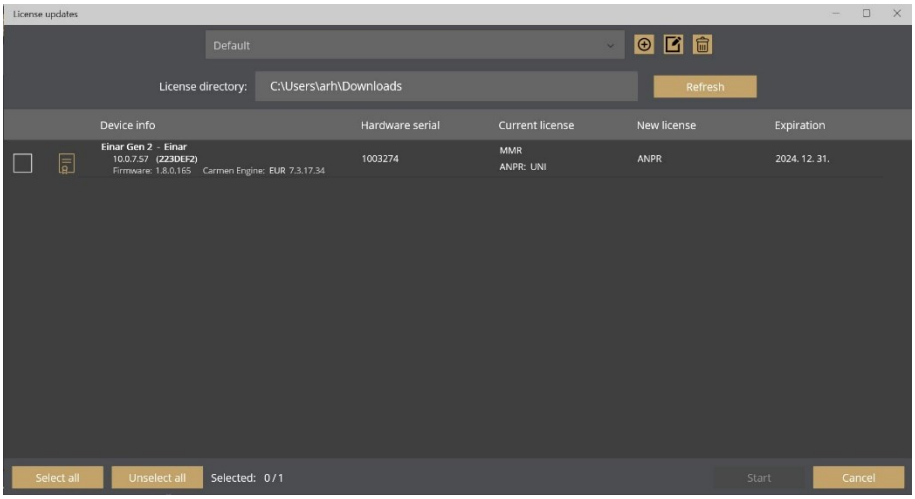
2.5. AVAILABLE UPDATES



AR DeviceTool downloads the appropriate firmware and/or engine from a central server, and uploads it to the selected device(s)/camera(s). A new folder will be created in your Download folder: ArDeviceTool.

Select the cameras you want to update, select the user associated with the cameras, click Start, and then confirm the upload. The cameras will restart during the update.

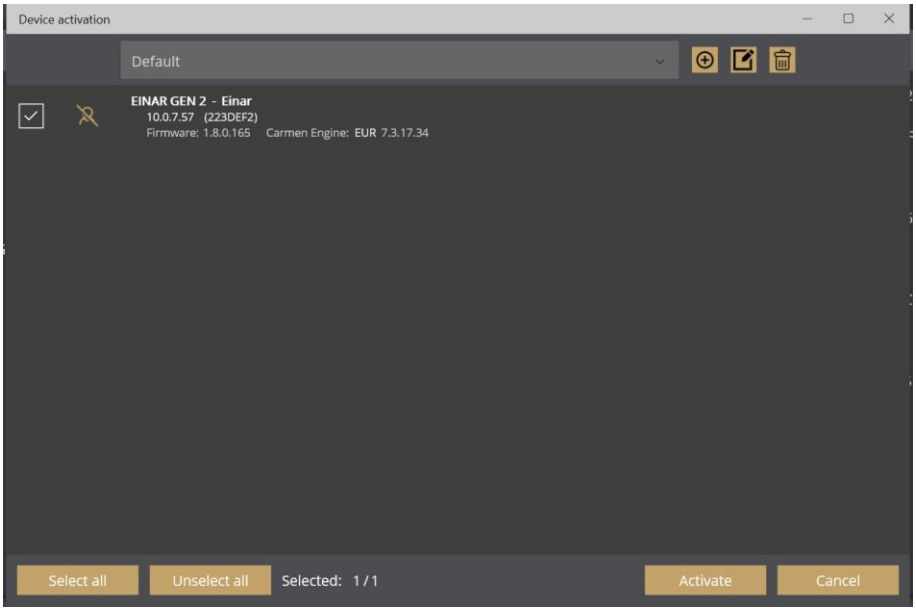
2.6. LICENSE UPDATES



Copy the license to a folder (unzipped) and ARDeviceTool will identify which camera it belongs to. Multiple licenses can be copied to the License directory folder.

Select the cameras you want to update, select the user associated with the cameras, click Start, and then confirm the upload. The cameras will restart during the update.

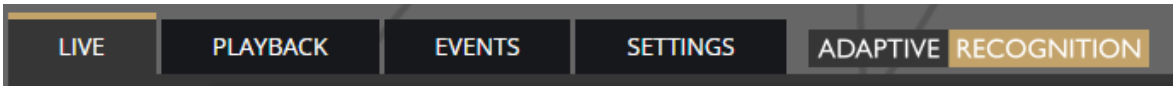
2.7. DEVICE ACTIVATION



Select the user you want to use, select the cameras you want to activate, then click on the Activate button. This will activate the selected cameras with this user. (This will be the default user.)

3. OVERVIEW OF THE WEB INTERFACE

The following menu items are available on the web interface:



1. LIVE

Shows a live view of the connected camera streams.

2. PLAYBACK

Browse recordings on the configured storage device.

3. EVENTS

Browse the recorded events on the configured storage device.

4. SETTINGS

Under this menu, you can access the following options:

SYSTEM

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

MEDIA

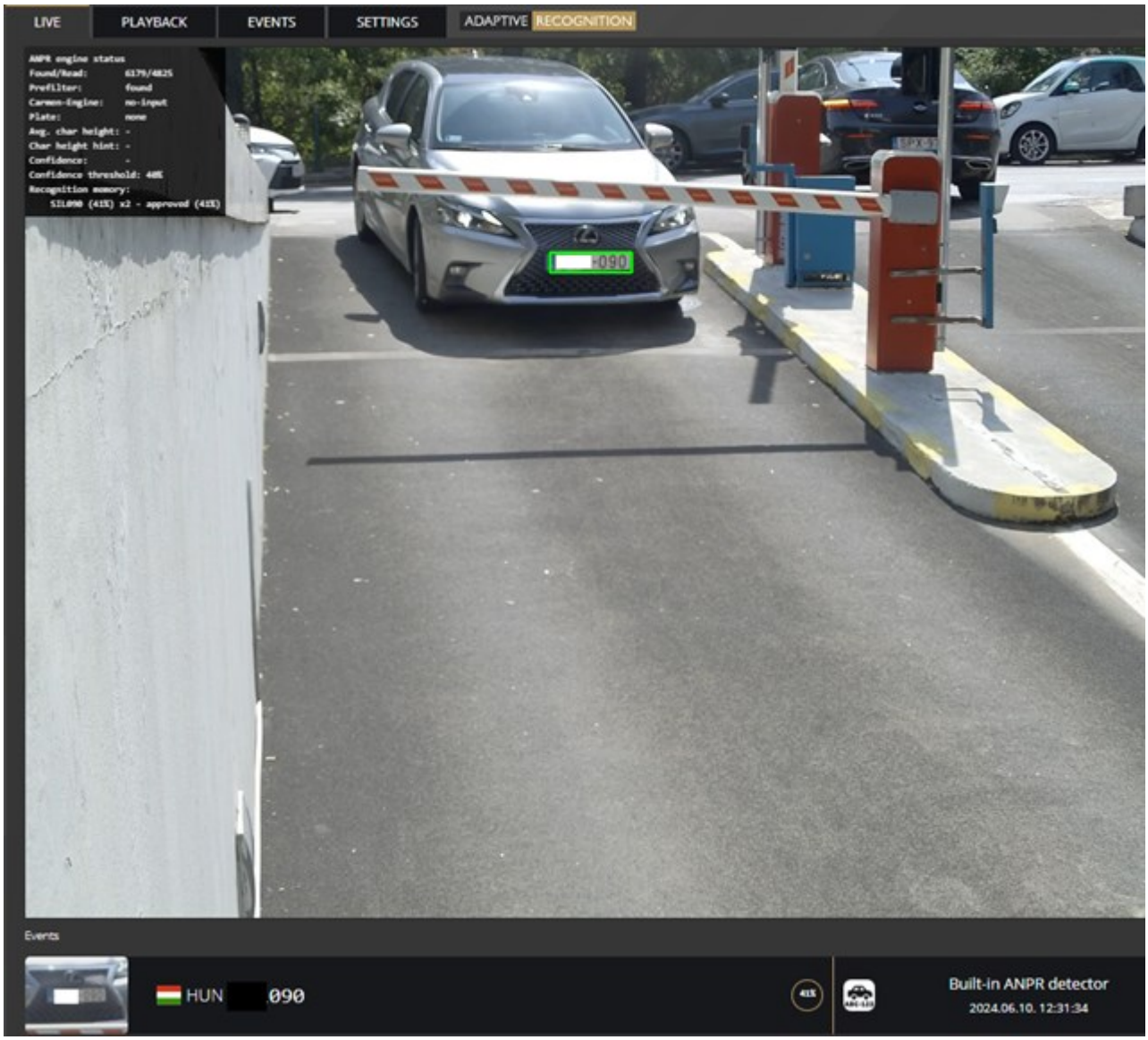
- Video

ANALYTICS

- Settings
- Detectors
- Uploaders

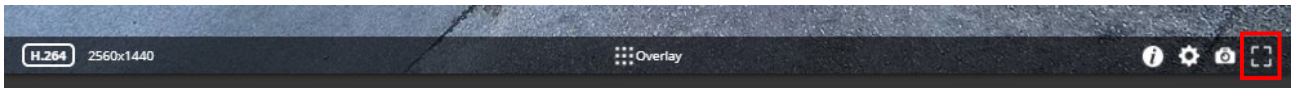
4. LIVE

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



4.1. FULL-SCREEN MODE

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



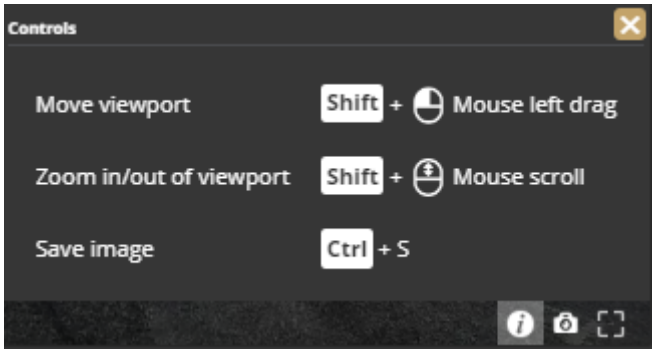
To exit from the full-screen mode, press the **ESC** keyboard key or click on the icon mentioned above.

4.2. SAVING IMAGE

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

4.3. CONTROLS

Next to the **[Streams]** icon is the **[Controls]** button. It brings up keyboard shortcuts on how to use and navigate the video feed.

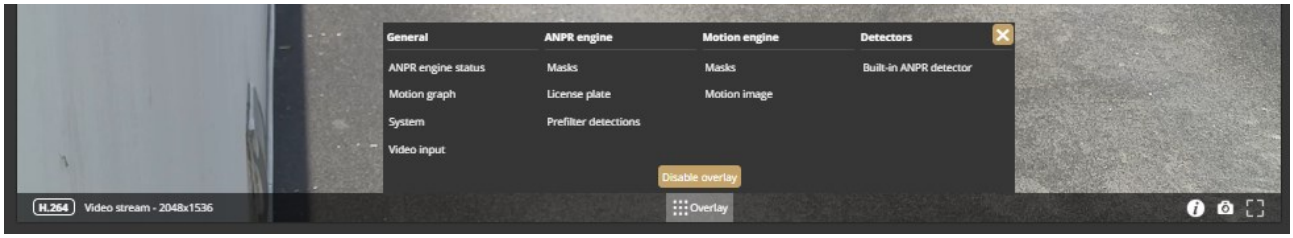


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

4.4. OVERLAY

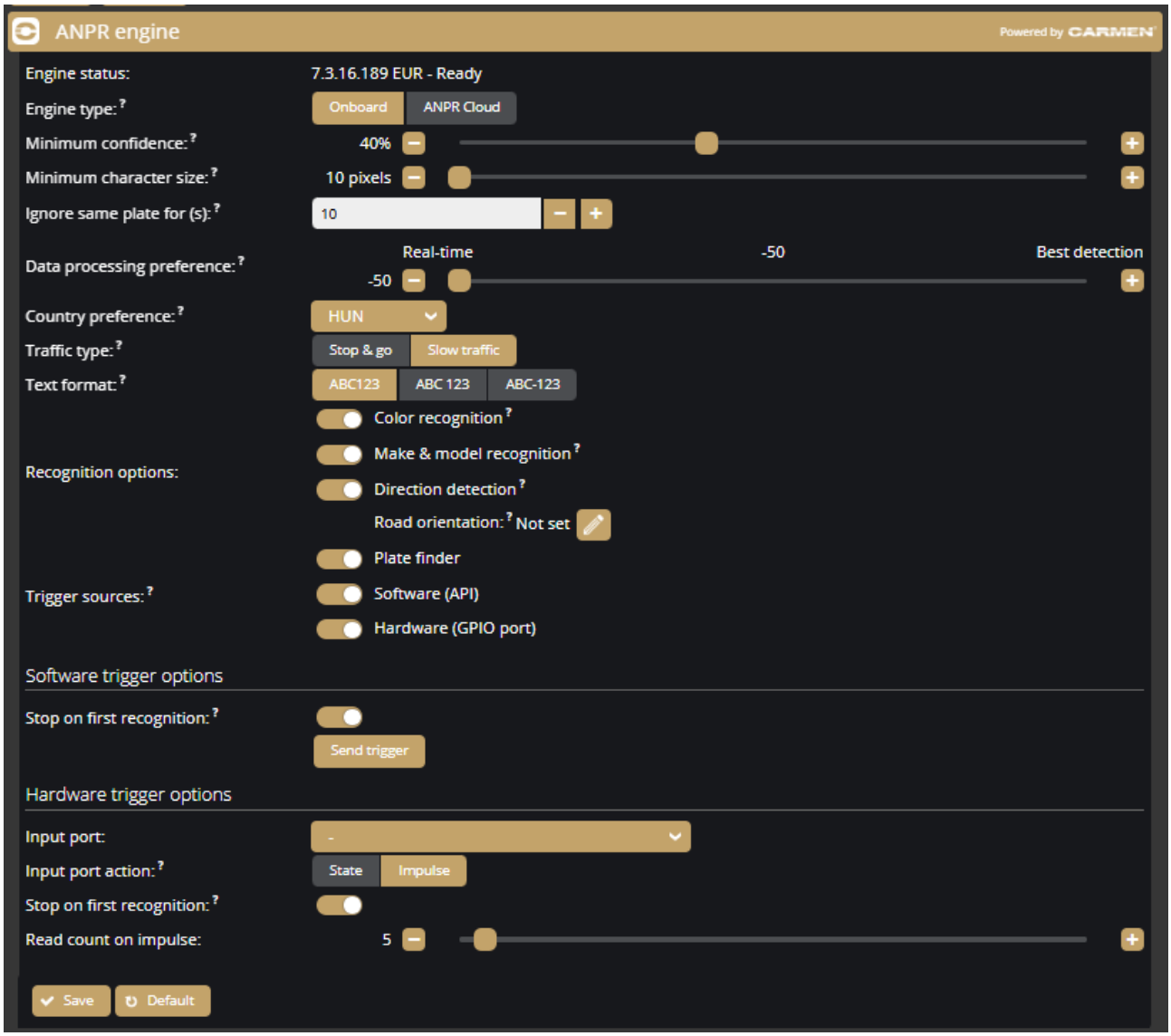
In the middle, at the bottom of the window, is the **[Overlay]** button. With it, you can turn on/off the OSD, and you can view the masks of the applied detectors, image information, motion data, etc.

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, e.g. ***{Media / Video}***. The OSD layers come in handy for observing the internal workflow of the device, setting up the device or troubleshooting.



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

Found/Read: Number of images on which the device 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
{



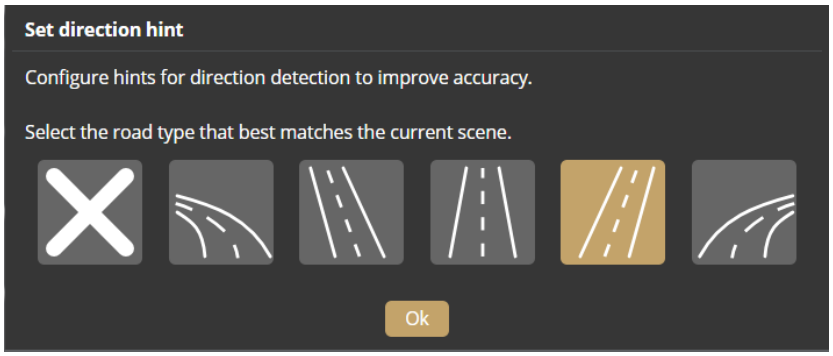
In the case of **Onboard ANPR**, you can adjust the primary functions of the ANPR engine:

- **Engine status:** It shows the current ANPR 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 ***{****Hiba! A hivatkozási forrás nem található.****}***.
- **Minimum character size:** License plates with character size below this threshold are discarded.
- **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.

- **Data processing preference:** Configure how the engine processes license plate data.
  Setting the slider towards Real-time will decrease the delay between a vehicle appearing and an event being emitted but also decreases the accuracy of the license plate data. Recommended for quick license plate validation when device is used for controlling vehicle entry points.
  Setting the slider towards Best detection will increase the license plate data precision including vehicle direction and vehicle speed but will dramatically increase the delay between a vehicle appearing and an event being emitted as the engine will spend more time making sure all data are sufficiently validated.\ Recommended when data accuracy is important and events are not used for live triggers but processed later or in bulk.
- **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:** Type of traffic the device processes. Some engines contain more than one recognition possibility. If the uploaded engine supports „Quick" recognition mode, you can select it as „SlowTraffic" mode.
- **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.
- **Text format:** By default license plate characters are returned without spacing. Depending on the selected option symbol groups on a license plate may be separated by whitespaces and/or hypens.
- **Color recognition:** When color recognition is active, the device will attempt to identify the plate and/or vehicle colors. This may slow down the speed of recognition.
- **Make & Model recognition:** When make and model recognition is active, the device will attempt to identify the make, model and color of the vehicle. This may slow down the speed of recognition.

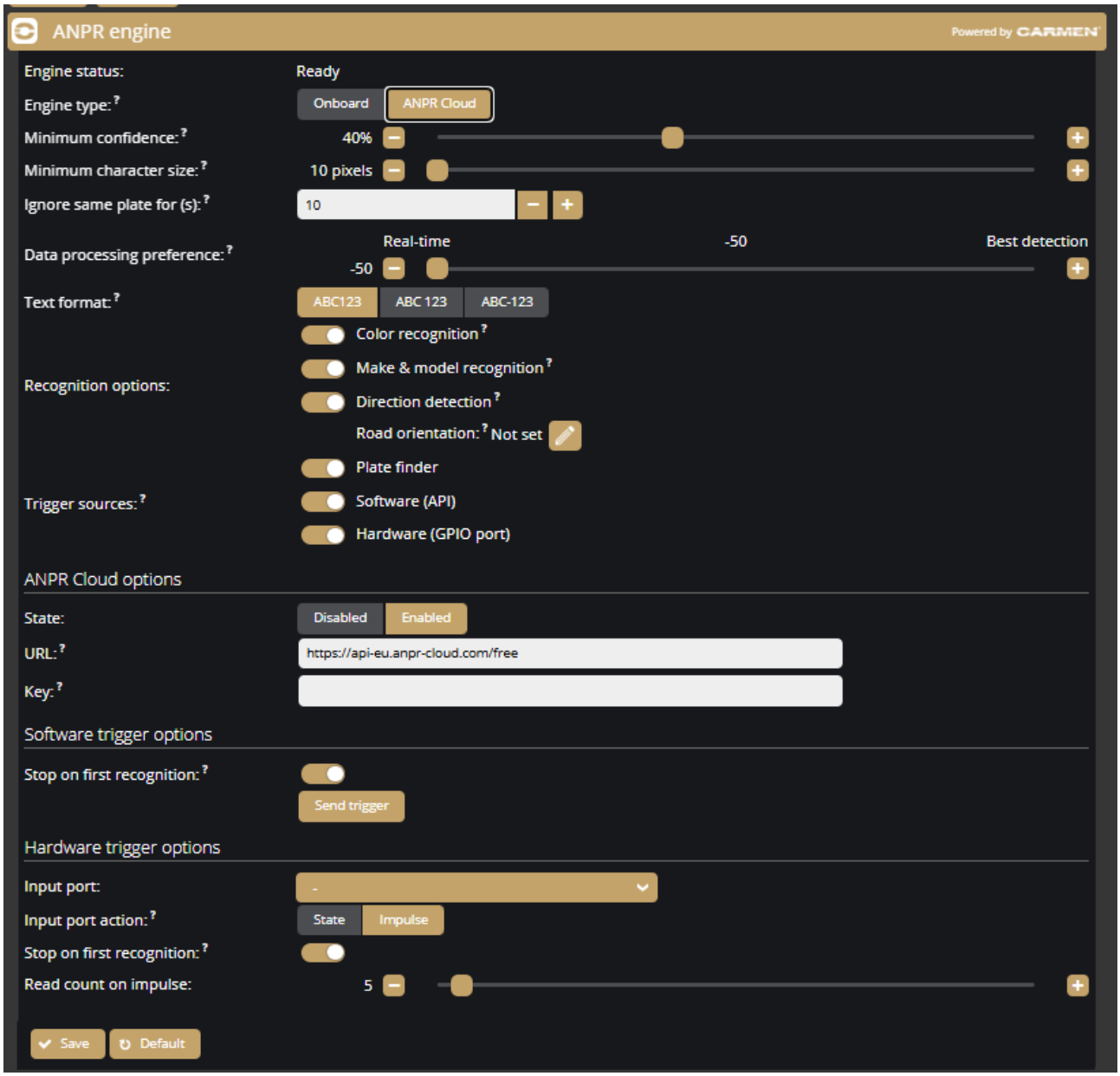
Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

- **Direction detection:** Attempt to detect direction of the vehicle movement by license plate.
  - **Road orientation:** Click on the edit button and set the road type to improve direction detection accuracy.



Important!

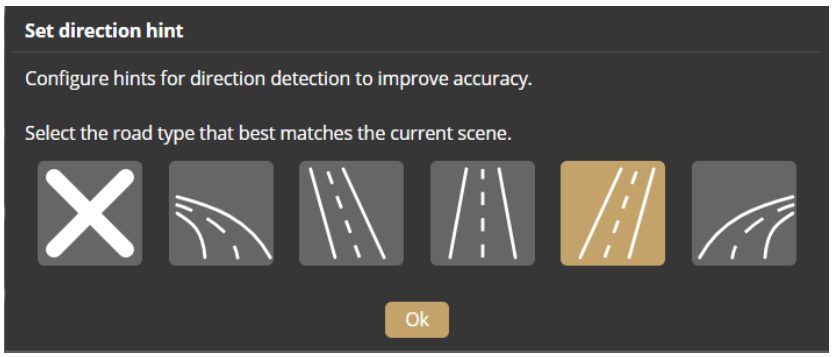
Direction detection is only performed if the same license plate number has been read at least twice by the camera and the height of the license plate number character has changed by at least 2 pixels in the image.


ANPR engine
Powered by CARMEN
Engine status:
Ready
Engine type:
Onboard
ANPR Cloud
Minimum confidence:
40%
Minimum character size:
10 pixels
Ignore same plate for (s):
10
Data processing preference:
Real-time
-50
Best detection
-50
Text format:
ABC123
ABC 123
ABC-123
Recognition options:
Color recognition
Make & model recognition
Direction detection
Road orientation:
Not set
Plate finder
Trigger sources:
Software (API)
Hardware (GPIO port)
ANPR Cloud options
State:
Disabled
Enabled
URL:
https://api-eu.anpr-cloud.com/free
Key:
Software trigger options
Stop on first recognition:
Send trigger
Hardware trigger options
Input port:
Input port action:
State
Impulse
Stop on first recognition:
Read count on impulse:
5
Save
Default

In the case of **Carmen Cloud**, the license plate recognition is not processed in the device. Instead, images with license plates are selected in the device 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/.

- **Engine status:** It shows the current ANPR 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 ***{****Hiba! A hivatkozási forrás nem található.****}***.
- **Minimum character size:** License plates with character size below this threshold are discarded.
- **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.
- **Data processing preference:** Configure how the engine processes license plate data.
  Setting the slider towards Real-time will decrease the delay between a vehicle appearing and an event being emitted but also decreases the accuracy of the license plate data. Recommended for quick license plate validation when device is used for controlling vehicle entry points.
  Setting the slider towards Best detection will increase the license plate data precision including vehicle direction and vehicle speed but will dramatically increase the delay between a vehicle appearing and an event being emitted as the engine will spend more time making sure all data are sufficiently validated.\ Recommended when data accuracy is important and events are not used for live triggers but processed later or in bulk.
- **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:** Type of traffic the device processes. Some engines contain more than one recognition possibility. If the uploaded engine supports „Quick" recognition mode, you can select it as „SlowTraffic" mode.
- **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.
- **Text format:** By default license plate characters are returned without spacing. Depending on the selected option symbol groups on a license plate may be separated by whitespaces and/or hypens.
- **Color recognition:** When color recognition is active, the device will attempt to identify the plate and/or vehicle colors. This may slow down the speed of recognition.
- **Make & Model recognition:** When make and model recognition is active, the device will attempt to identify the make, model and color of the vehicle. This may slow down the speed of recognition.
- **Direction detection:** Attempt to detect direction of the vehicle movement by license plate.
    - **Road orientation:** Click on the edit button and set the road type to improve direction detection accuracy.

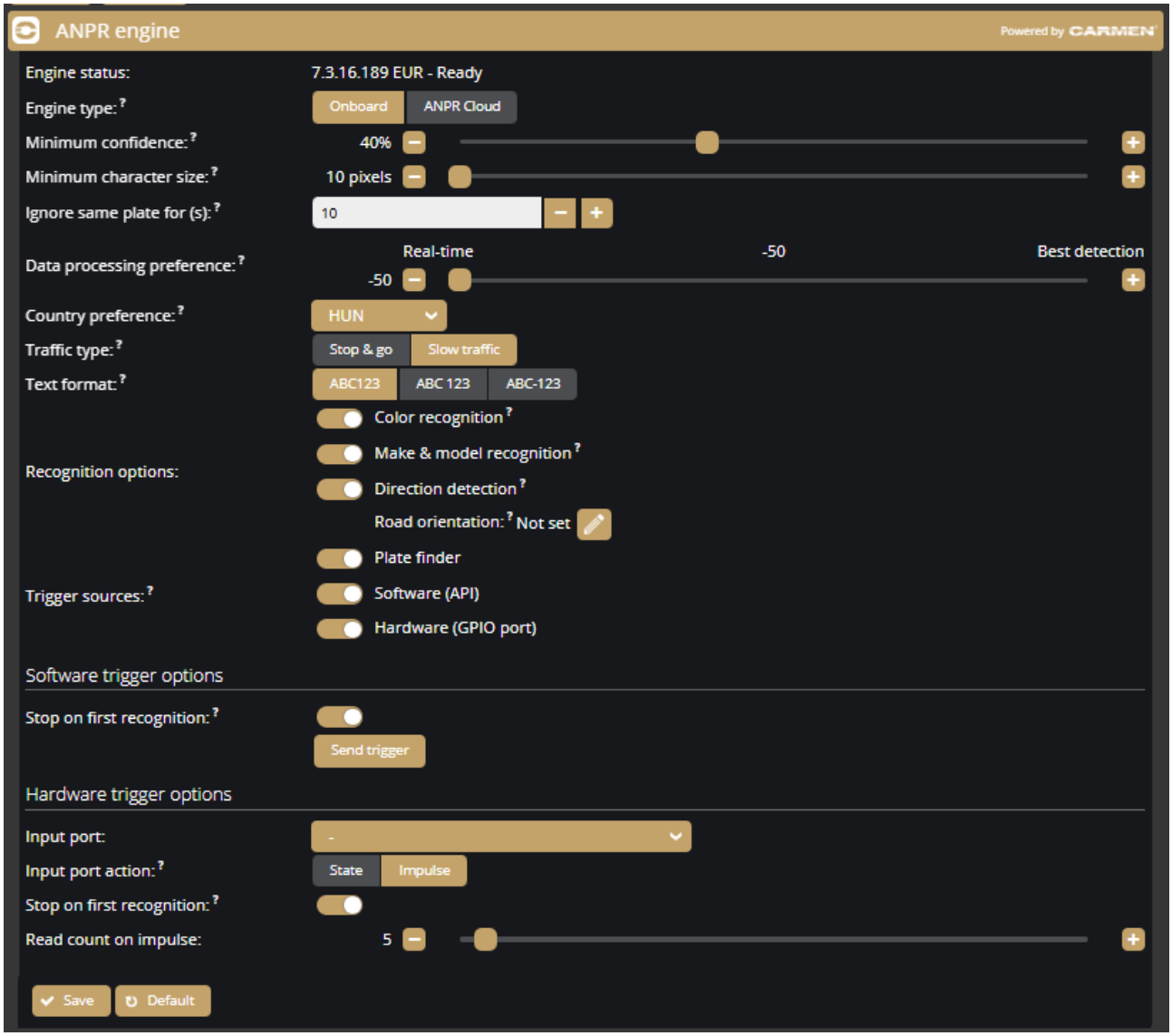


- **ANPR Cloud URL:** Paste the URL you received when subscribing to the field.
- **ANPR Cloud key:** Enter the individual key of the subscription

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

}. 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 **{**



In the case of **Onboard ANPR**, you can adjust the primary functions of the ANPR engine:

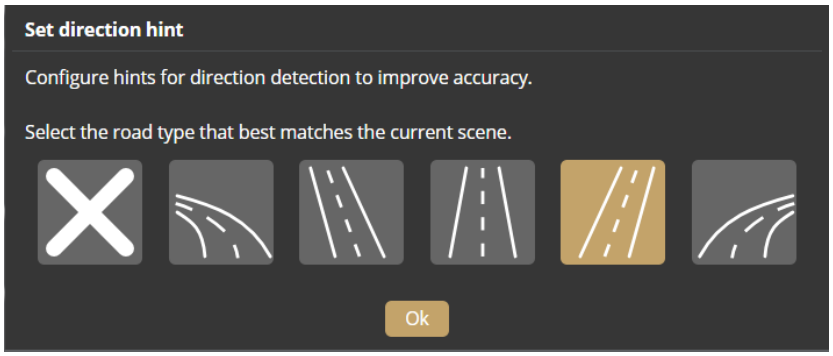
- **Engine status:** It shows the current ANPR version and region.
- **Minimum confidence:** Use the slider to define what reading confidence percentage should trigger an event. A low value potentially results in more reading errors. A high value potentially leads to missed reading cases. Therefore, 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 ***{****Hiba! A hivatkozási forrás nem található.****}***.

- **Minimum character size:** License plates with character size below this threshold are discarded.
- **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.

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

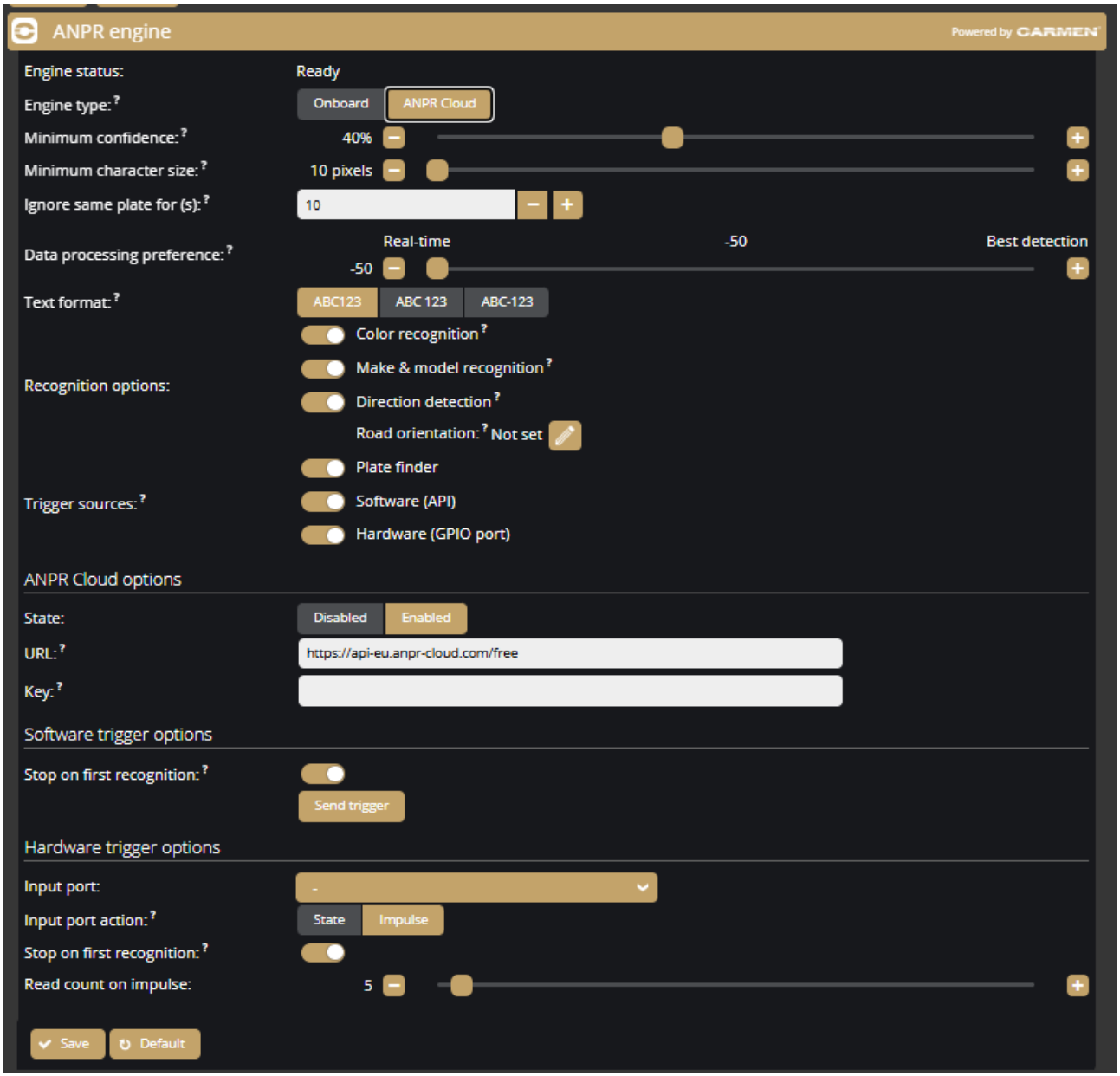
- **Data processing preference:** Configure how the engine processes license plate data.
  Setting the slider towards Real-time will decrease the delay between a vehicle appearing and an event being emitted but also decreases the accuracy of the license plate data. Recommended for quick license plate validation when device is used for controlling vehicle entry points.
  Setting the slider towards Best detection will increase the license plate data precision including vehicle direction and vehicle speed but will dramatically increase the delay between a vehicle appearing and an event being emitted as the engine will spend more time making sure all data are sufficiently validated.\ Recommended when data accuracy is important and events are not used for live triggers but processed later or in bulk.
- **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:** Type of traffic the device processes. Some engines contain more than one recognition possibility. If the uploaded engine supports „Quick" recognition mode, you can select it as „SlowTraffic" mode.
- **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.
- **Text format:** By default license plate characters are returned without spacing. Depending on the selected option symbol groups on a license plate may be separated by whitespaces and/or hypens.
- **Color recognition:** When color recognition is active, the device will attempt to identify the plate and/or vehicle colors. This may slow down the speed of recognition.
- **Make & Model recognition:** When make and model recognition is active, the device will attempt to identify the make, model and color of the vehicle. This may slow down the speed of recognition.

- **Direction detection:** Attempt to detect direction of the vehicle movement by license plate.
  - **Road orientation:** Click on the edit button and set the road type to improve direction detection accuracy.



Important!

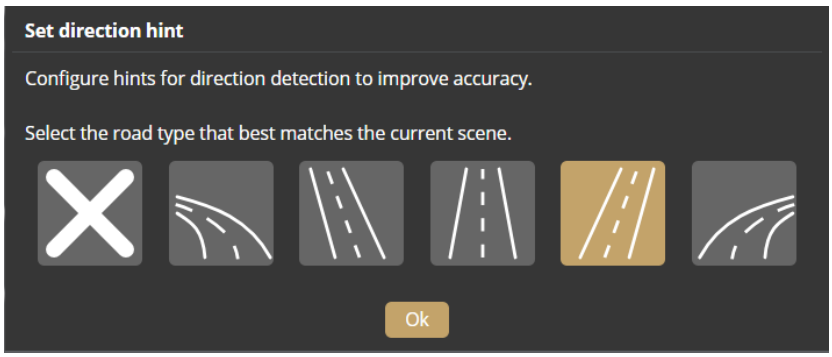
Direction detection is only performed if the same license plate number has been read at least twice by the camera and the height of the license plate number character has changed by at least 2 pixels in the image.


ANPR engine
Powered by CARMEN
Engine status:
Ready
Engine type:
Onboard
ANPR Cloud
Minimum confidence:
40%
Minimum character size:
10 pixels
Ignore same plate for (s):
10
Data processing preference:
Real-time
-50
Best detection
-50
Text format:
ABC123
ABC 123
ABC-123
Recognition options:
Color recognition
Make & model recognition
Direction detection
Road orientation:
Not set
Plate finder
Trigger sources:
Software (API)
Hardware (GPIO port)
ANPR Cloud options
State:
Disabled
Enabled
URL:
https://api-eu.anpr-cloud.com/free
Key:
Software trigger options
Stop on first recognition:
Send trigger
Hardware trigger options
Input port:
Input port action:
State
Impulse
Stop on first recognition:
Read count on impulse:
5
Save
Default

In the case of **Carmen Cloud**, the license plate recognition is not processed in the device. Instead, images with license plates are selected in the device 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/.

- **Engine status:** It shows the current ANPR 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 ***{****Hiba! A hivatkozási forrás nem található.****}***.
- **Minimum character size:** License plates with character size below this threshold are discarded.
- **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.
- **Data processing preference:** Configure how the engine processes license plate data.
  Setting the slider towards Real-time will decrease the delay between a vehicle appearing and an event being emitted but also decreases the accuracy of the license plate data. Recommended for quick license plate validation when device is used for controlling vehicle entry points.
  Setting the slider towards Best detection will increase the license plate data precision including vehicle direction and vehicle speed but will dramatically increase the delay between a vehicle appearing and an event being emitted as the engine will spend more time making sure all data are sufficiently validated.\ Recommended when data accuracy is important and events are not used for live triggers but processed later or in bulk.
- **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:** Type of traffic the device processes. Some engines contain more than one recognition possibility. If the uploaded engine supports „Quick" recognition mode, you can select it as „SlowTraffic" mode.
- **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.
- **Text format:** By default license plate characters are returned without spacing. Depending on the selected option symbol groups on a license plate may be separated by whitespaces and/or hypens.
- **Color recognition:** When color recognition is active, the device will attempt to identify the plate and/or vehicle colors. This may slow down the speed of recognition.
- **Make & Model recognition:** When make and model recognition is active, the device will attempt to identify the make, model and color of the vehicle. This may slow down the speed of recognition.
- **Direction detection:** Attempt to detect direction of the vehicle movement by license plate.
    - **Road orientation:** Click on the edit button and set the road type to improve direction detection accuracy.



- **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.

} "Ignore same plate for (s)".

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

4.5. EVENT PREVIEW

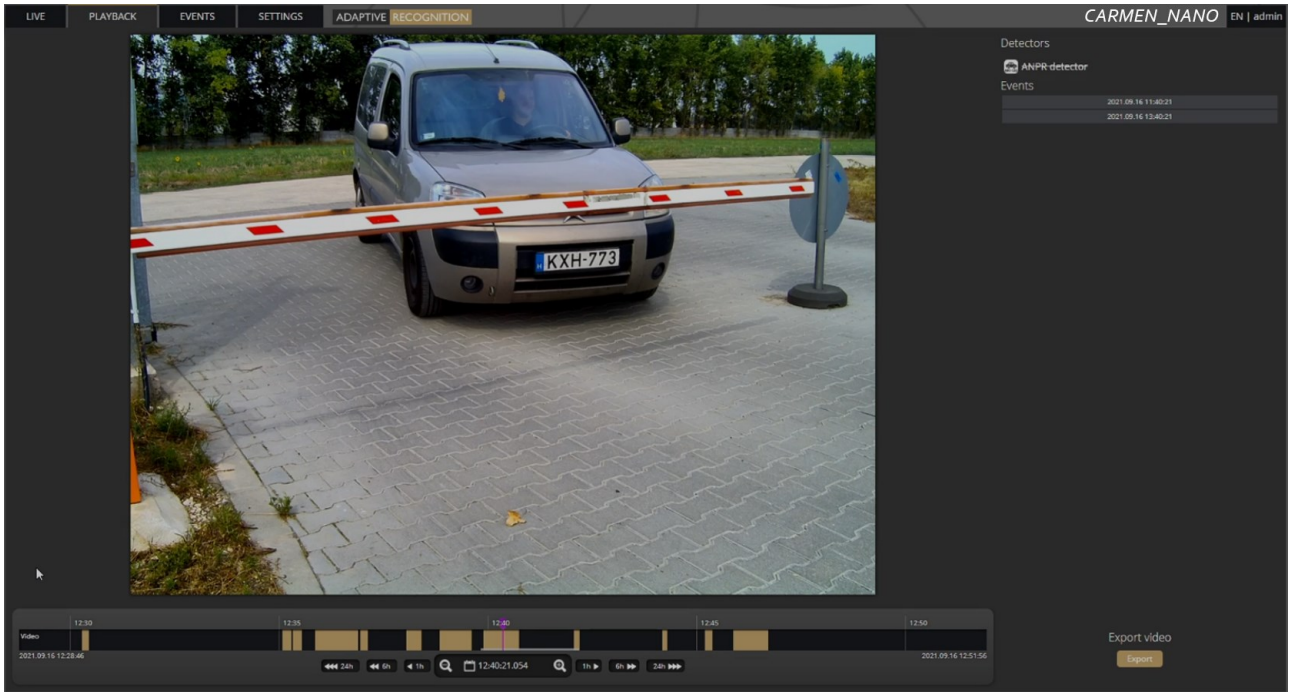
You can find the event preview section under the live stream image, displaying the notifications about the latest received events.

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

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.



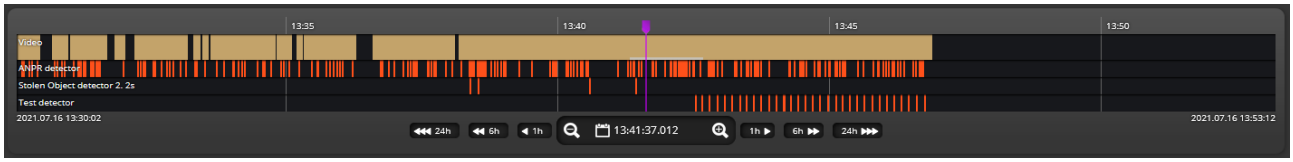
> **Note**
>
> In case the storage is turned off but the storage device is available, the previously recorded elements can be viewed and played if the storage function is switched on.

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.

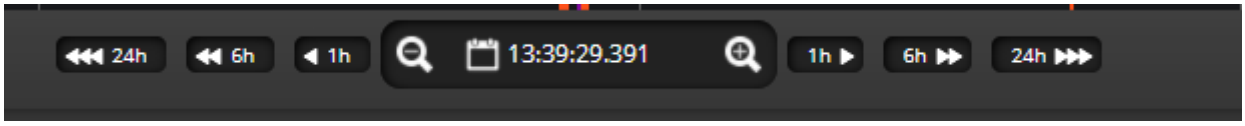
The **red markers** point where events have taken place.



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.

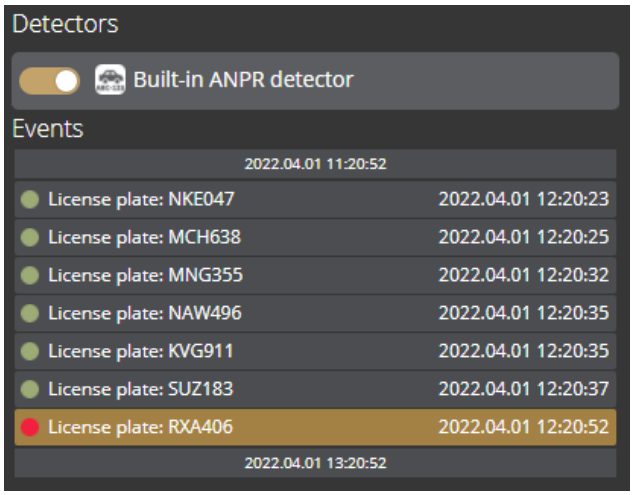
The small image that appears when the mouse cursor is positioned over the timeline shows a preview image of the video near that location.

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.

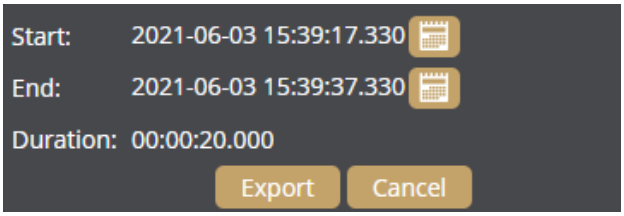


The events and timeline of each detector can be turned on/off by clicking on the appropriate detector button. Clicking on an event in the list, navigates the playback to the date and time of the event.

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**").

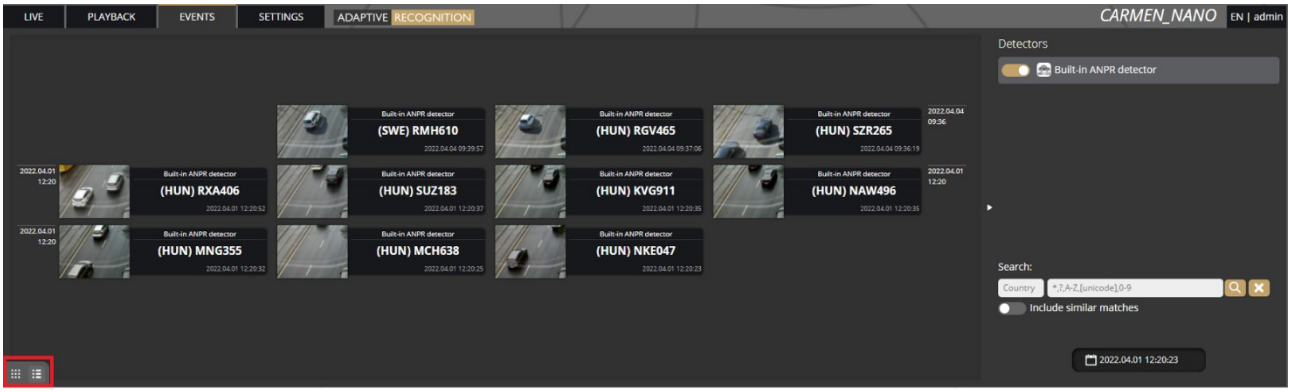
> **Note**
>
> You can adjust the exact time by clicking on the calendar icon.

6. EVENTS

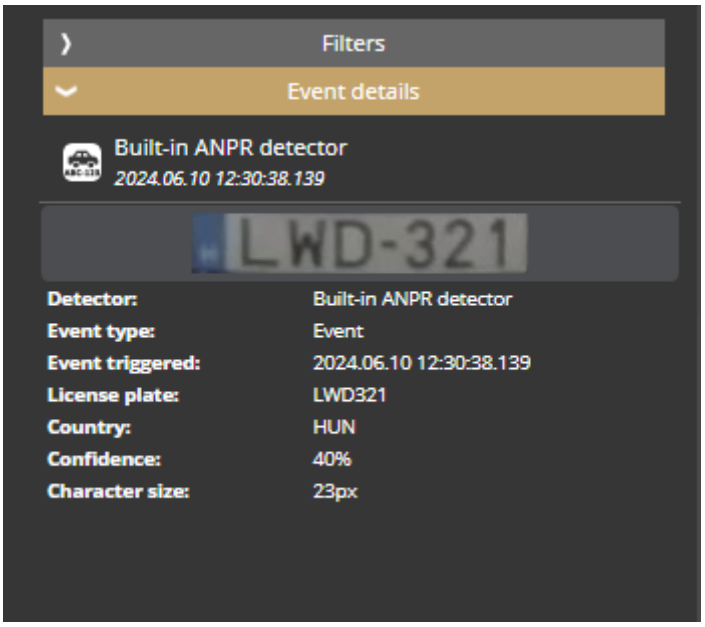
You can access the **EVENTS** interface provided that the storage is turned on. By clicking on this tab, all events recorded by the device will be listed.

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

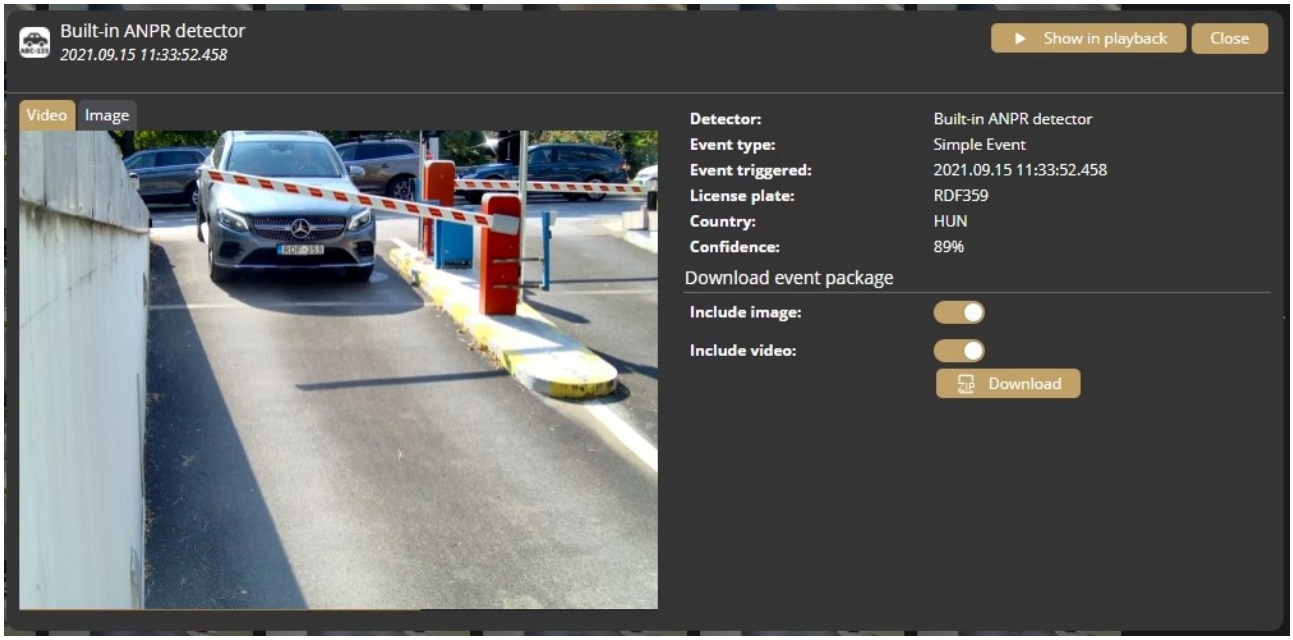
Use the buttons in the bottom left corner to switch between **Grid view** and **Detailed view**.



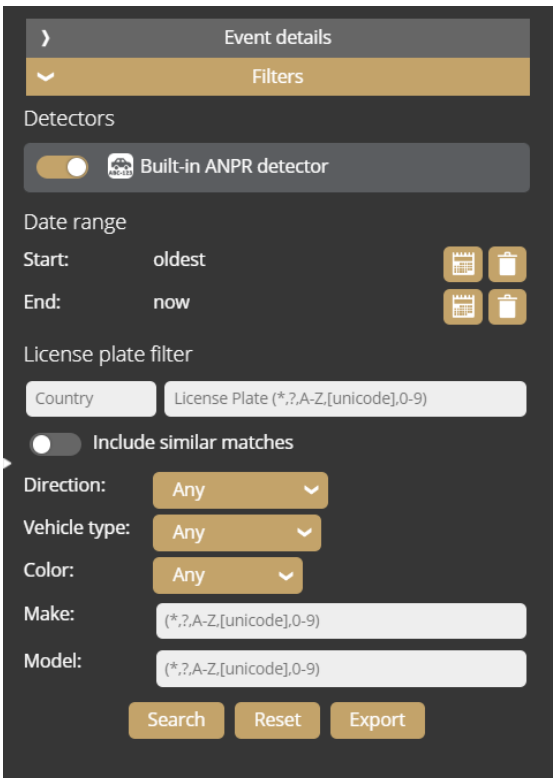
Hovering the cursor over an event, the detector related to the event is highlighted in the list on the right. Simultaneously, a video clip of the event will be loaded and played automatically.



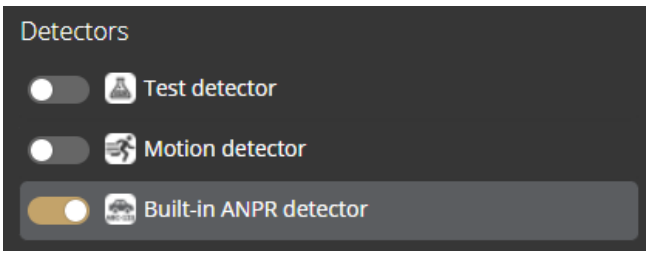
Clicking on an event brings up a detailed view of that event, including a video clip and any related image. The interface can be redirected to the **PLAYBACK** menu item by clicking on the **[Show in playback]** button. The data belonging to the event can be saved as a ZIP file by clicking on the **[Download]**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



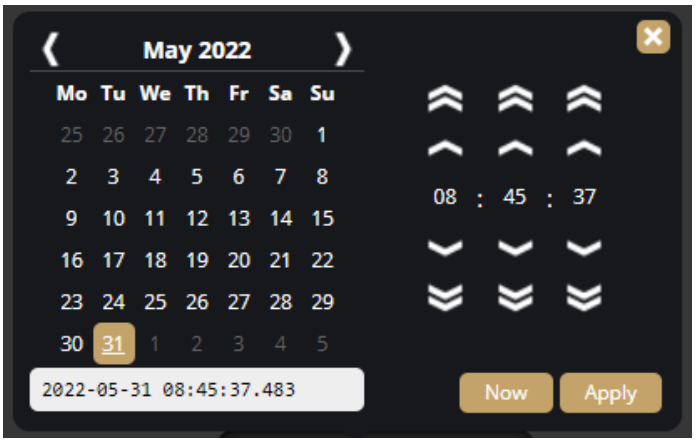
In the top right corner, you can choose to display Event details or Filter options.



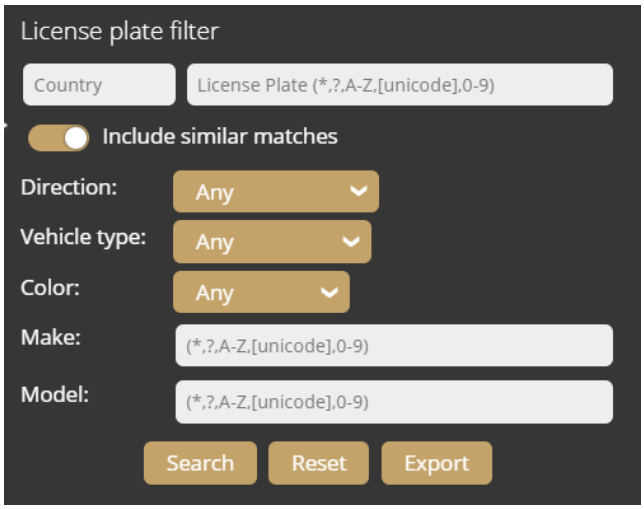
The configured detectors are displayed on the right. By moving the cursor over the detector, the events related to the detector will be highlighted in the event browser. By clicking on the detector, the display of its events can be turned on/off.



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.



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 device in .csv format by clicking on the Export button.

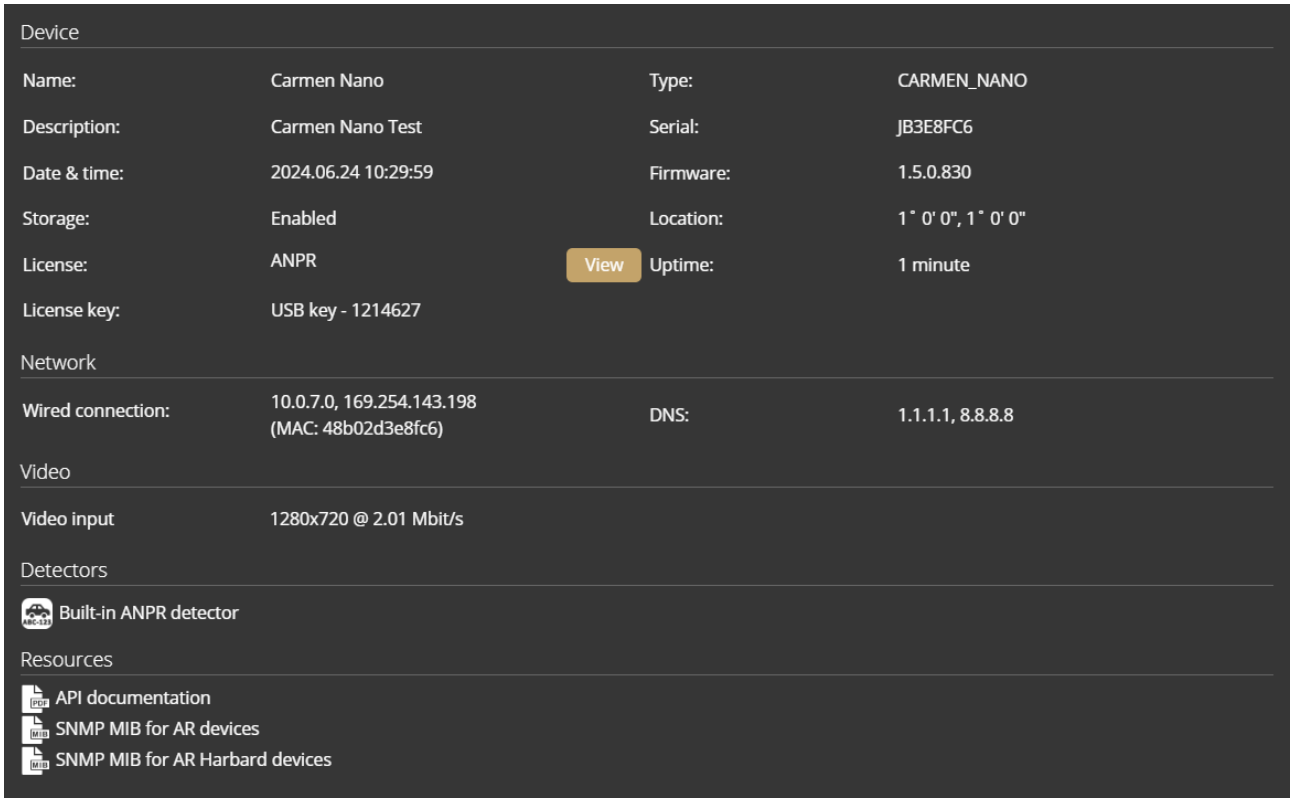
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	Timestamp,"Detector name","Detector ID","Event ID","Event code","Event type","ANPR text","ANPR confidence","ANPR country","ANPR direction","ANPR category","ANPR text color","ANPR make","ANPR model"																					
2	2024-04-07 10:19:01.917,"Built-in 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 device.

7.1. SYSTEM / STATUS

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



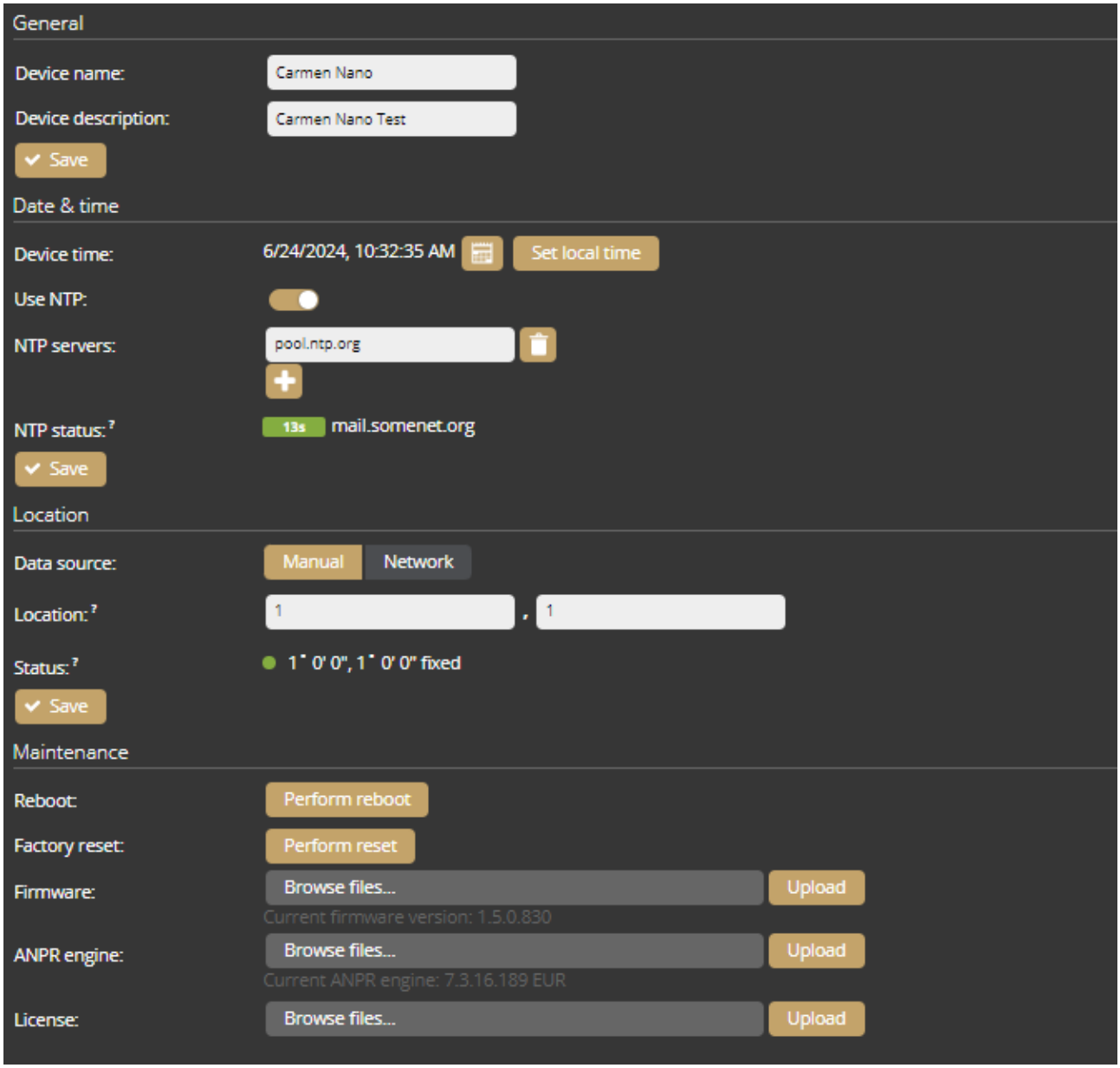
Click on the View button to display the license details. You can see which license is valid for which region and until which date.

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:

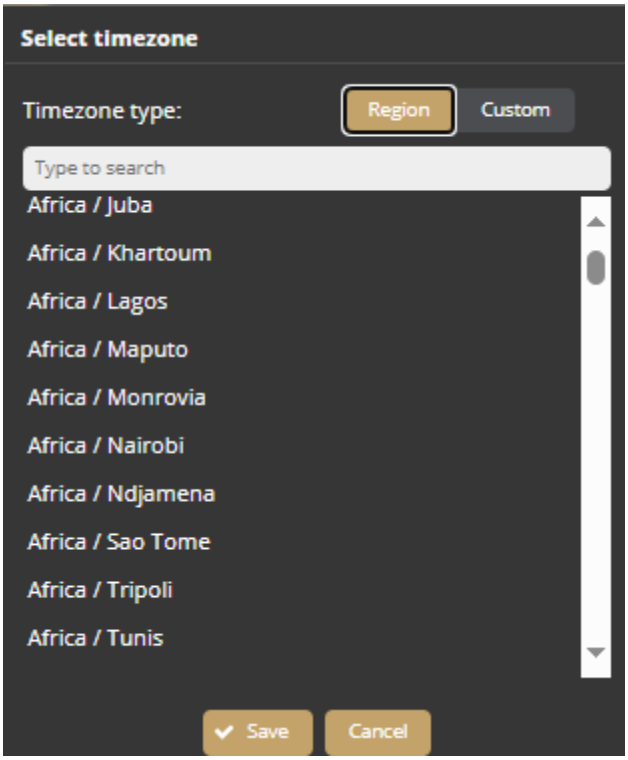
- Modify the name, description of the device
- Reboot the device remotely
- Perform a factory reset (after clicking on the button, the original manufacturer settings are restored except for the network settings)
- Set the date and time
- Upload firmware, ANPR engine and license.
- Set location data/GPS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

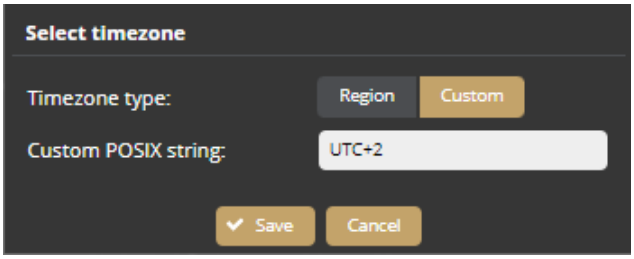


Date & time settings

The current time of the device 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 a local NTP server if you manage more than one camera and/or use integration via API/HTTP/FTP/etc.

Important!

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

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

Color states are the following:

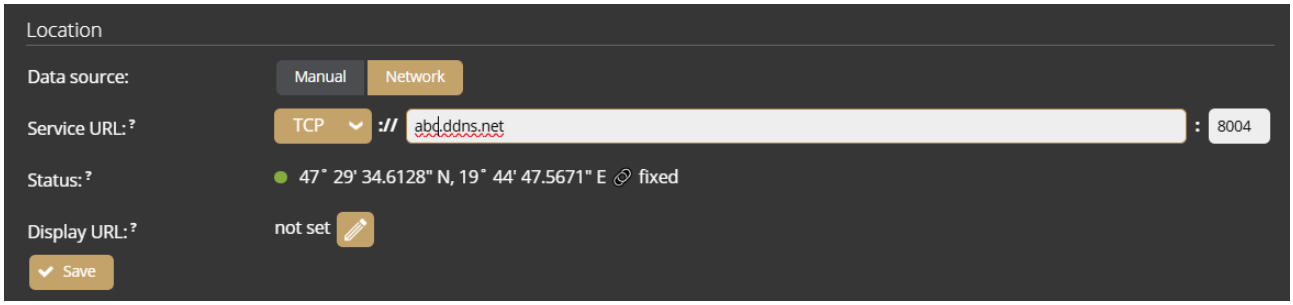
Red: Server is not suitable or unreachable.

Green: Server is working and used for synchronization.

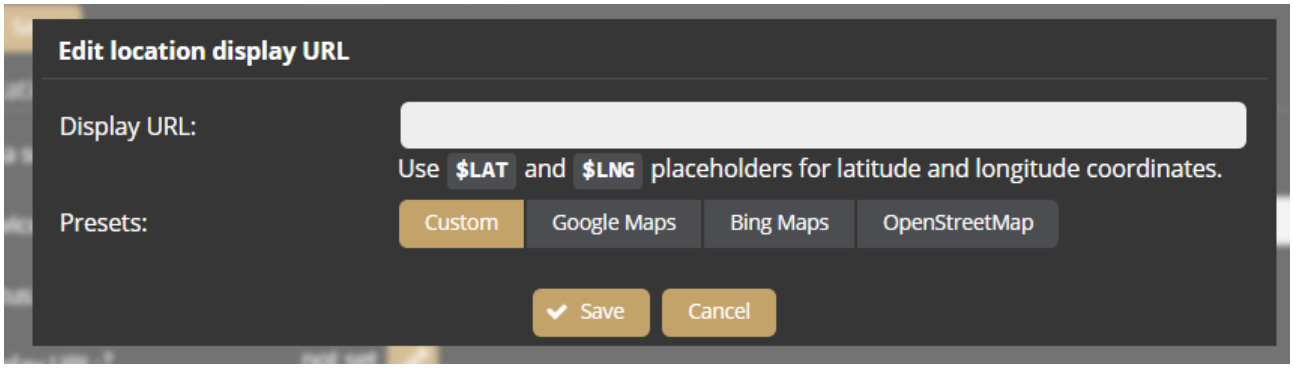
Gray: Server is not used because there is a better alternative.

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.

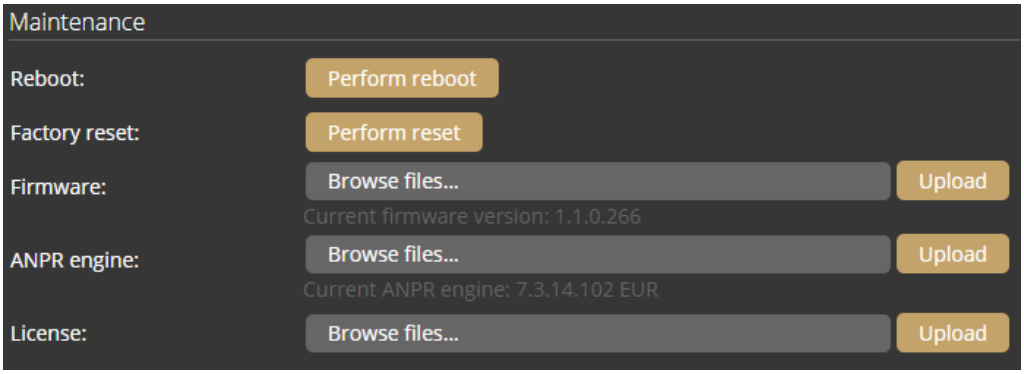


Display URL: By settings an HTTP or HTTPS URL to an external website all location data displayed on this web interface will be a hyperlink to the external site with coordinates as parameters.



Firmware, ANPR, License update

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

The latest firmware and engine can be downloaded here.

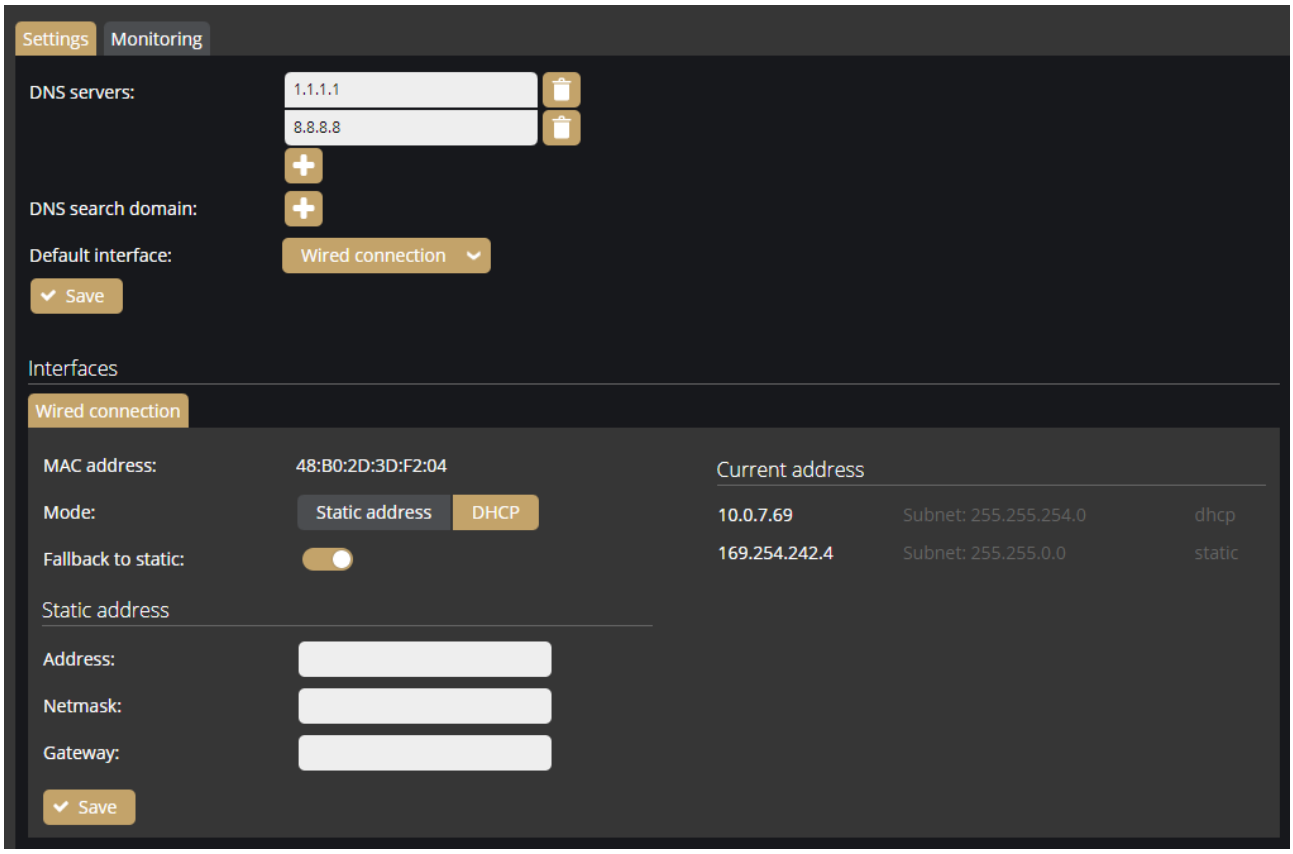
Important!

- During the update process **do not** unplug the device.
- To use the device with the on-board ANPR function, license file and engine file must be uploaded to the device.
- If you try to upload firmware, engine or license for another product to the device, an error message will pop up and the upload will be cancelled.
- If you upload an engine for a region other than the license region to the device, the upload will succeed, but then the device will display an error message in the Settings/Detectors/ANPR engine section, the license plate recognition will not work.
- The device can be upgraded with firmware released after the license expires. The license expiration only affects the engine, the device cannot be updated with the engine released after the license expiration. License expiration does not affect the device functionality.

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.

1. 

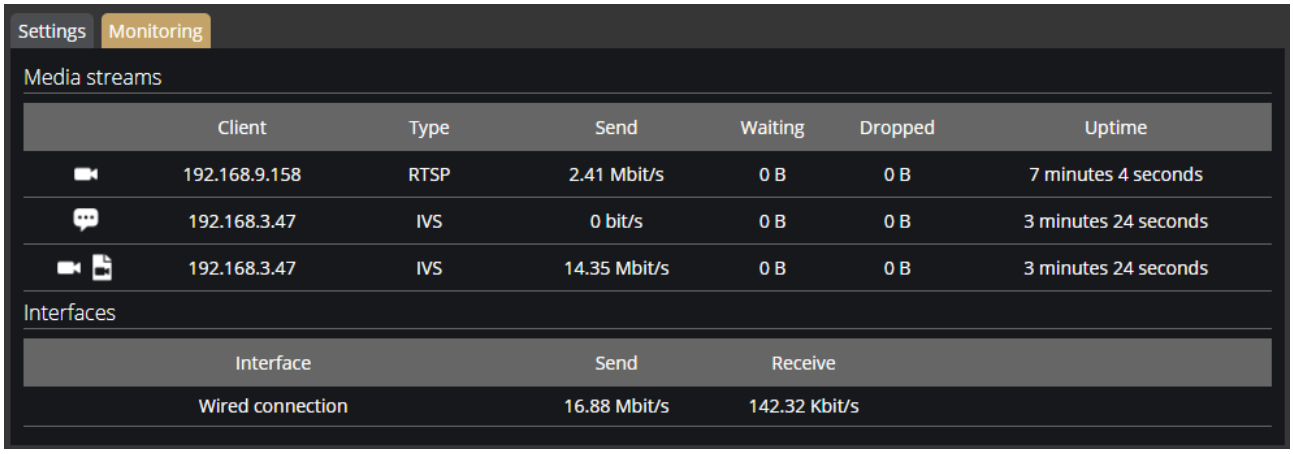
Fallback to static

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

Put the Carmen Nano device and your camera in the same network segment.

Monitoring

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



7.4. SYSTEM / SECURITY

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

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

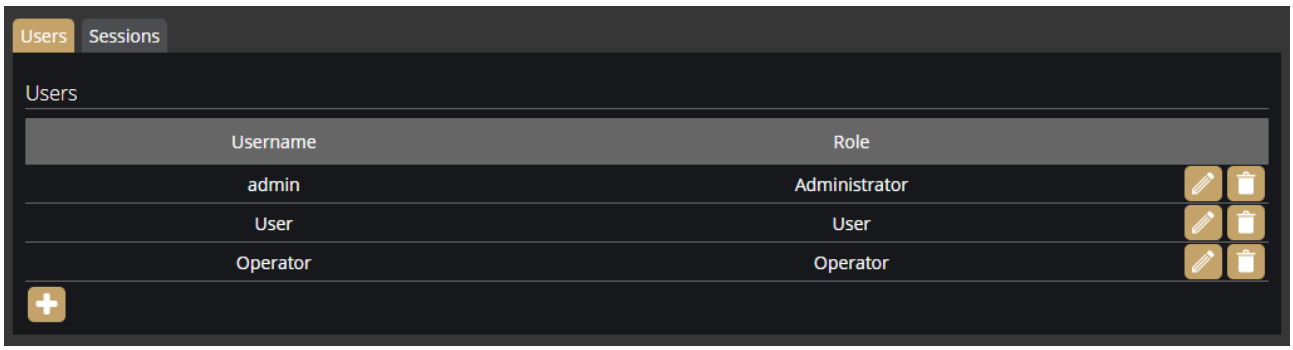
The default user name and password is "**admin**".

Important!

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

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

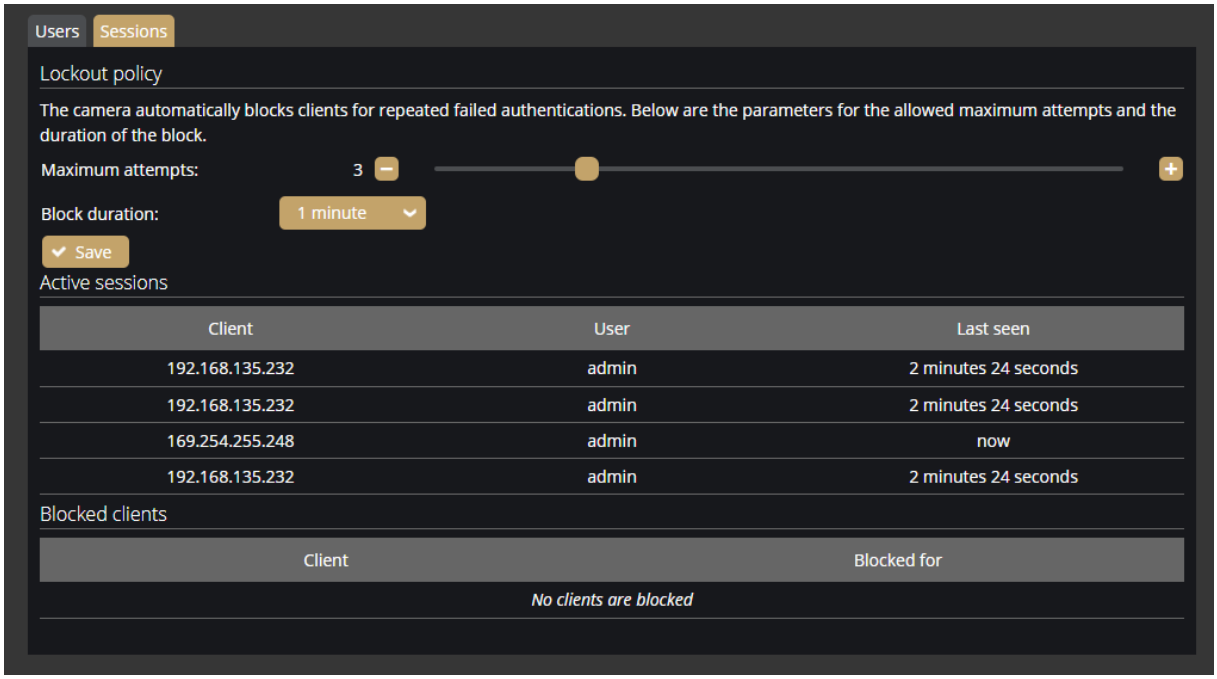
1. **Administrator**: The administrator can access and edit all parameters of the device.
2. **User**: The user can view but not edit the parameters of the device. Some pages containing sensitive information may be hidden.
3. **Operator**: The operator has the same privileges as a user.



Sessions

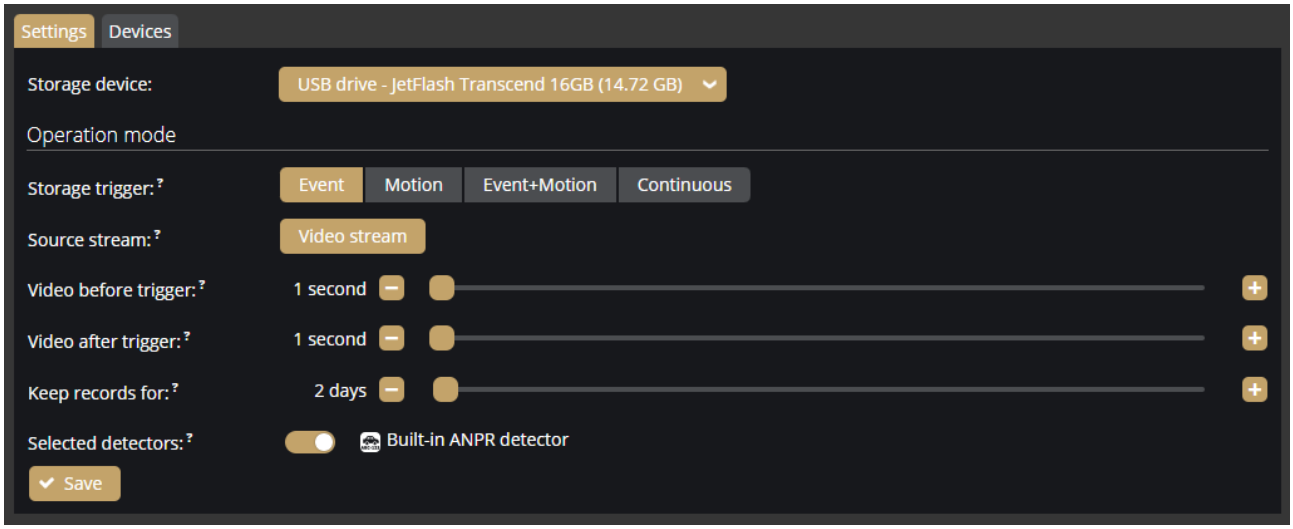
At **Lockout policy,** the maximum number of failed login attempts can be adjusted. After reaching the specified number, the device blocks that session. By default, after three failed login attempts, the device blocks the IP address of the client for a minute. Note that the number of **Maximum attempts**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

The Active sessions and Blocked clients can also be seen on this tab.



7.5. SYSTEM / STORAGE

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



Operation mode

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

Important!

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

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

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

Recordings before and after activation (seconds)

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

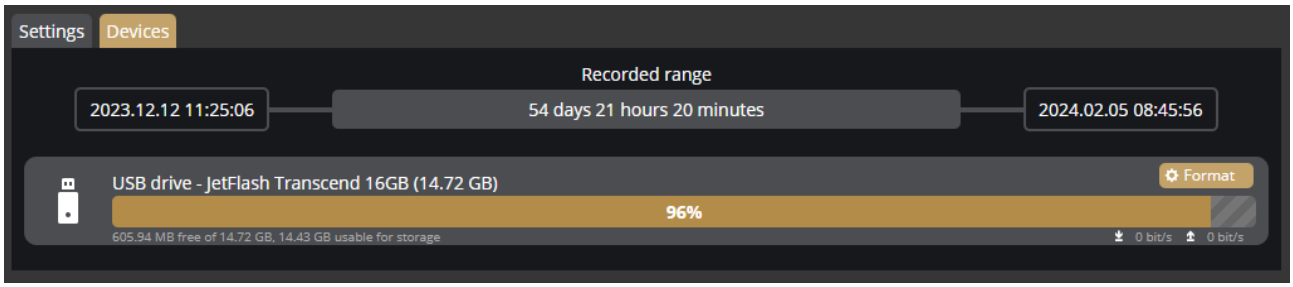
Keep records for: Recordings are automatically deleted from the storage medium after the specified time. You can set the number of days after which data is deleted from the SD card.

Selected detectors

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

Devices

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



Formatting the storage device

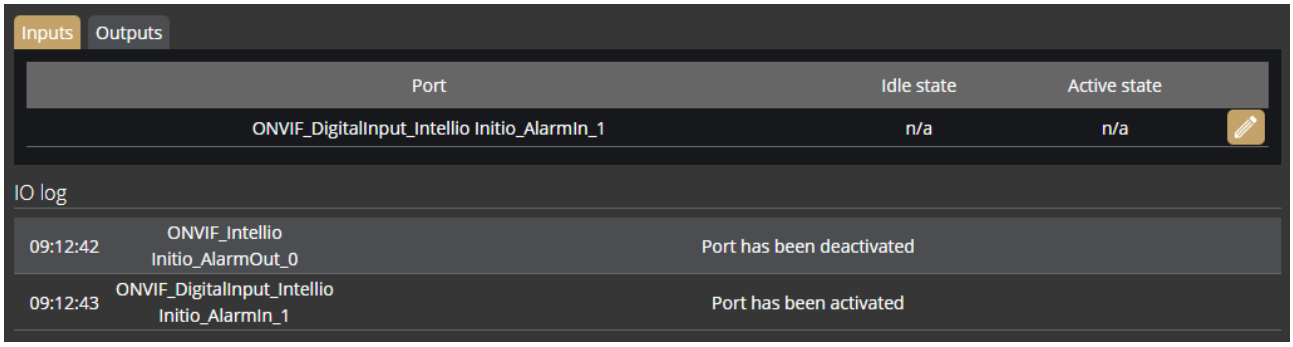
With the **[Format]** button, you can format the storage unit immediately. After clicking on the **[Format]** button, a window pops up. Click on the **[Yes]** button to start the operation. The capacity bar indicates the remaining time of the formatting process.

Important!

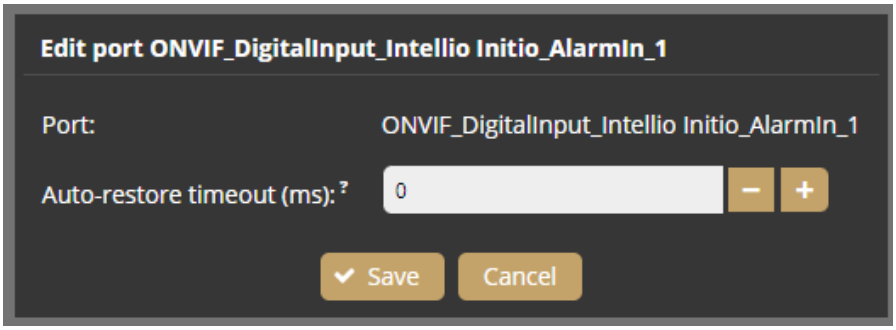
The formatting deletes every data from the storage device.

7.6. I/O

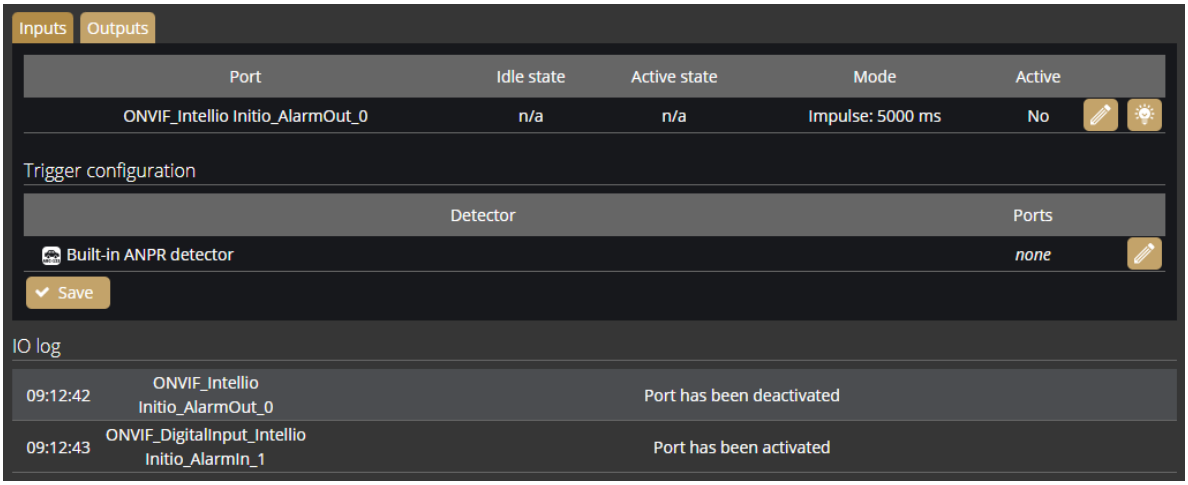
In the I/O menu you can modify the input and output settings of the added ONVIF devices, the trigger configuration and you can monitor the state changes of the input/output ports at IO log section.



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

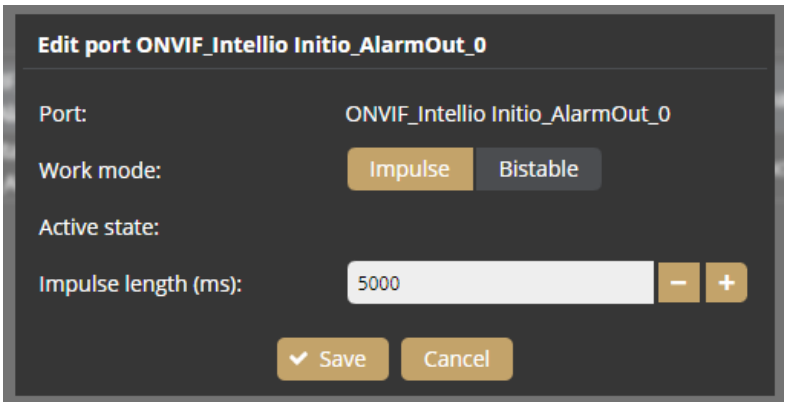


You can also modify the Output ports at the Outputs tab.



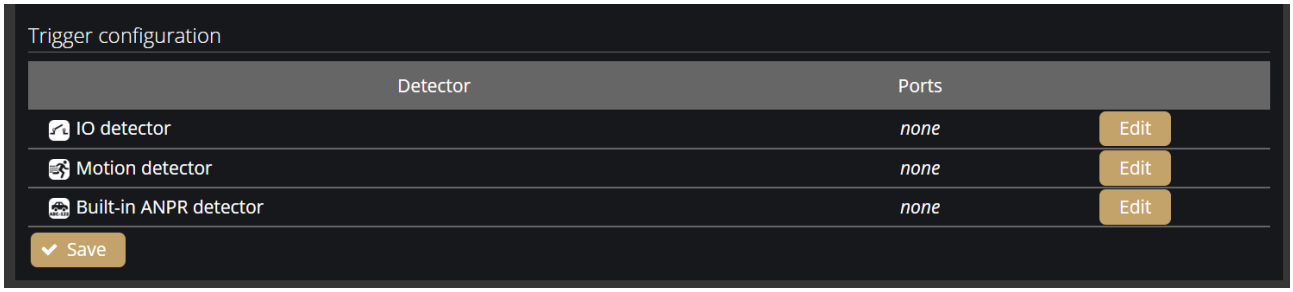
The following parameters can be adjusted after clicking on the Edit button:

- **Work mode:** Impulse or Bistable
- **Active state:** The active state of the port. If it is "Open", the port is open when an event occurs. If it is "Closed", the port closes when an event occurs.
- **Impulse lenght (ms):** In the case of activating the output port, the length of the active state can be adjusted.

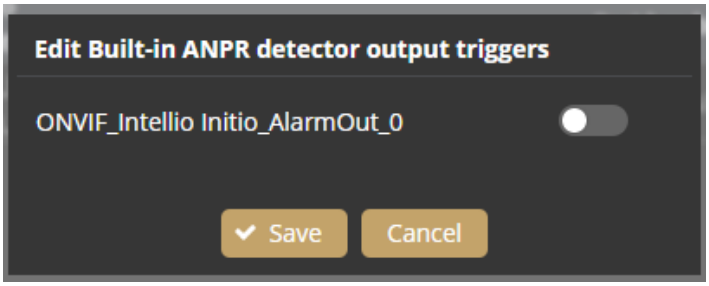


The detectors configured in the Analitycs/Detectors menu can be associated to the Ports of the added ONVIF device in the Trigger configuration menu. (You can add an ONVIF device in the **External** menu.)

For some cameras, the IO port must also be enabled separately on the camera.



Select which ONVIF device the Detector should be associated with.



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.

WS Discovery:

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

IVS

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

RTSP

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

SNMP

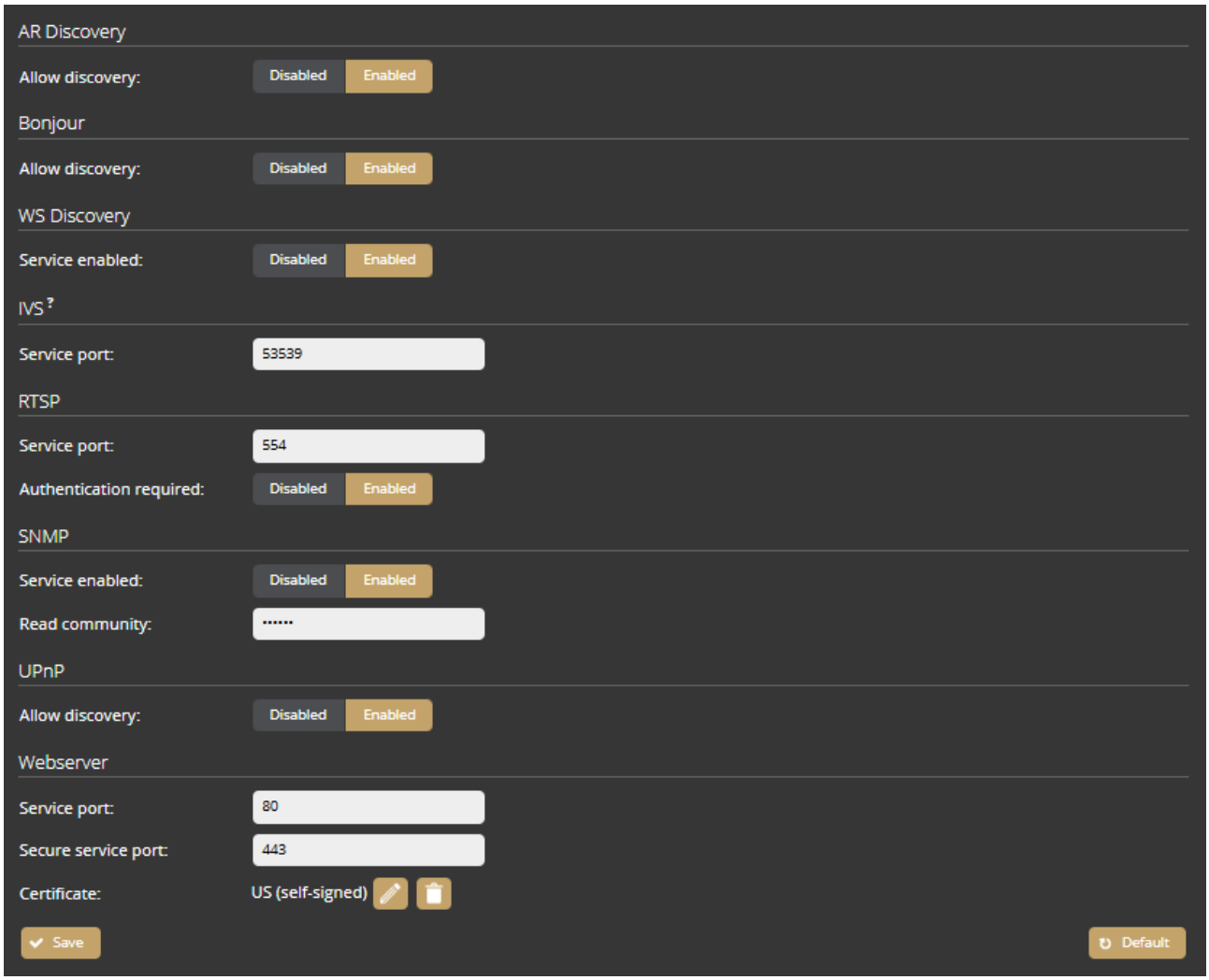
- **Service enabled:** The SNMP service can be enabled or disabled.
- **Read community:** Password required for the camera properties on SNMP.

UPnP

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

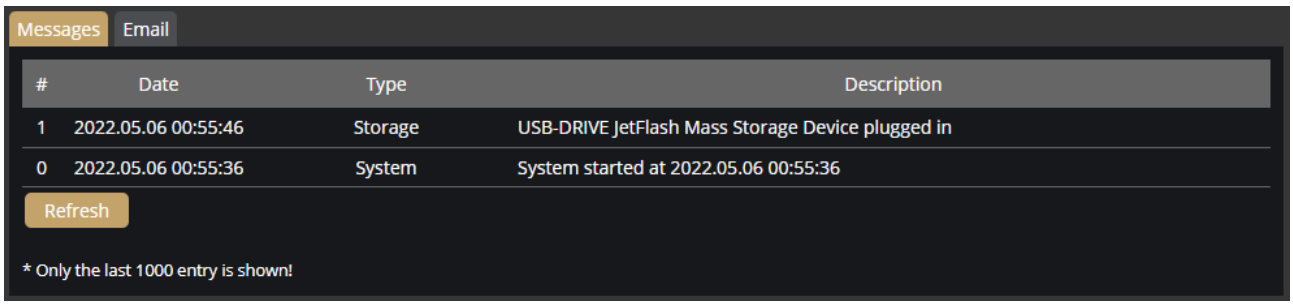
Webserver

- **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 self-signed SSL certificate. This certification is used by the camera to provide HTTPs services.
  (If you want 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
WS Discovery
Service enabled:
Disabled
Enabled
IVS
Service port:
53539
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

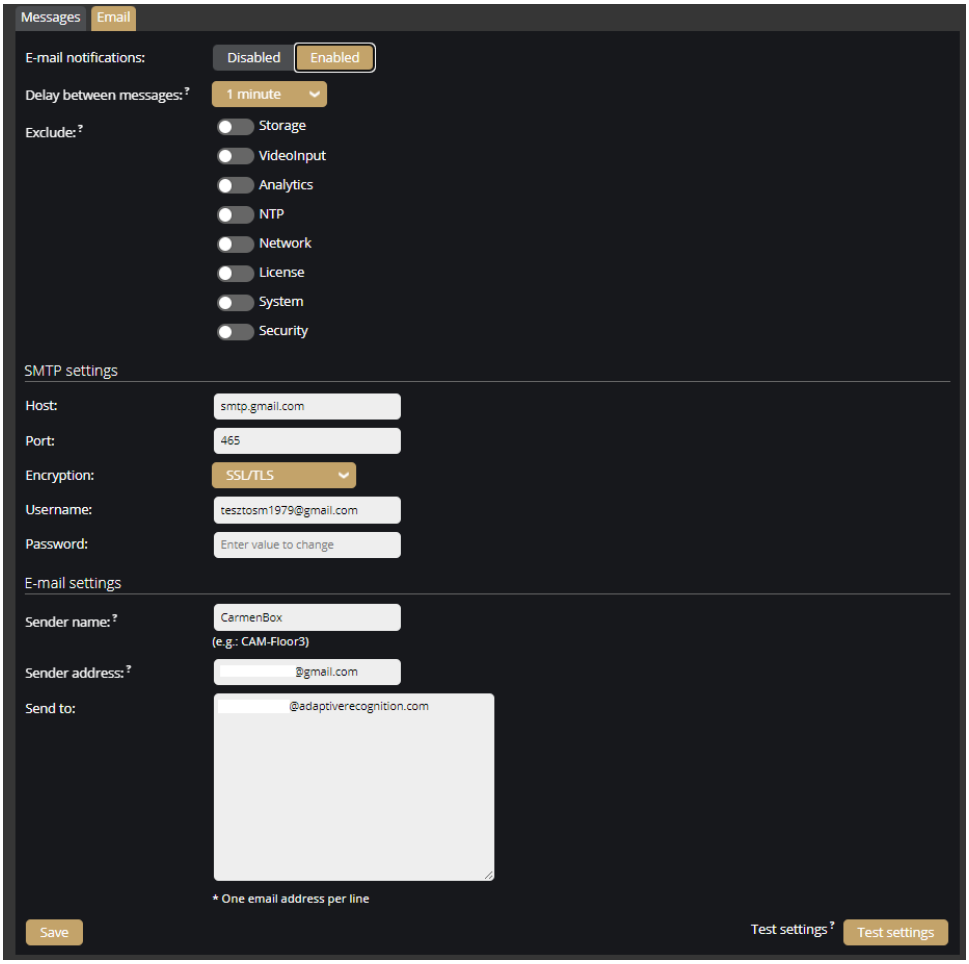
7.8. SYSTEM / NOTIFICATIONS

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



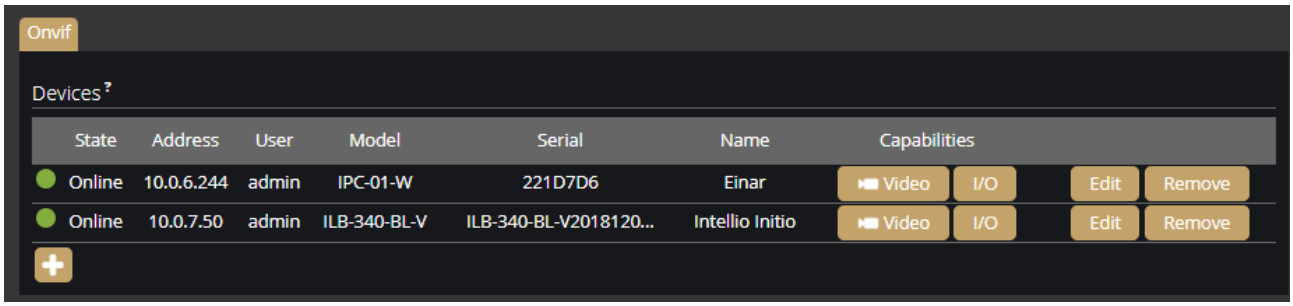
In the **Email** tab, you can specify the email settings for sending messages. The following parameters can be adjusted after clicking 62nt he **[Enabled]** button:

- **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

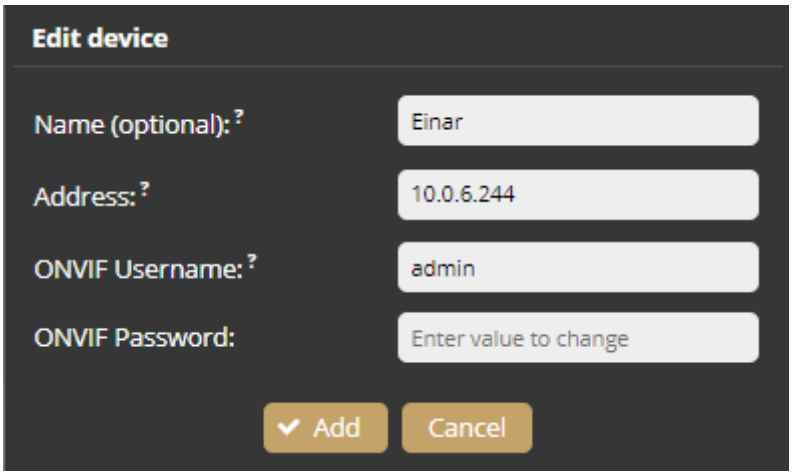
You can manage the associated Onvif devices in the External menu. You can add a new device, edit the data of existing devices and delete a device.



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

The following should be set when adding a new device:

- **Name: (optional):** The name of the device can be entered.
- **Adress:** 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

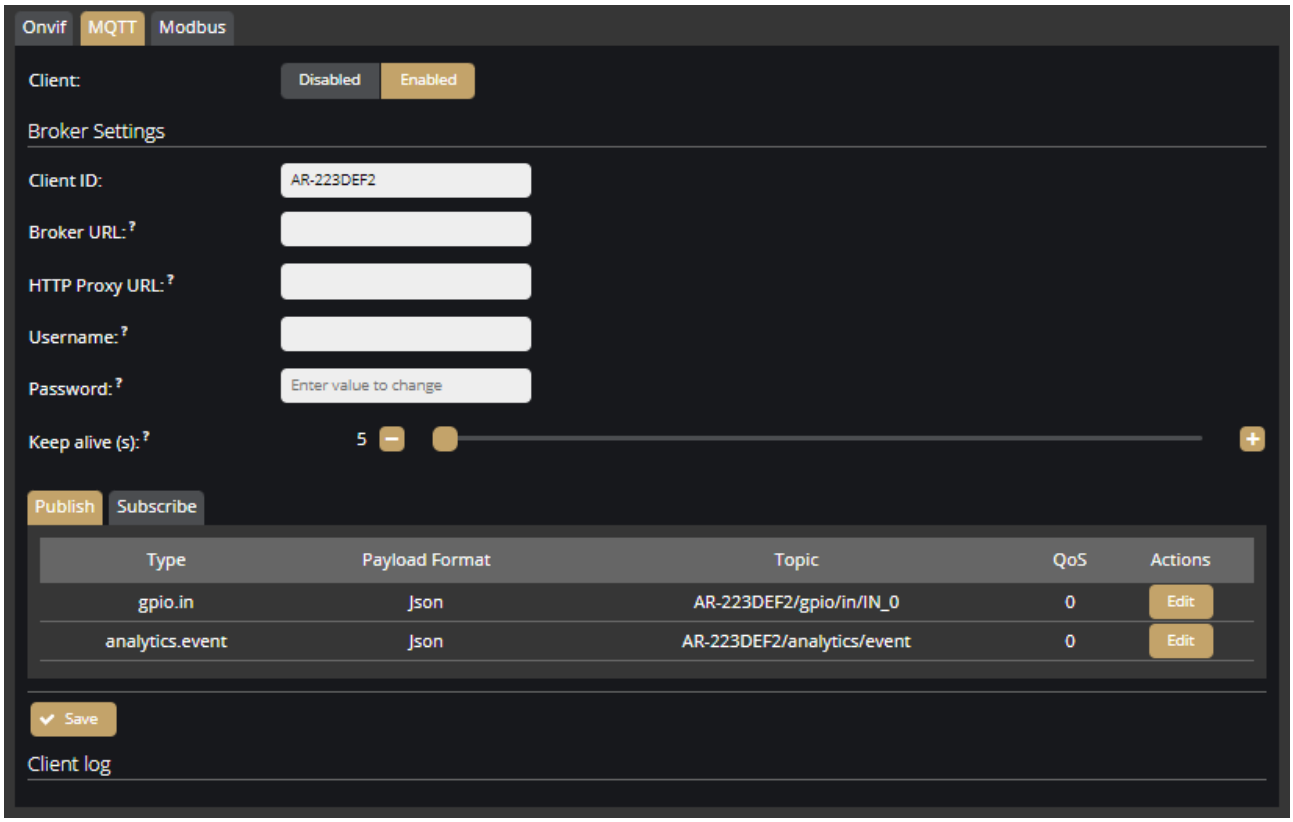
wss://<address>:<port>[/path]*[?query] - Secure Websocket

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

Username: Username for the MQTT Broker service.

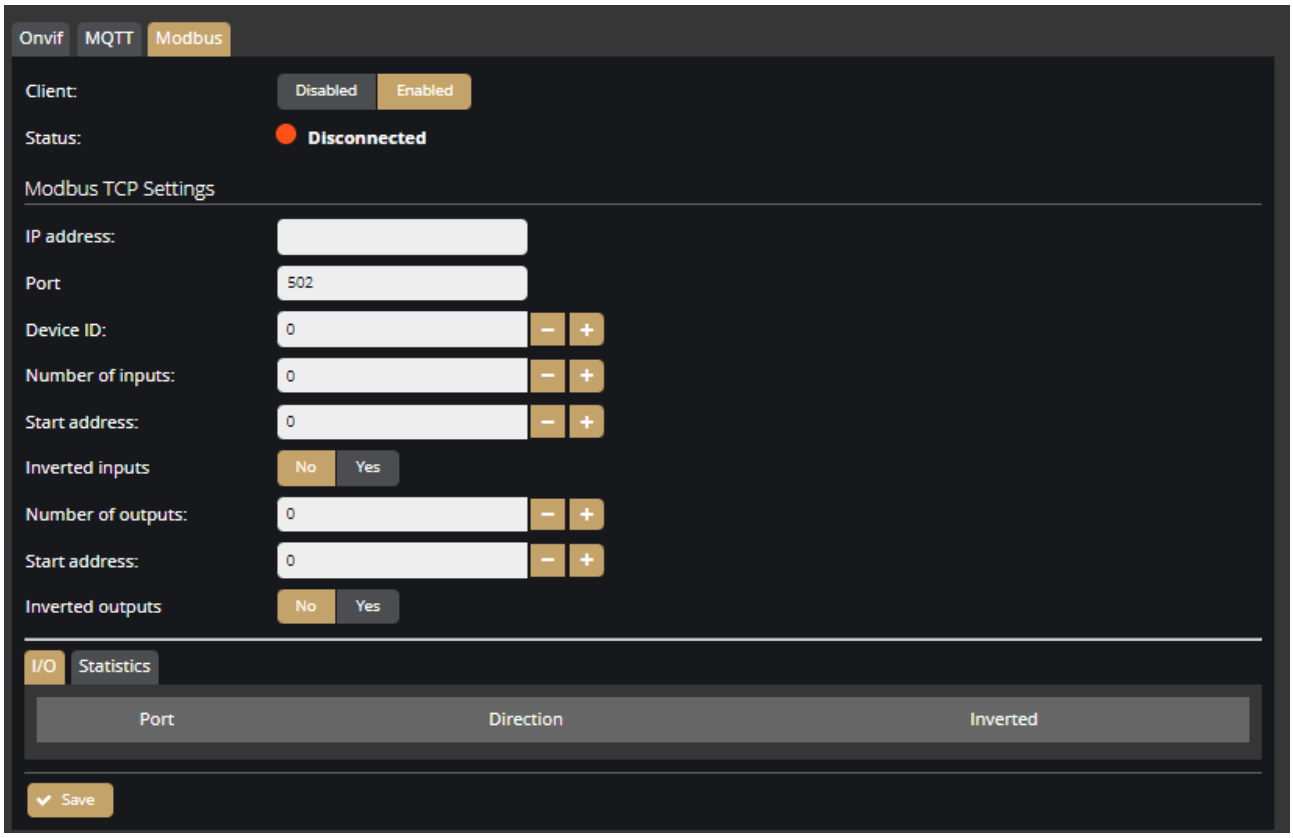
Password: Password for the MQTT Broker service.

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



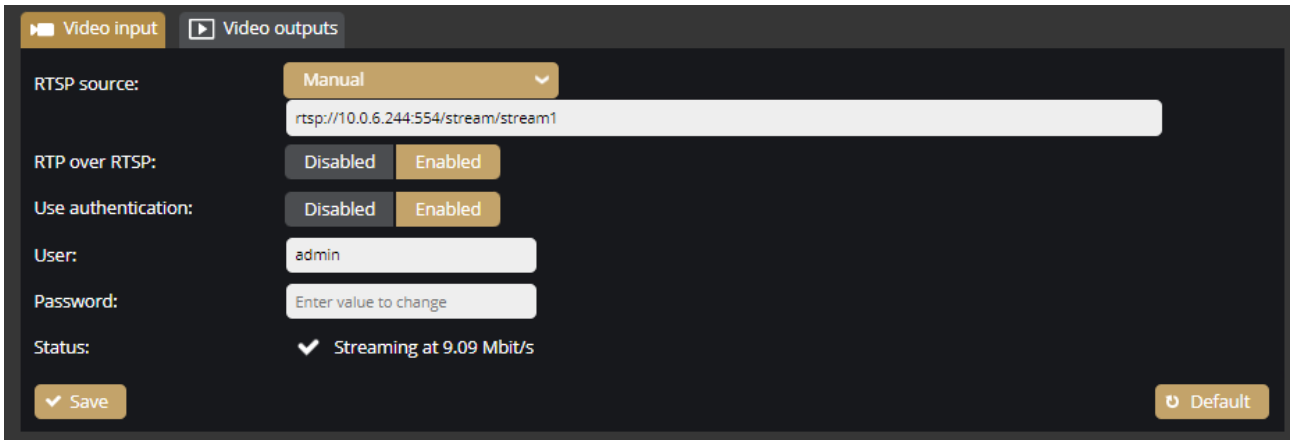
7.11. SYSTEM / EXTERNAL / MODBUS

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



7.12. MEDIA / VIDEO

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



Video input

The following parameters can be set:

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

> Note
>
> **Video input stream criteria:**
>
> - Camera stream protocols: RTSP
> - Camera stream format: H.264
> - Supported resolution: up to 5MP
> - Recommended FPS: 25-30

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

AR Vidar, MicroCam:

rtsp://Camera_IP/stream/h264

Intellio Visus:

rtsp://Camera_IP:554/primary/h264

rtsp://Camera_IP:554/secondary/h264

Intellio Initio:

rtsp://Camera_IP:554

AXIS:

rtsp://Camera_IP/axis-media/media.amp

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

rtsp://Camera_IP:554/onvif-media/media.amp?profile=profile_1_h264&sessiontimeout=60&streamtype=unicast

rtsp://Camera_IP:554/onvif-media/media.amp?profile=profile2&sessiontimeout=60&streamtype=unicast

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

Bosch:

rtsp://Camera_IP:554/rtsp_tunnel?p=0&h26x=4

rtsp://Camera_IP:554/rtsp_tunnel?p=1&inst=2&h26x=4

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

Dahua:

Main stream:

rtsp://Camera_IP:554/cam/realmonitor?channel=1&subtype=0&unicast=true

rtsp://Camera_IP:554/live

Sub streams:

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

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

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

Hanwha:

rtsp://Camera_IP/profile2/media.smp

(tested with PNO-A6081R, FW: 2.11.02_20210630_R206)

Hikvision:

Main stream:

rtsp://Camera_IP:554/Streaming/Channels/101

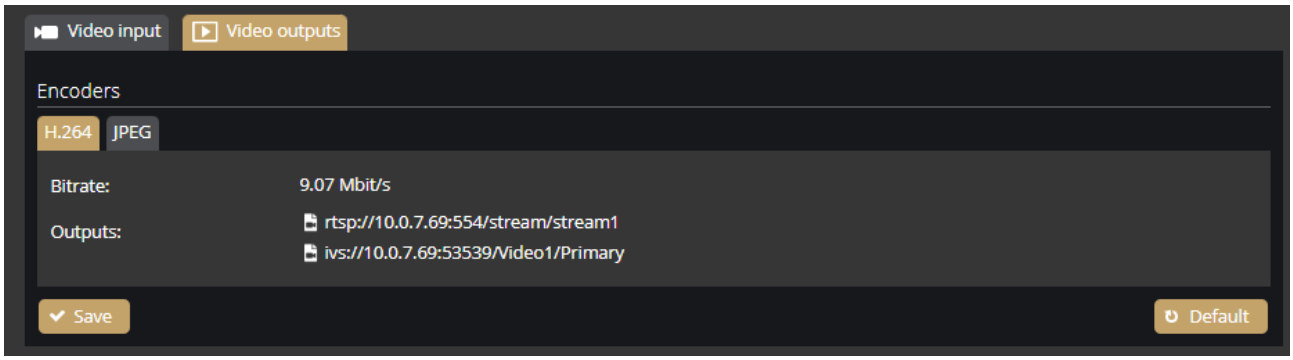
Sub stream:

rtsp://Camera_IP:554/Streaming/Channels/102

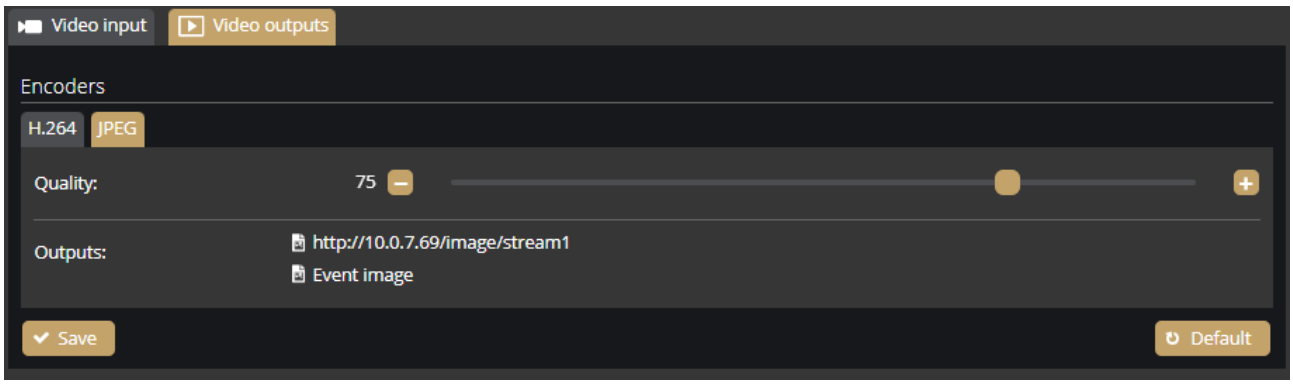
Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

Video outputs

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



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



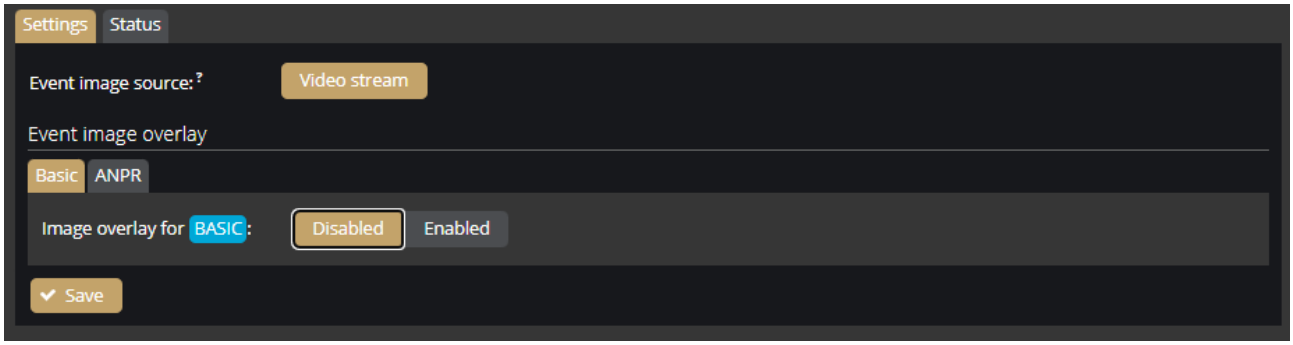
Videostream / image URL links can be found on this interface, such as:

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

7.13. ANALYTICS / SETTINGS

Settings

You can use the Settings tab to set the event image source and event image overlay.



Event image overlay

The text overlay field containing the event data can be placed on the saved event image.

You can make the following settings:

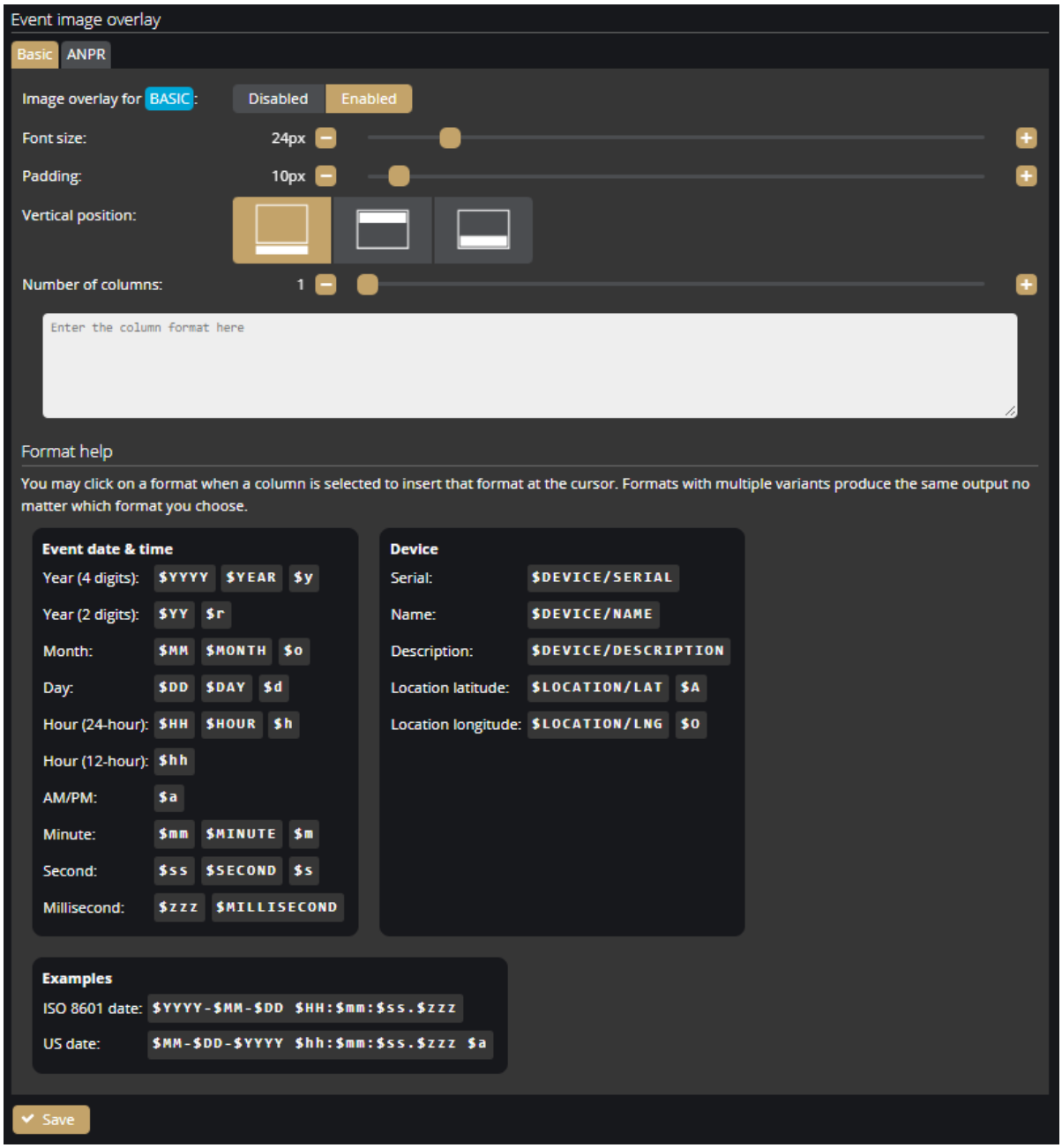
- Front size
- Padding
- Vertical position
- Number of columns
- Column format

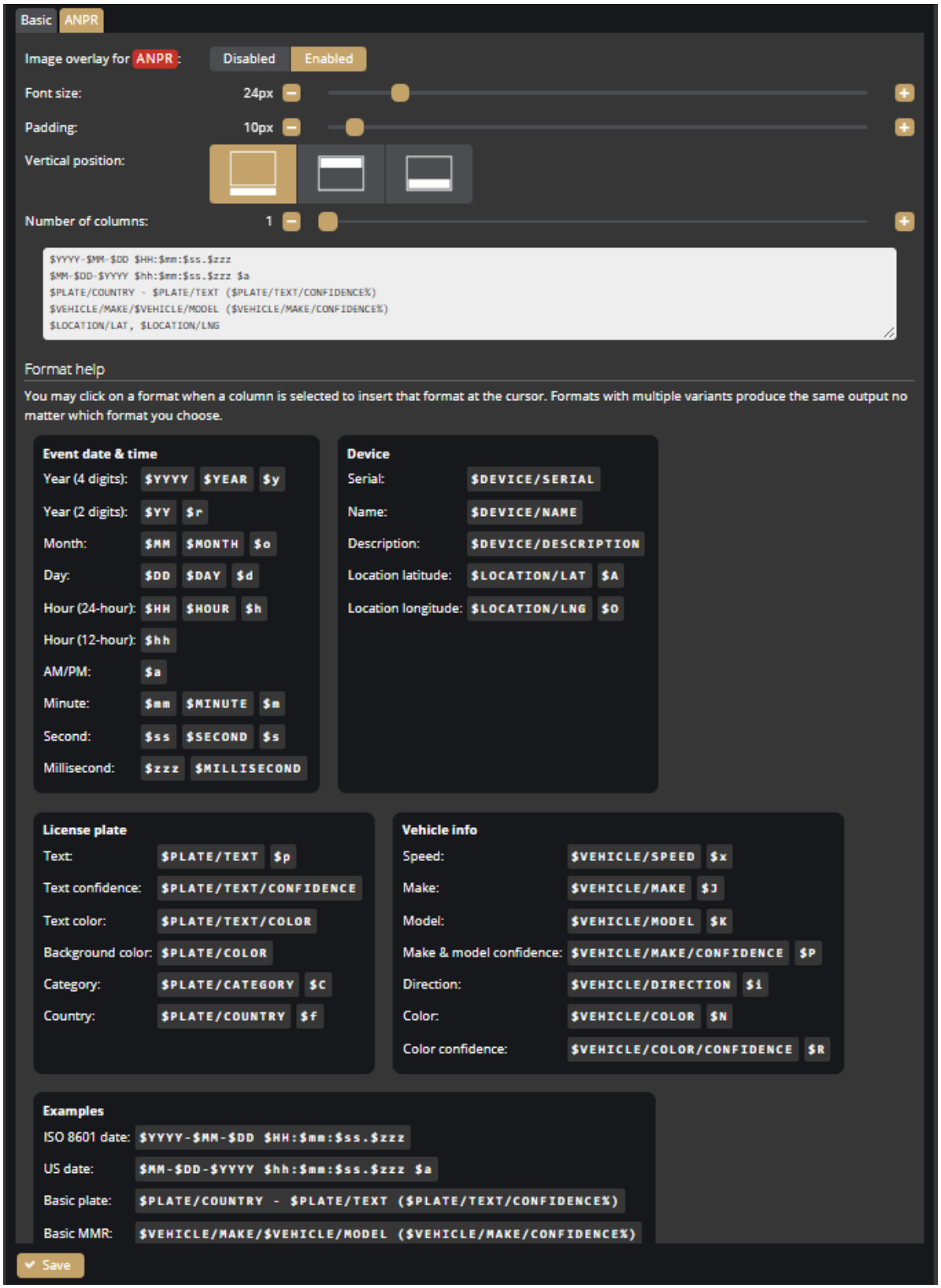
Format help

You may click on a format when a column is selected to insert that format at the cursor. Formats with multiple variants produce the same output no matter which format you choose.

Important!

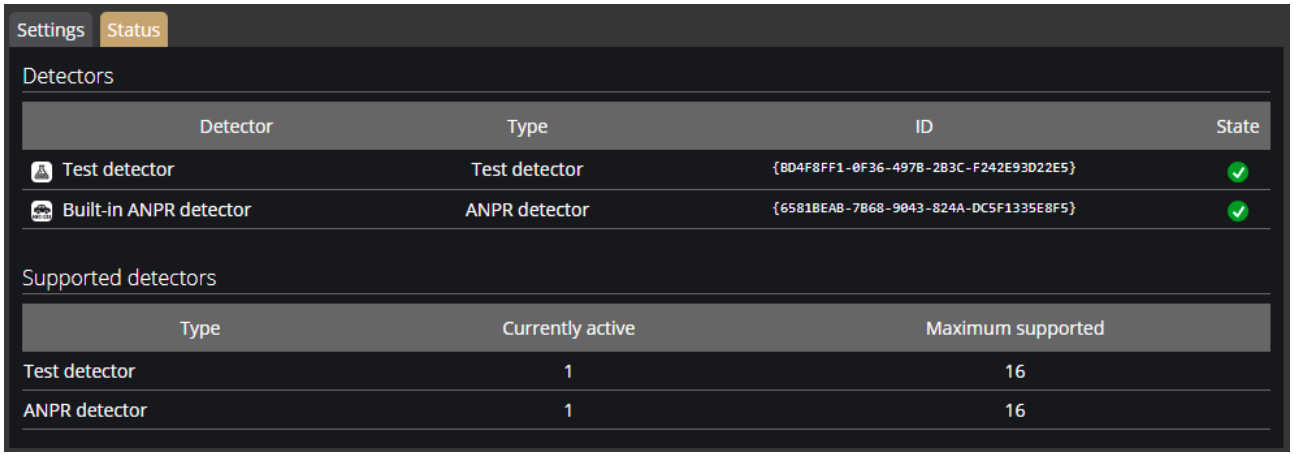
To overlay ANPR event images, use the ANPR option. With the BASIC option, you can only overlay images of events created by the Motion, IO and Test detectors.


Event image overlay
Basic
ANPR
Image overlay for BASIC:
Disabled
Enabled
Font size:
24px
Padding:
10px
Vertical position:
Number of columns:
1
Enter the column format here
Format help
You may click on a format when a column is selected to insert that format at the cursor. Formats with multiple variants produce the same output no matter which format you choose.
Event date & time
Year (4 digits): $YYYY $YEAR $y
Year (2 digits): $YY $r
Month: $MM $MONTH $o
Day: $DD $DAY $d
Hour (24-hour): $HH $HOUR $h
Hour (12-hour): $hh
AM/PM: $a
Minute: $mm $MINUTE $m
Second: $ss $SECOND $s
Millisecond: $zzz $MILLISECOND
Device
Serial: $DEVICE/SERIAL
Name: $DEVICE/NAME
Description: $DEVICE/DESCRIPTION
Location latitude: $LOCATION/LAT $A
Location longitude: $LOCATION/LNG $O
Examples
ISO 8601 date: $YYYY-$MM-$DD $HH:$mm:$ss.$zzz
US date: $MM-$DD-$YYYY $hh:$mm:$ss.$zzz $a
Save


Basic
ANPR
Image overlay for ANPR:
Disabled
Enabled
Font size:
24px
Padding:
10px
Vertical position:
Number of columns:
1
$YYYY-$MM-$DD $HH:$mm:$ss.$zzz
$MM-$DD-$YYYY $hh:$mm:$ss.$zzz $a
$PLATE/COUNTRY - $PLATE/TEXT ($PLATE/TEXT/CONFIDENCE%)
$VEHICLE/MAKE/$VEHICLE/MODEL ($VEHICLE/MAKE/CONFIDENCE%)
$LOCATION/LAT, $LOCATION/LNG
Format help
You may click on a format when a column is selected to insert that format at the cursor. Formats with multiple variants produce the same output no matter which format you choose.
Event date & time
Year (4 digits): $YYYY $YEAR $y
Year (2 digits): $YY $r
Month: $MM $MONTH $o
Day: $DD $DAY $d
Hour (24-hour): $HH $HOUR $h
Hour (12-hour): $hh
AM/PM: $a
Minute: $mm $MINUTE $m
Second: $ss $SECOND $s
Millisecond: $zzz $MILLISECOND
Device
Serial: $DEVICE/SERIAL
Name: $DEVICE/NAME
Description: $DEVICE/DESCRIPTION
Location latitude: $LOCATION/LAT $A
Location longitude: $LOCATION/LNG $O
License plate
Text: $PLATE/TEXT $p
Text confidence: $PLATE/TEXT/CONFIDENCE
Text color: $PLATE/TEXT/COLOR
Background color: $PLATE/COLOR
Category: $PLATE/CATEGORY $C
Country: $PLATE/COUNTRY $F
Vehicle info
Speed: $VEHICLE/SPEED $x
Make: $VEHICLE/MAKE $J
Model: $VEHICLE/MODEL $K
Make & model confidence: $VEHICLE/MAKE/CONFIDENCE $P
Direction: $VEHICLE/DIRECTION $i
Color: $VEHICLE/COLOR $N
Color confidence: $VEHICLE/COLOR/CONFIDENCE $R
Examples
ISO 8601 date: $YYYY-$MM-$DD $HH:$mm:$ss.$zzz
US date: $MM-$DD-$YYYY $hh:$mm:$ss.$zzz $a
Basic plate: $PLATE/COUNTRY - $PLATE/TEXT ($PLATE/TEXT/CONFIDENCE%)
Basic MMR: $VEHICLE/MAKE/$VEHICLE/MODEL ($VEHICLE/MAKE/CONFIDENCE%)
Save

Status

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



7.14. ANALYTICS / DETECTORS

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

7.14.1. MOTION ENGINE AND GENERAL USE OF MASKS

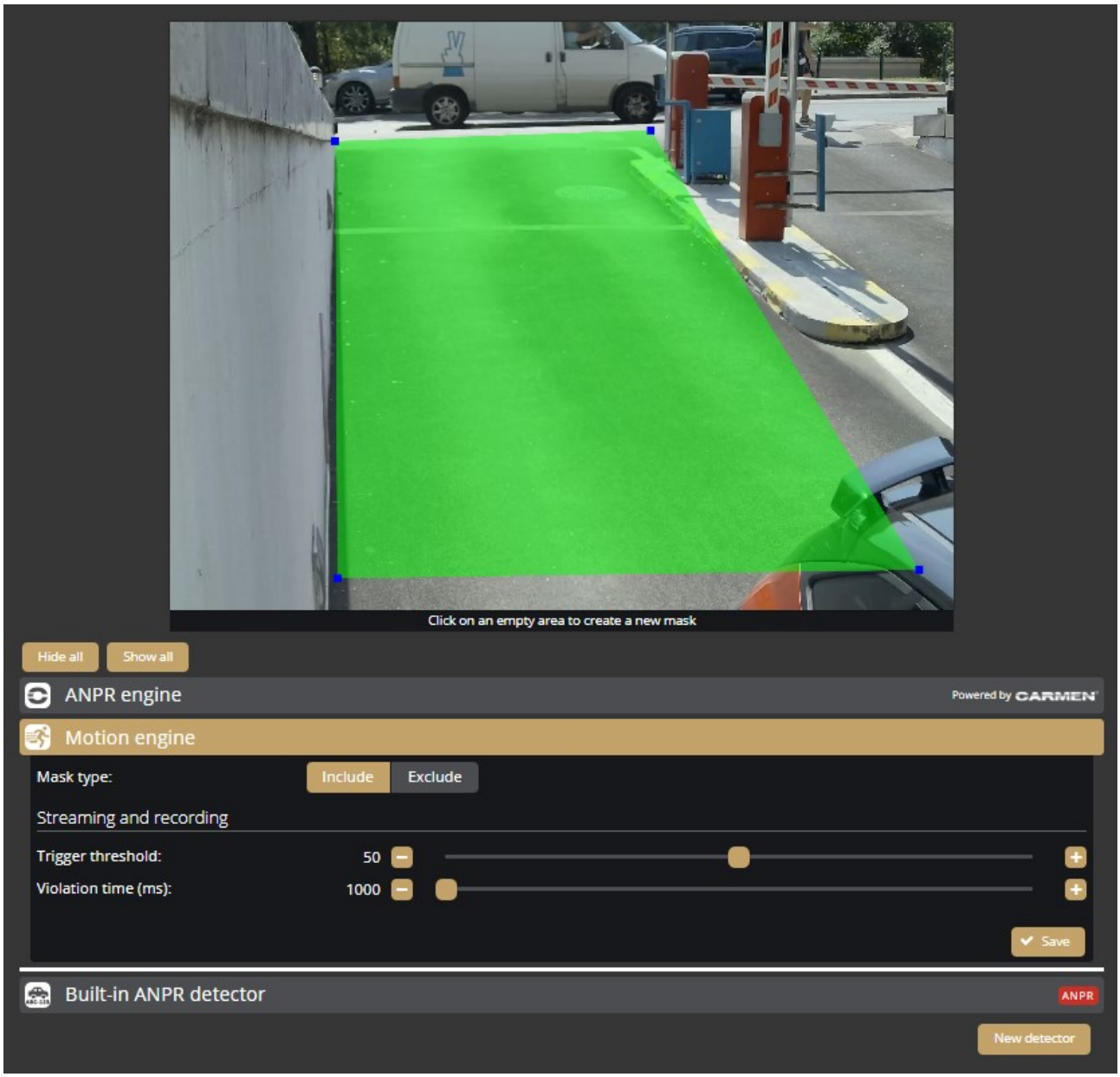
The Motion engine is a fundamental engine that regulates motion-based storage. It cannot be deleted.

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

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

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

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



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

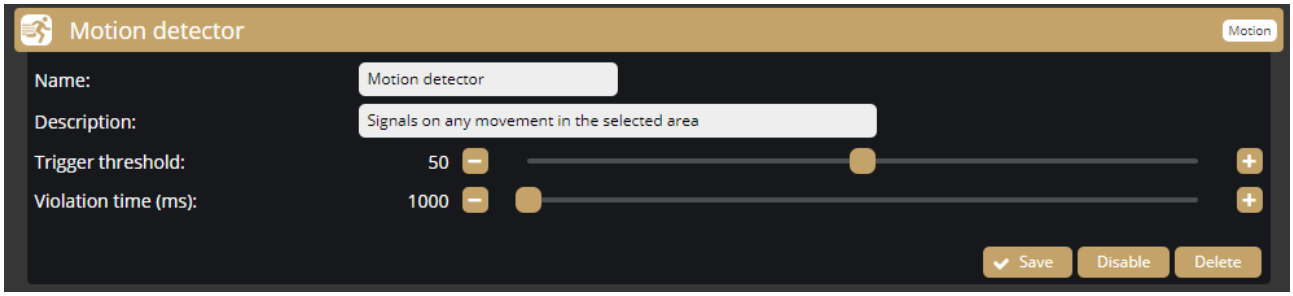
Trigger threshold: You can use it to define the sufficient level of motion in the image to trigger the motion engine. Further filtering can be done with the previously set sensitivity conditions to determine the degree of action intensity triggering recording. The **Motion graph** is the OSD belonging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

Violation time (ms): The time between sensing movement and the alarm event. If the movement stops during the masked area during this time, the alarm will not sound.

7.14.2. MOTION DETECTOR

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

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



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 device 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.

7.14.3. ANPR ENGINE

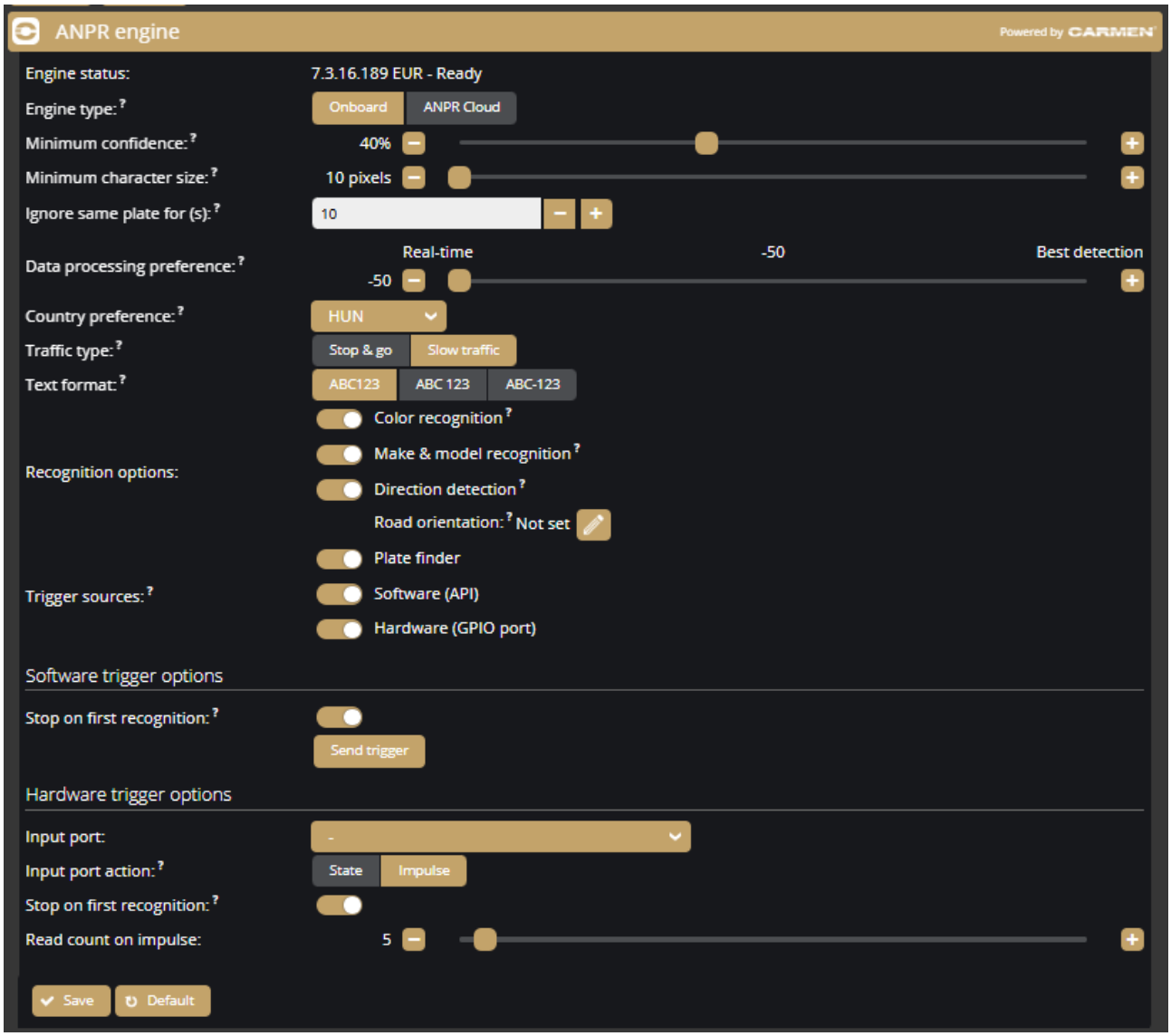
You can choose between the Onboard ANPR Engine running on the device or the Carmen Cloud service for license plate recognition.

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 in 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).

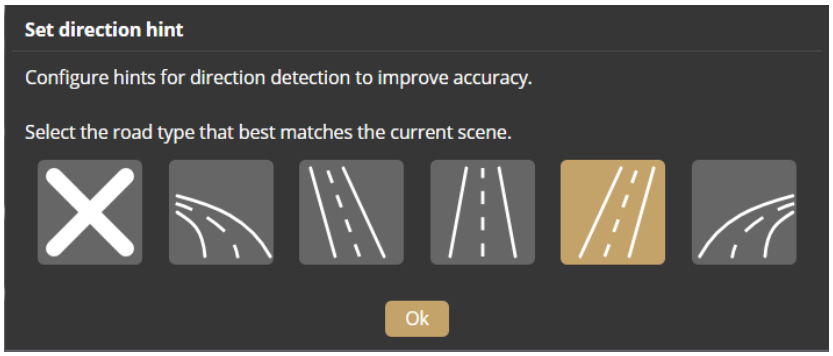


In the case of **Onboard ANPR**, you can adjust the primary functions of the ANPR engine:

- **Engine status:** It shows the current ANPR 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 ***{****Hiba! A hivatkozási forrás nem található.****}***.
- **Minimum character size:** License plates with character size below this threshold are discarded.
- **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.

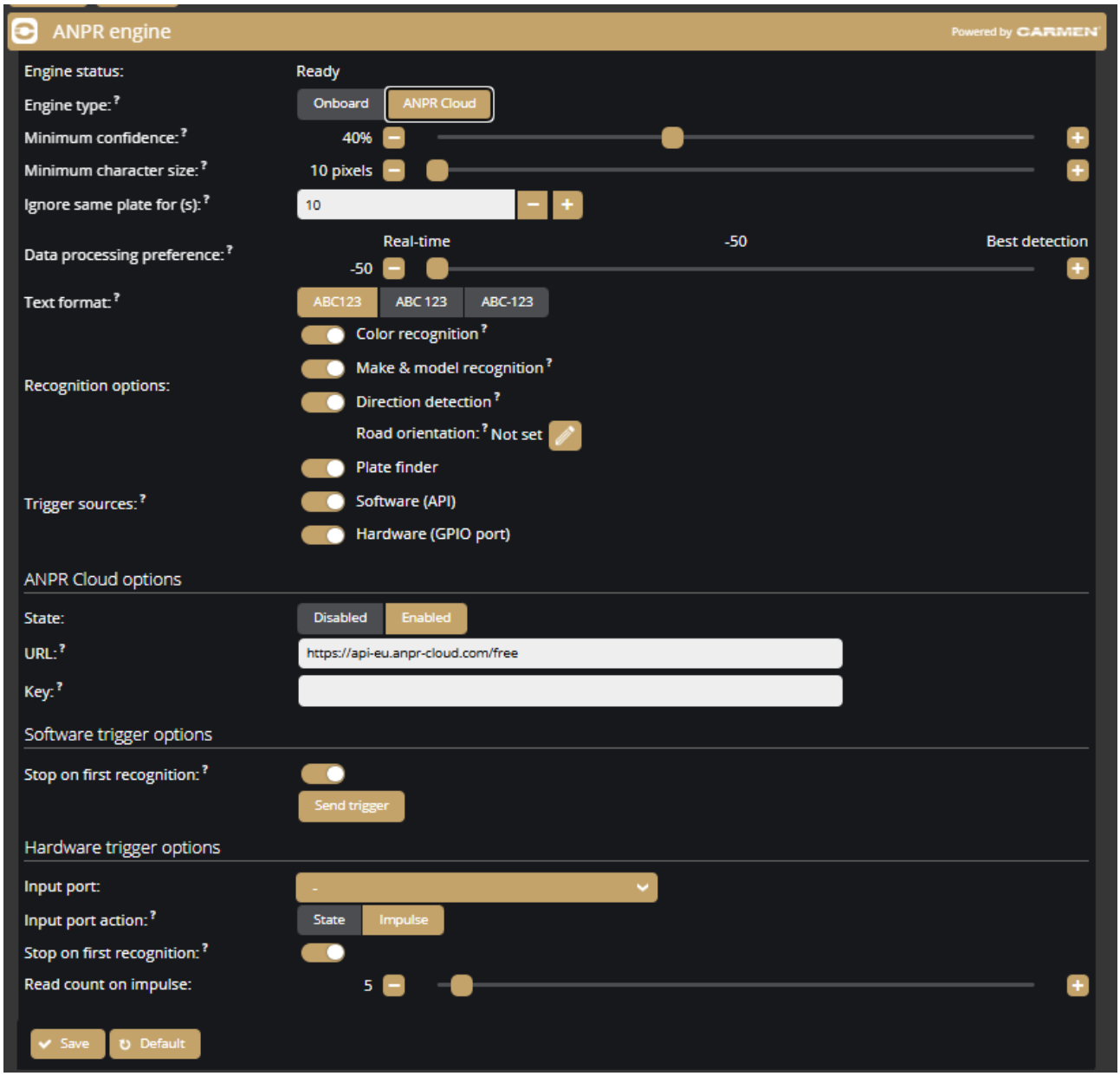
- **Data processing preference:** Configure how the engine processes license plate data.
  Setting the slider towards Real-time will decrease the delay between a vehicle appearing and an event being emitted but also decreases the accuracy of the license plate data. Recommended for quick license plate validation when device is used for controlling vehicle entry points.
  Setting the slider towards Best detection will increase the license plate data precision including vehicle direction and vehicle speed but will dramatically increase the delay between a vehicle appearing and an event being emitted as the engine will spend more time making sure all data are sufficiently validated.\ Recommended when data accuracy is important and events are not used for live triggers but processed later or in bulk.
- **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:** Type of traffic the device processes. Some engines contain more than one recognition possibility. If the uploaded engine supports „Quick" recognition mode, you can select it as „SlowTraffic" mode.
- **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.
- **Text format:** By default license plate characters are returned without spacing. Depending on the selected option symbol groups on a license plate may be separated by whitespaces and/or hypens.
- **Color recognition:** When color recognition is active, the device will attempt to identify the plate and/or vehicle colors. This may slow down the speed of recognition.
- **Make & Model recognition:** When make and model recognition is active, the device will attempt to identify the make, model and color of the vehicle. This may slow down the speed of recognition.

- **Direction detection:** Attempt to detect direction of the vehicle movement by license plate.
  - **Road orientation:** Click on the edit button and set the road type to improve direction detection accuracy.



Important!

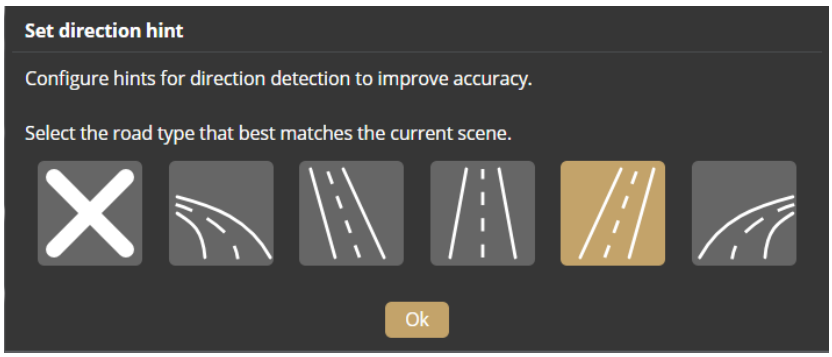
Direction detection is only performed if the same license plate number has been read at least twice by the camera and the height of the license plate number character has changed by at least 2 pixels in the image.


ANPR engine
Powered by CARMEN
Engine status:
Ready
Engine type:
Onboard
ANPR Cloud
Minimum confidence:
40%
Minimum character size:
10 pixels
Ignore same plate for (s):
10
Data processing preference:
Real-time
-50
Best detection
-50
Text format:
ABC123
ABC 123
ABC-123
Recognition options:
Color recognition
Make & model recognition
Direction detection
Road orientation:
Not set
Plate finder
Trigger sources:
Software (API)
Hardware (GPIO port)
ANPR Cloud options
State:
Disabled
Enabled
URL:
https://api-eu.anpr-cloud.com/free
Key:
Software trigger options
Stop on first recognition:
Send trigger
Hardware trigger options
Input port:
Input port action:
State
Impulse
Stop on first recognition:
Read count on impulse:
5
Save
Default

In the case of **Carmen Cloud**, the license plate recognition is not processed in the device. Instead, images with license plates are selected in the device 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/.

- **Engine status:** It shows the current ANPR 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 ***{****Hiba! A hivatkozási forrás nem található.****}***.
- **Minimum character size:** License plates with character size below this threshold are discarded.
- **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.
- **Data processing preference:** Configure how the engine processes license plate data.
  Setting the slider towards Real-time will decrease the delay between a vehicle appearing and an event being emitted but also decreases the accuracy of the license plate data. Recommended for quick license plate validation when device is used for controlling vehicle entry points.
  Setting the slider towards Best detection will increase the license plate data precision including vehicle direction and vehicle speed but will dramatically increase the delay between a vehicle appearing and an event being emitted as the engine will spend more time making sure all data are sufficiently validated.\ Recommended when data accuracy is important and events are not used for live triggers but processed later or in bulk.
- **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:** Type of traffic the device processes. Some engines contain more than one recognition possibility. If the uploaded engine supports „Quick" recognition mode, you can select it as „SlowTraffic" mode.
- **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.
- **Text format:** By default license plate characters are returned without spacing. Depending on the selected option symbol groups on a license plate may be separated by whitespaces and/or hypens.
- **Color recognition:** When color recognition is active, the device will attempt to identify the plate and/or vehicle colors. This may slow down the speed of recognition.
- **Make & Model recognition:** When make and model recognition is active, the device will attempt to identify the make, model and color of the vehicle. This may slow down the speed of recognition.
- **Direction detection:** Attempt to detect direction of the vehicle movement by license plate.
    - **Road orientation:** Click on the edit button and set the road type to improve direction detection accuracy.

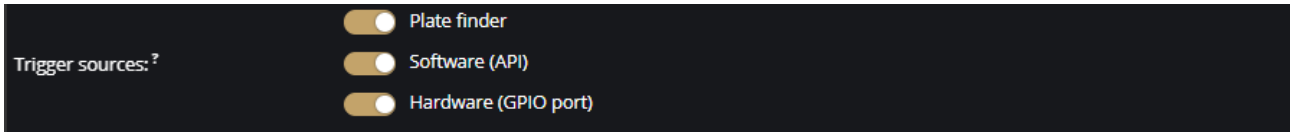


- **ANPR Cloud URL:** Paste the URL you received when subscribing to the field.
- **ANPR Cloud key:** Enter the individual key of the subscription

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.

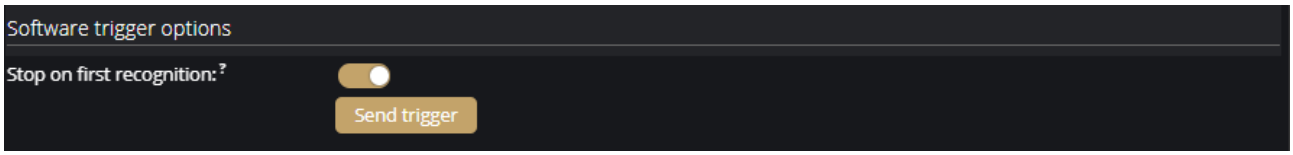


Available trigger modes are:

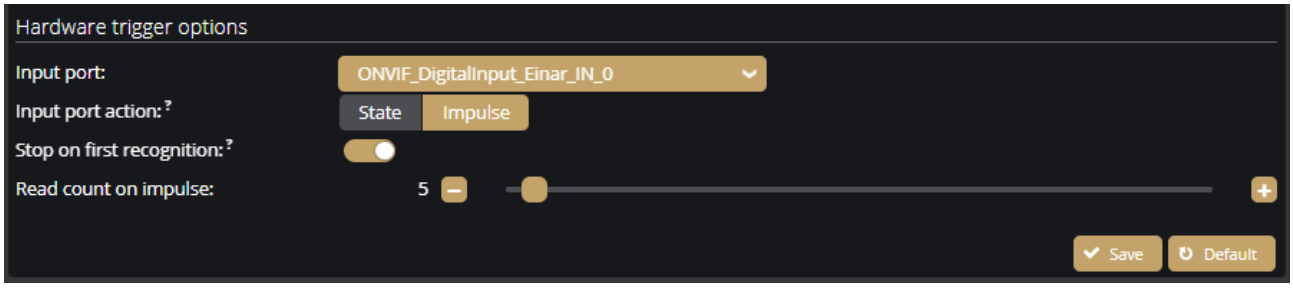
Plate Finder: Engine is triggered automatically by the on-board license plate finder

Software (API): Engine can only be triggered using an API call (Analytics/TriggerEngine). See also API documentation in ***{System / Status}***

Hardware (GPIO port): Input settings for the added ONVIF device can be specified.



Stop on first recognition: Any further license plate detection is stoppped after the first succesful plate detection.



Input port: Select the input port of the added ONVIF device.

Input port action: Activation on input port can trigger detectioan as follows:

- State: License plate detection runs continuously while the port is active.
- Impulse: Activation of the port triggers a set number of the detections. License plate detection runs until the set number of license plates are succesfully recognised.

Stop on first recognition: Any further license plate detection is stoppped after the first succesful plate detection.

Read count on impulse: Specify the number of reads per impulse.

7.14.4. ANPR DETECTOR

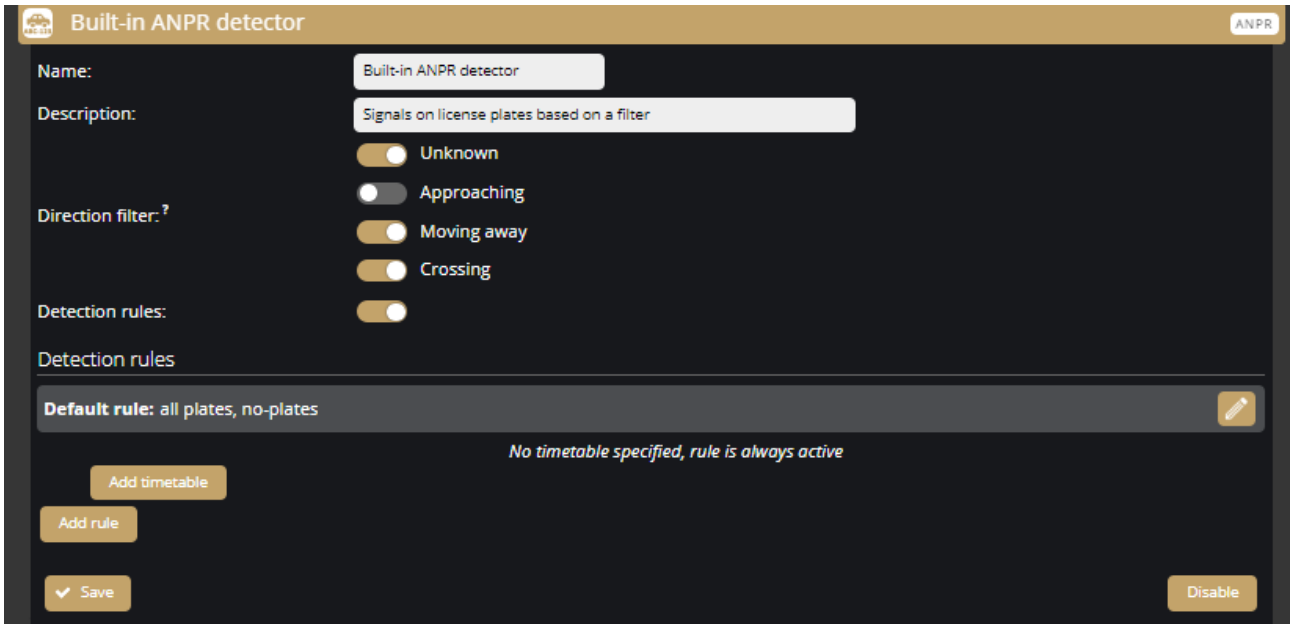
ANPR detector(s) is responsible for creating events from the results of the ANPR Engine.
A factory-set and non-erasable detector, the Built-in ANPR detector, is always present in the device and generates the events of the license plate numbers detected by the ANPR Engine as soon as the device is switched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

Important!

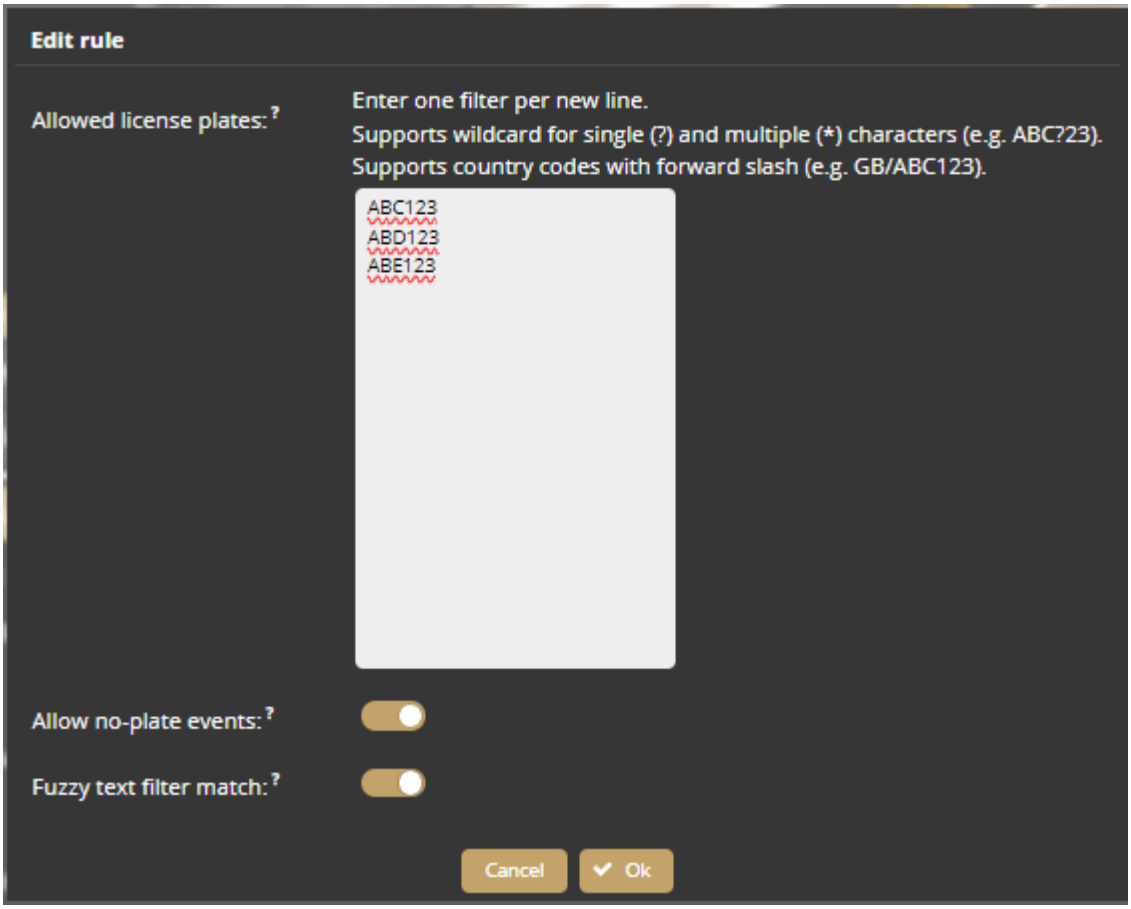
The ANPR detector must be defined separately. The primary functions of the license plate recognition cannot be set under the ANPR detector section, but on the interface of the ANPR engine.

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.
- **Direction filter:** Only emit events of license plate with the selected direction.
- **Detection rules:** If enabled, detection rules can be specified. (White list can be specified.)
  If detection rules are disabled, plate and „no plate" events can be created.



- **Text filter** (White List): You can specify which number plates the camera should create events for, and you can also set time rules. Enable the Text Filter and click on the Edit button (Pencil icon).



  - **Allowed license plates:**

    Only emit events of license plate that matches one of the filters specified here.

    Enter one filter per new line.

    Supports wildcard for single (?) and multiple (*) characters (e.g. ABC?23).

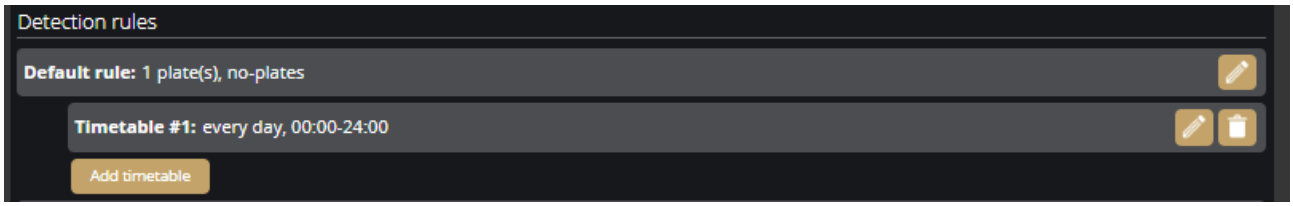
    Supports country codes with forward slash (e.g. GB/ABC123).

  - **Allowed no-plate plates:**

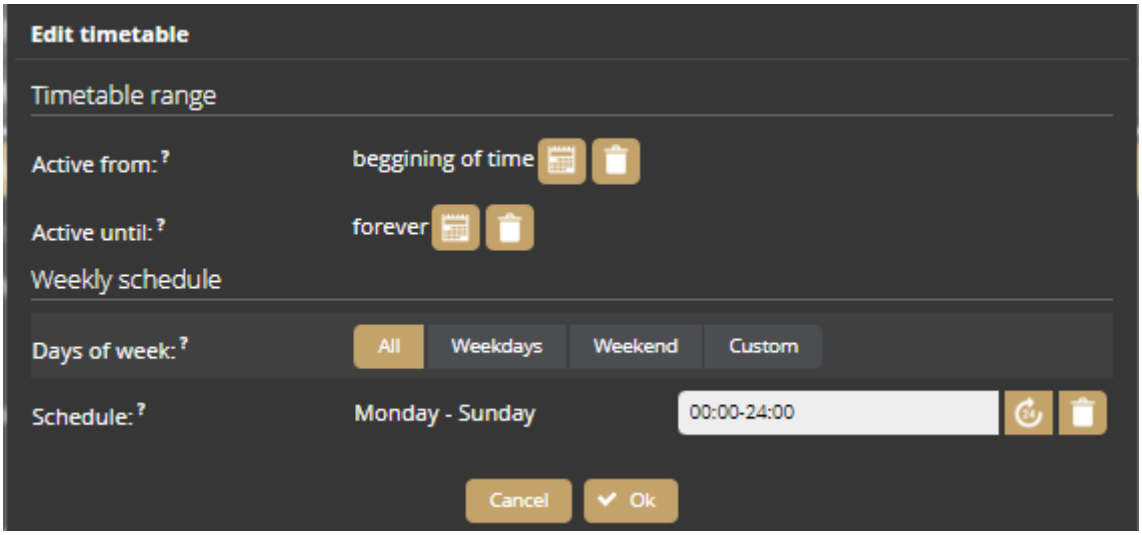
    Allow this detector to emit events of unsuccessful license plate reads.

    Note: This option does not work with plate finder.

  - **Fuzzy text filter 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)



Click Add Timetable to set the time validity of the rule. Multiple timetables can be applied to a rule.



- **Active from:** Specifies the earliest date the timetable is active from
- **Active until:** Specifies the latest date the timetable is active before
- **Days of week:** Select which days the timetable is active on
- **Schedule:** Configure time intervals for the set day. The license plates are recognized only when in a time interval.

  Use HH-HH or HH:MM-HH:MM format to specify a range.

  Only 24-hour clock is supported.

  Multiple intervals may separated by a comma.

  - Examples:

    8-16

    08:00-16:30

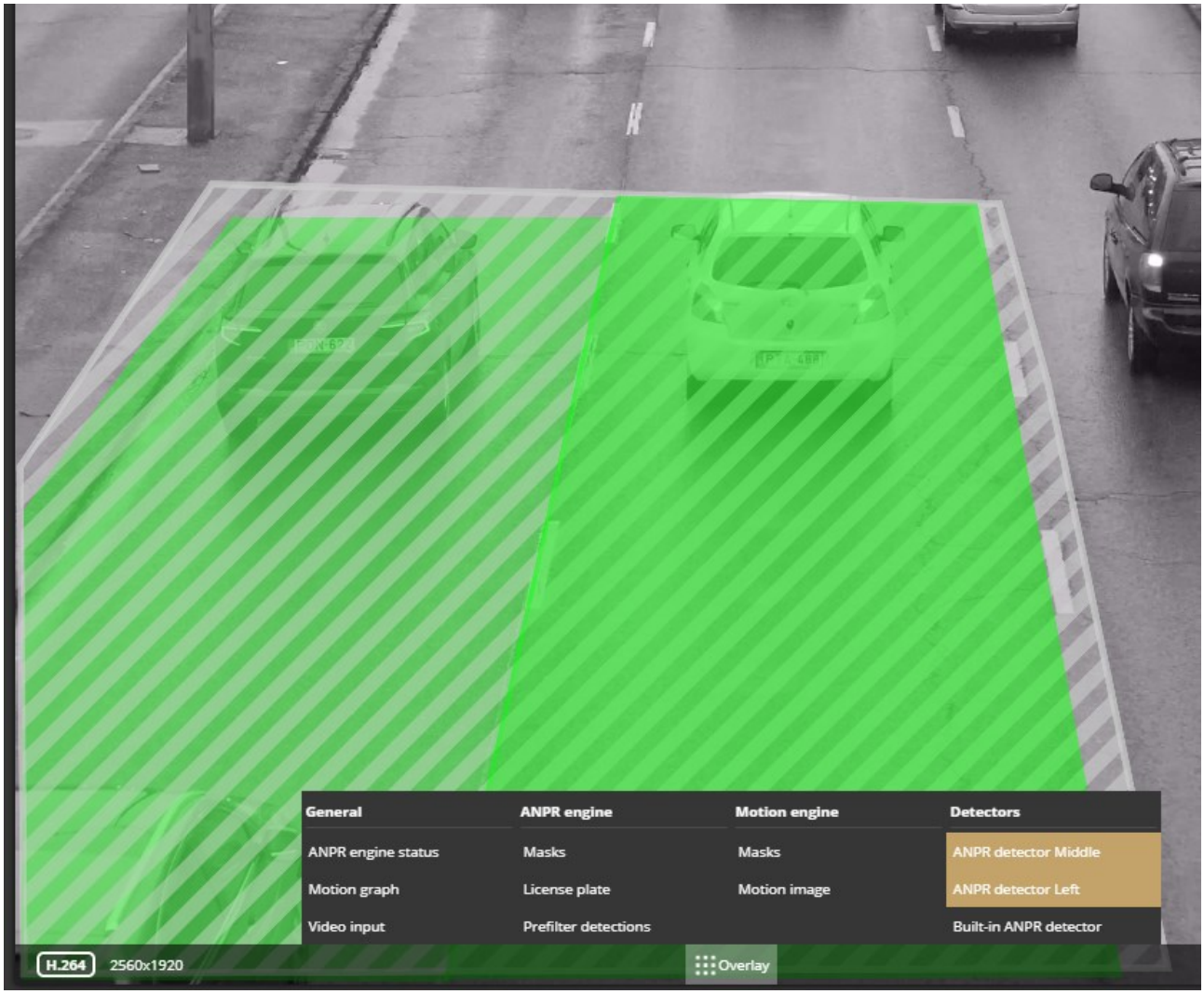
    8-10,14:30-16:30

Multiple rules can be specified for a detector. Click on the Add rules button to add more rules to the detector.

If you want to open a barrier when the above license plates are detected, you have to enable the output port with the above detector in System / I/O menu at Trigger configuration section ***{Hiba! A hivatkozási forrás nem található.}***.

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).

If only one ANPR detector is configured, it is sufficient to specify only the detector mask, it is not necessary to draw a mask for the ANPR engine.

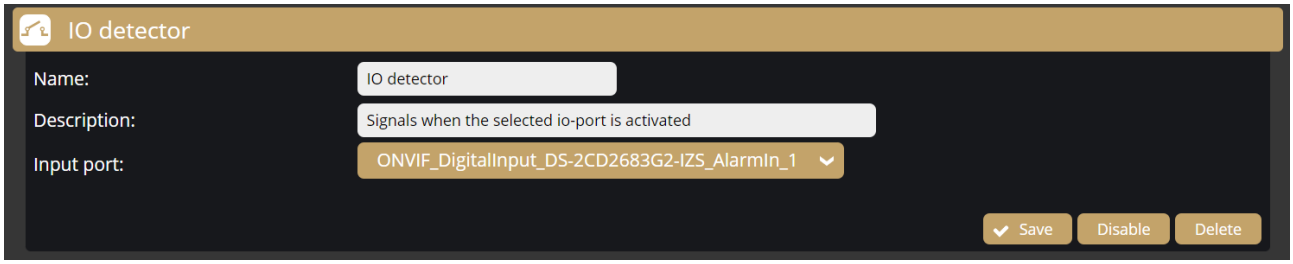


7.14.5. IO DETECTOR

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

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

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

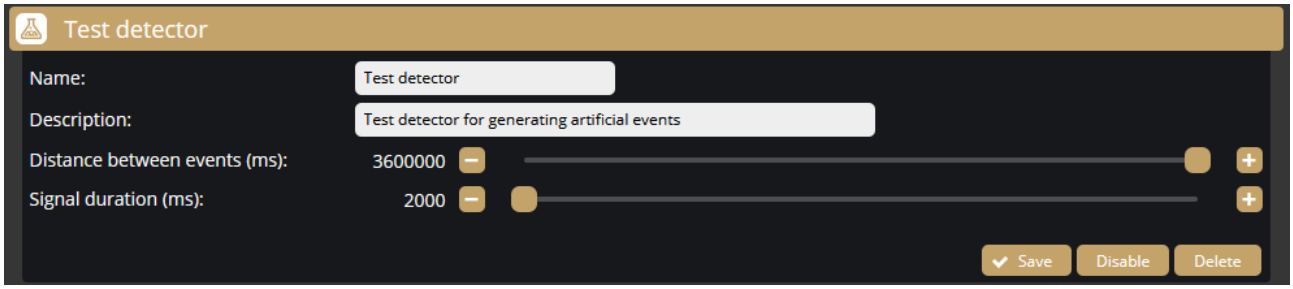


7.14.6. TEST DETECTOR

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

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

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



7.15. ANALYTICS / UPLOADERS

In the Uploaders section, click on the New uploader button to select the uploader option you want to apply.

You can use from the following:

Generic options:

- FTP upload: Upload events to an FTP service
- HTTP upload: Post events as files to an HTTP service

Custom options:

- Auror upload: Post events to an Auror service
- Entervo upload (preview): Post events to a Scheidt & Bachmann entervo service
- Flowbird upload (preview): Post events to a Flowbird service
- GDS upload: Upload events to an Adaptive Recognition Globessey Data Server
- Zatpark upload: Post events to a Unity5 Zatpark service

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

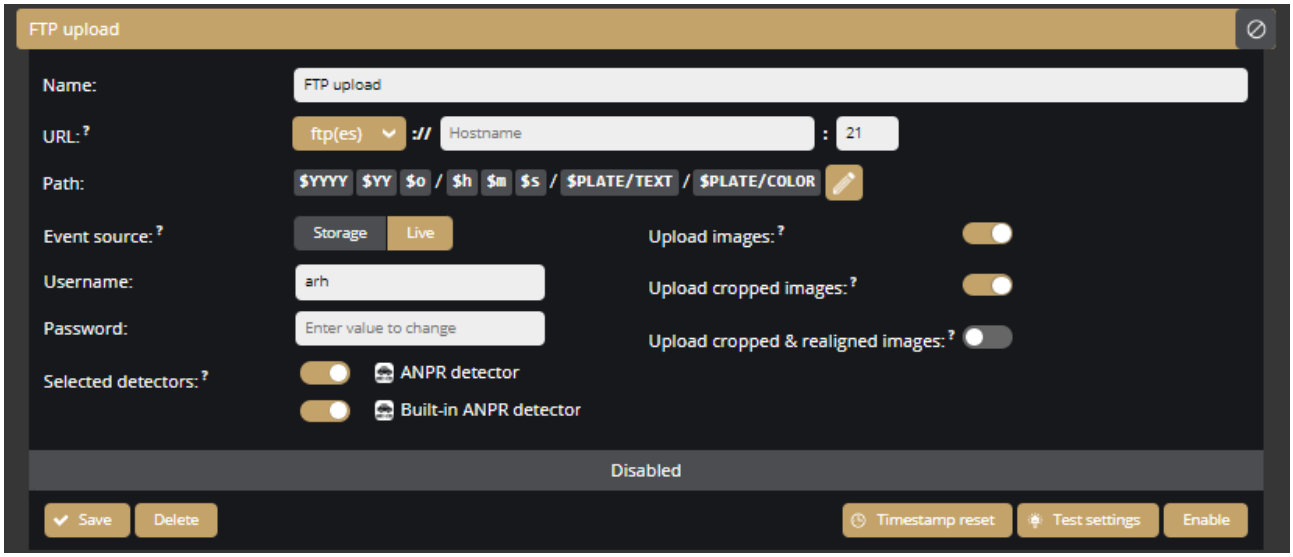
Setting the FTP upload

Select the upload to the FTP, then enter the required data to set the FTP upload. The microSD card is required for FTP upload to work

The following fields should be defined:

- **URL:** Protocol, hostname or IP and port of the FTP service that can accept an FTP upload
    - Protocols: FTP(es), FTPS, SFTP
- **Path:** Path on the server, where the events will be stored. The path support templates for dynamic path generation. Click the Edit button and select the templates you want to use, then click finish editing to close the editing view.
- **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.
- **Username/password:** Username and password required for the identification
- **Upload images:** Optionally upload related images about the event
- **Upload cropped & realigned images:** Optionally upload cropped images abour the event showing an area of interest like a license plate. If the area of interest is distorted it will be realigned as best as possible. In case of a license plate the plate will be transformed to be a rectangle.
- **Upload videos:** Optionally upload a short video depicting the event
- **Selected detectors:** The uploader will only upload events from the selected detectors

With the **Test settings** button you can test the connection between the camera and the FTP server.
In the **Uploader status** section, you can view the status and the data of the uploader.
Timestamp reset: This button forces the uploader to skip all earlier data and continue with the latest record.



Setting the HTTP(S) upload

Select the upload to the HTTP, 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 following fields should be defined:

- **URL:** Protocol, hostname or IP, optional port and path of the service that can accept a HTTP upload.
- **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.
- **Upload images:** Optionally upload related images about the event
- **Upload cropped images:** Optionally upload cropped images about the event showing an area of interest like a license plate.

- **Upload cropped & realigned images:** Optionally upload cropped images abour the event showing an area of interest like a license plate. If the area of interest is distorted it will be realigned as best as possible. In case of a license plate the plate will be transformed to be a rectangle.
- **Upload videos:** Optionally upload a short video depicting the event
- **Media content sent with:** Media attachment may be uploaded with only a name or both a name and filename fields. The correct option depends on the capabilities of the used web service.
- **Additional header:** Optional headers to include when posting.
  The format is the standard HTTP header format of **Header-Name: Header value**
- **Selected detectors:** The uploader will only upload events from the selected detectors

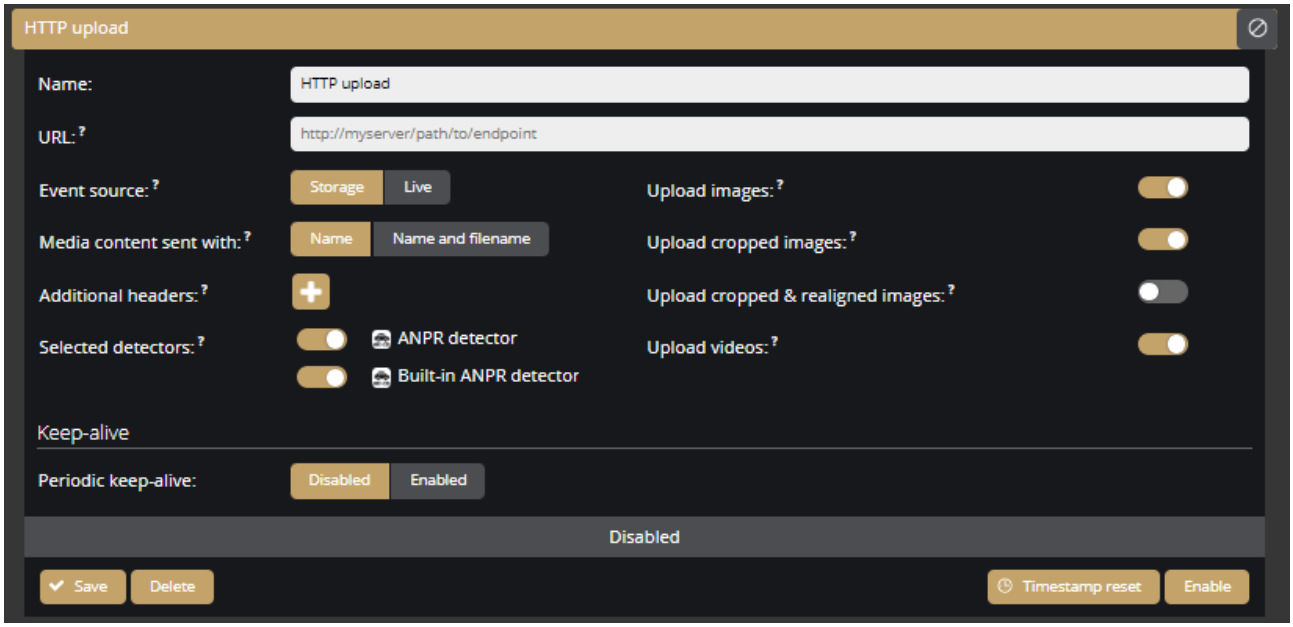
Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

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

Timestamp reset: This button forces the uploader to skip all earlier data and continue with the latest record.

Keep-alive sends a specific message to the specified URL at selected intervals

- **URL:** Protocol, hostname or IP, optional port and path of the service that can accept a HTTP upload.
- **Frequency:** Configured how often a keep-alive message is sent to the server.



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.

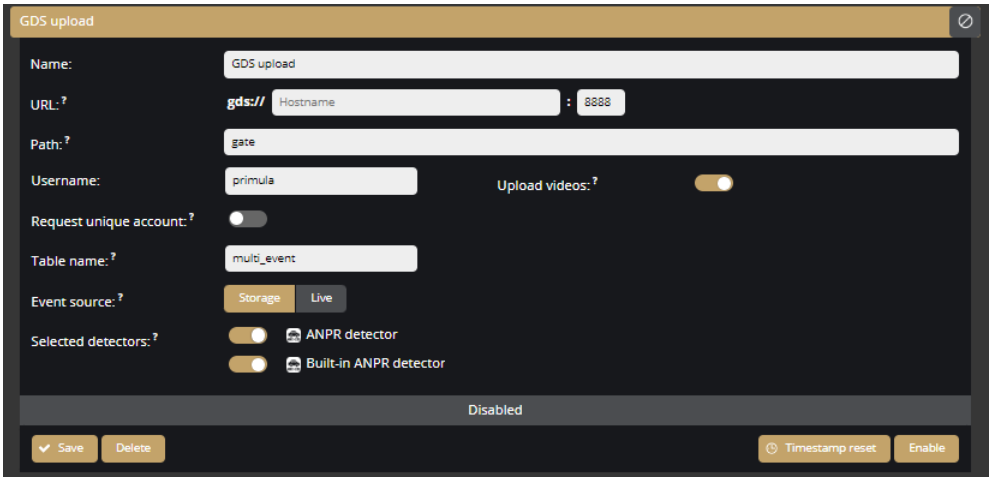
Setting the GDS upload

Select the upload to the GDS (Globessey Data Server), then enter the required data to set the GDS server. The microSD card is required for GDS upload to work.

The following fields should be defined:

- **URL:** Hostname or IP, and path of service on the server of the GDS service
- **Path:** Optional path of the GDS service on the specified server.
- **Table name:** The name of the GDS table where the upload will be done
- **Username:** Username required for the identification
- **Request unique account:** When uploading to GDS the provided username is used for logging in by default. When this setting is enabled the uploader requests a unique device account using the provided username as the requester.
- **Upload videos:** Optionally upload a short video depicting the event
- **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.
- **Selected detectors:** The uploader will only upload events from the selected detectors

In the **Uploader status** section, you can view the status and the data of the uploader.
Timestamp reset: This button forces the uploader to skip all earlier data and continue with the latest record.



Setting the Auror upload

Select the upload to the Auror, then enter the required data to set the Auror upload. The microSD card is required for Auror upload to work.

The following fields should be defined:

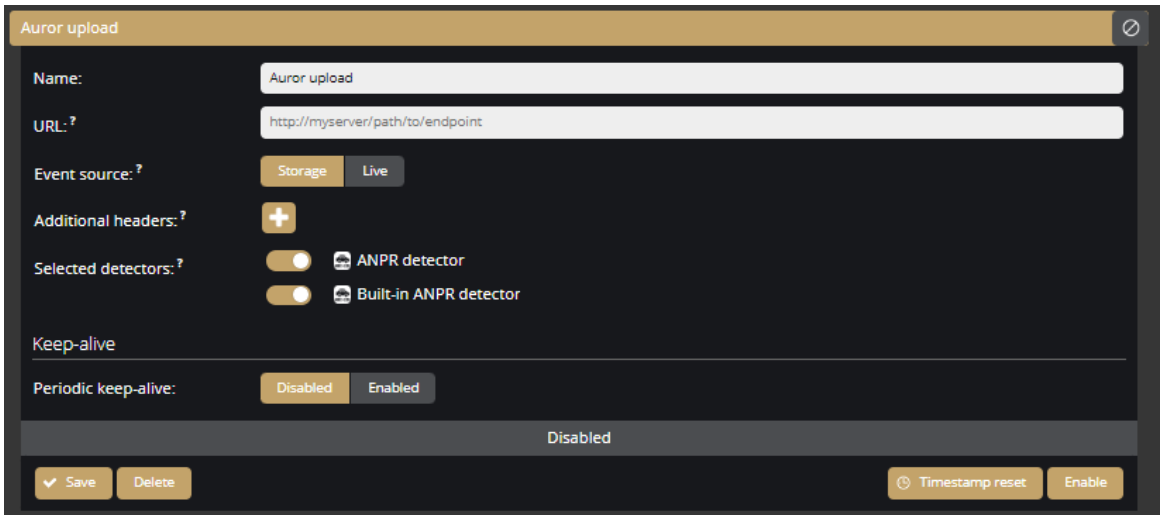
- **URL:** Protocol, hostname or IP, optional port and path of the service that can accept a HTTP upload.
- **Event source:** Selecting the proper event source depend on the usecase.
    - **Storage** requires a storage device with configured on-board storage and uploads are delayed to some extent. This 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.
- **Additional header:** Optional headers to include when posting.
  The format is the standard HTTP header format of **Header-Name: Header value**
- **Selected detectors:** The uploader will only upload events from the selected detectors

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

Timestamp reset: This button forces the uploader to skip all earlier data and continue with the latest record.

Keep-alive sends a specific message to the specified URL at selected intervals

- **URL:** Protocol, hostname or IP, optional port and path of the service that can accept a HTTP upload.
- **Frequency:** Configured how often a keep-alive message is sent to the server.



Setting the Flowbird upload (preview)

Select the upload to the Flowbird upload, then enter the required data to set the Flowbird upload. The microSD card is required for Flowbird upload to work.

The following fields should be defined:

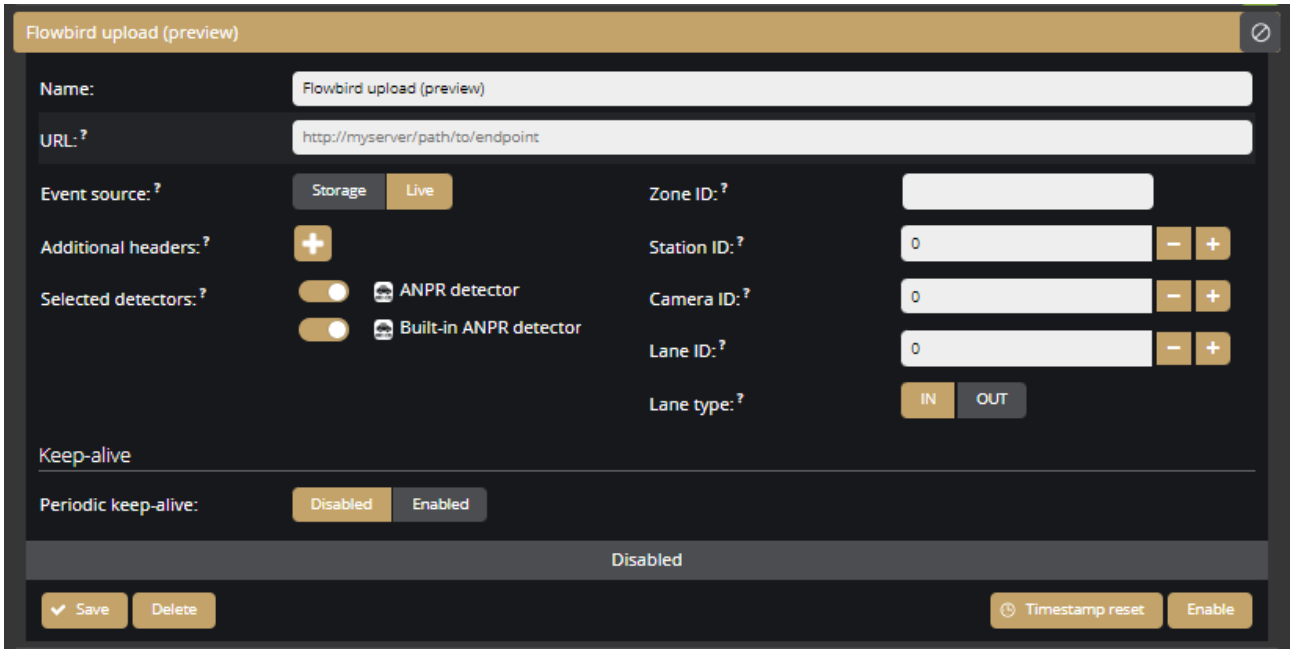
- **URL:** Protocol, hostname or IP, optional port and path of the service that can accept a HTTP upload.
- **Event source:** Selecting the proper event source depend on the usecase.
    - **Storage** requires a storage device with configured on-board storage and uploads are delayed to some extent. This 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.
- **Additional header:** Optional headers to include when posting.
  The format is the standard HTTP header format of **Header-Name: Header value**
- **Selected detectors:** The uploader will only upload events from the selected detectors
- **Zone ID:** Unique ID of the zone the camera is monitoring
- **Station ID:** Unique ID of the station the camera is installed at
- **Camera ID:** Station-wide unique ID of this camera
- **Lane ID:** Station-wide unique ID of the lane the camera is monitoring
- **Lane Type**: Excepted traffic direction on the lane the camera is monitoring

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

Timestamp reset: This button forces the uploader to skip all earlier data and continue with the latest record.

Keep-alive sends a specific message to the specified URL at selected intervals

- **URL:** Protocol, hostname or IP, optional port and path of the service that can accept a HTTP upload.
- **Frequency:** Configured how often a keep-alive message is sent to the server.



Setting the Entervo upload (preview)

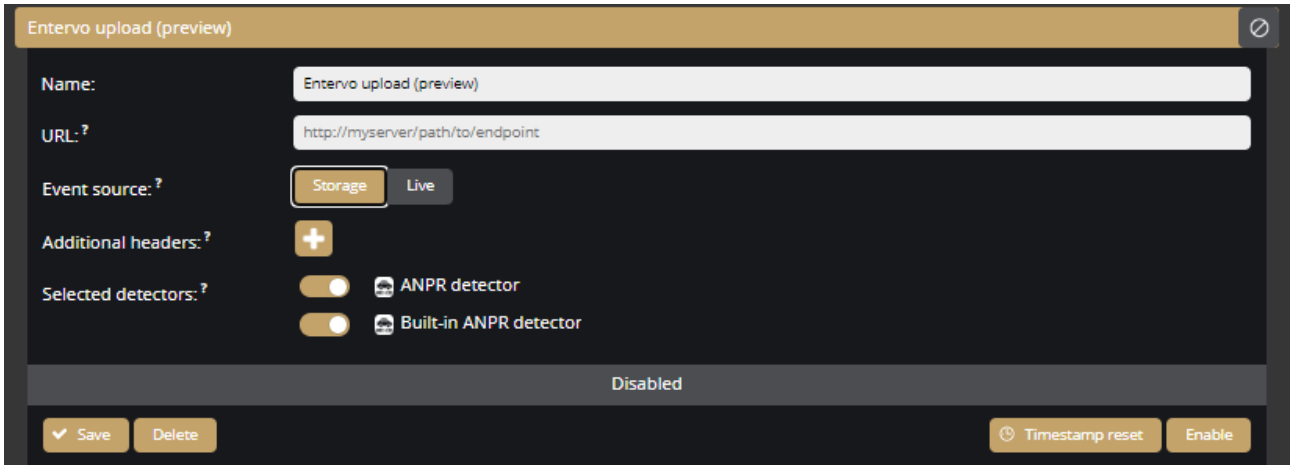
Select the upload to the Entervo, then enter the required data to set the Entervo upload. The microSD card is required for Auror upload to work.

The following fields should be defined:

- **URL:** Protocol, hostname or IP, optional port and path of the service that can accept a HTTP upload.
- **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.
- **Additional header:** Optional headers to include when posting.
  The format is the standard HTTP header format of **Header-Name: Header value**
- **Selected detectors:** The uploader will only upload events from the selected detectors

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

Timestamp reset: This button forces the uploader to skip all earlier data and continue with the latest record.



Setting the Zatpark upload

Select the upload to the Zatpark, then enter the required data to set the Zatpark upload. The microSD card is required for Auror upload to work.

The following fields should be defined:

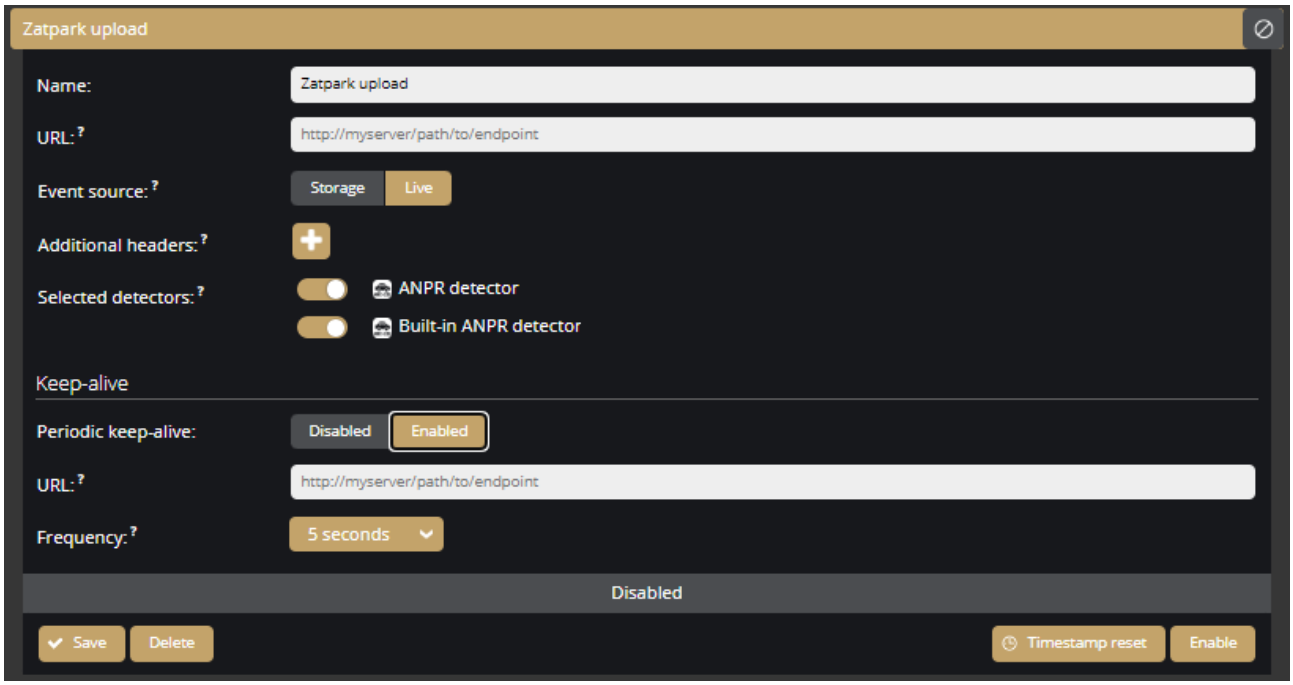
- **URL:** Protocol, hostname or IP, optional port and path of the service that can accept a HTTP upload.
- **Event source:** Selecting the proper event source depend on the usecase.
    - **Storage** requires a storage device with configured on-board storage and uploads are delayed to some extent. This 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.
- **Additional header:** Optional headers to include when posting.
  The format is the standard HTTP header format of **Header-Name: Header value**
- **Selected detectors:** The uploader will only upload events from the selected detectors

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

Timestamp reset: This button forces the uploader to skip all earlier data and continue with the latest record.

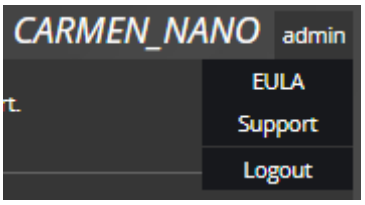
Keep-alive sends a specific message to the specified URL at selected intervals

- **URL:** Protocol, hostname or IP, optional port and path of the service that can accept a HTTP upload.
- **Frequency:** Configured how often a keep-alive message is sent to the server.



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.

9. HOW TO USE THE CARMEN NANO

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

9.1. DEVICE INSTALLATION

1. Mount the device into a weatherproof cabinet near the connected camera or place it in your server room. Consider the temperature tolerance and IP protection of your device.
    a. Provide uninterruptible power to the device's **power supply**.
    b. Connect the device to your network.
    c. To store events in the device, use storage media that your device supports, eg. USB flash drive, microSD card, etc.
    d. Plug in your Carmen USB Neural Network Controller into the device, or install Carmen NNC miniPCIe card before switching on the device.
2. **Find the device on the network**, then access the web interface of the device.
    a. From the MAC address of the device, 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 ***{****Hiba! A hivatkozási forrás nem található.****}***. However, if a DHCP server is available on the network, the device will also get an IP address from the DHCP server. You can access the device from both IP addresses.
    B. Optionally, you can use the **AR Device Tool** to locate the device on your local network ***{AR Device Tool***

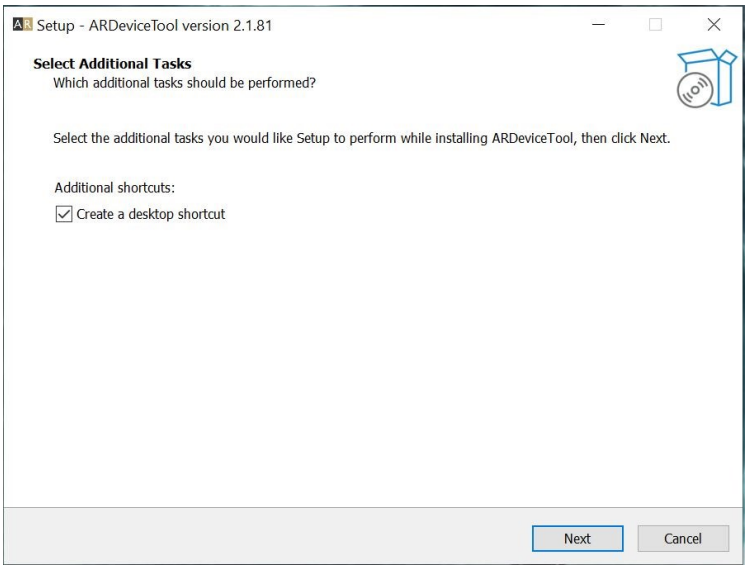
With the AR Device Tool, you can discover Einar or Visus cameras, Carmen Box; Carmen Nano or Enforce Box devices on the local network. You can upload Firmware, License and Engine files to these cameras/devices.

The AR Device Tool also finds all AR cameras (Vidar, MicroCAM, SmartCAM) via ONVIF protocol.

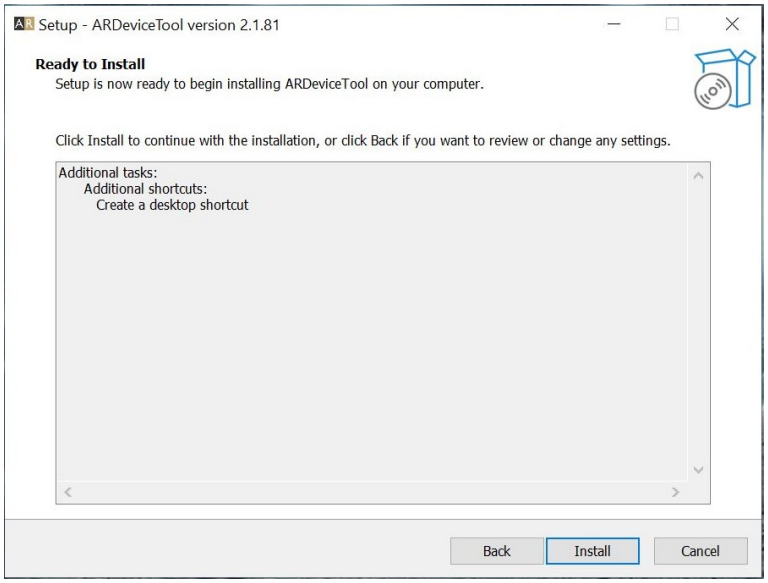
Download the program here: AR DeviceTool.

9.2. AR DEVICE TOOL SETUP

After downloading, start the installation file. Select the desktop shortcut you want to use and click Next.

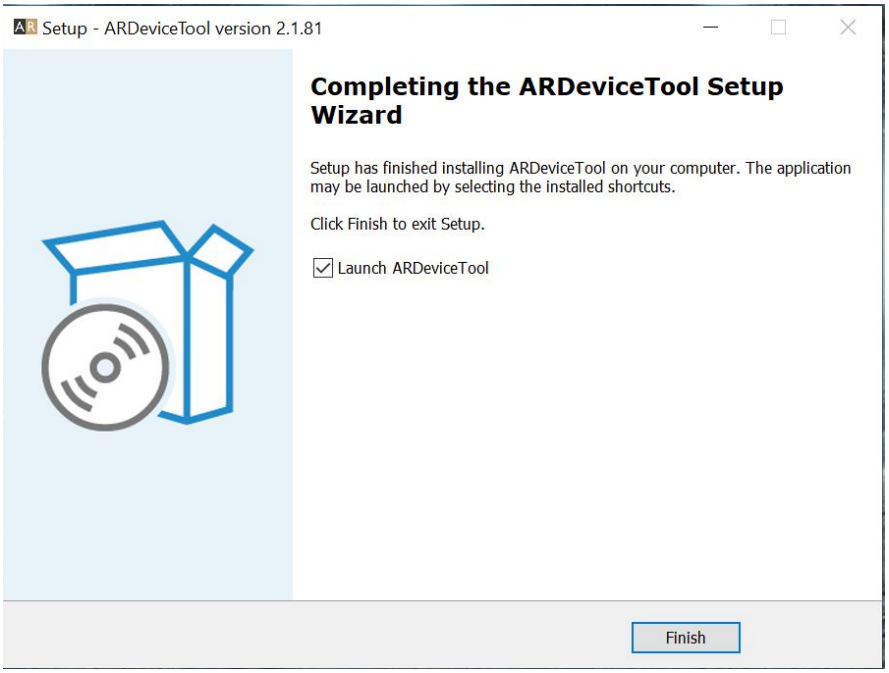


The next step is to start the installation by clicking on the Install button.



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

Select if you want to start ARDeviceTool and click Finish.

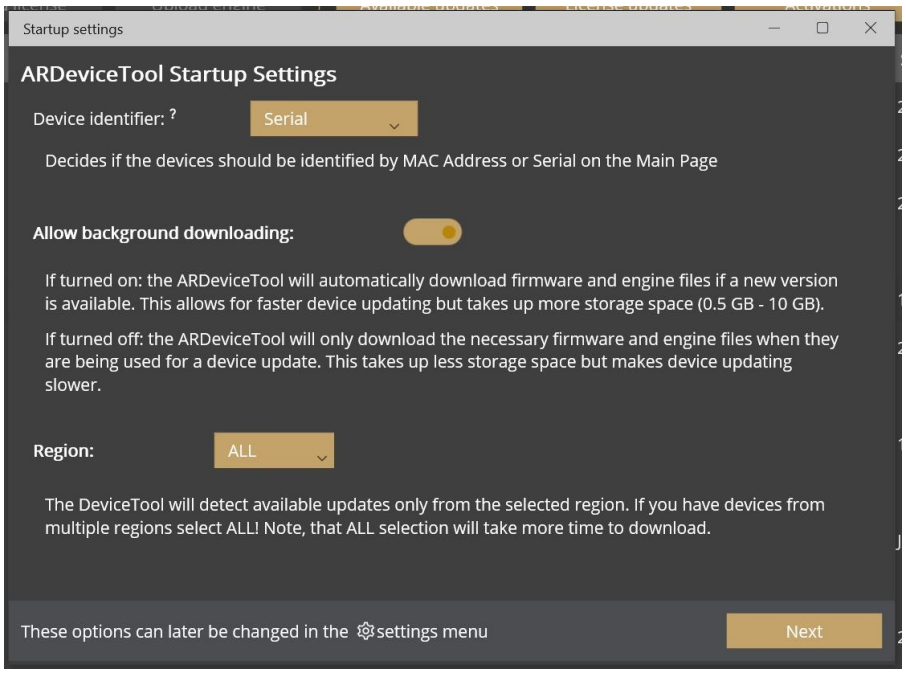


After installation, enable the Device Tool to access networks in the firewall settings. DeviceTool communicates with cameras via HTTP (default port 80).

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

9.3. AR DEVICE TOOL STARTUP SETTINGS

When you start AR DeviceTool for the first time, you can make the initial settings.



Device Identifier: Decides is the devices should be identified by MAC Address or Serial on the Main page.

Allow background:

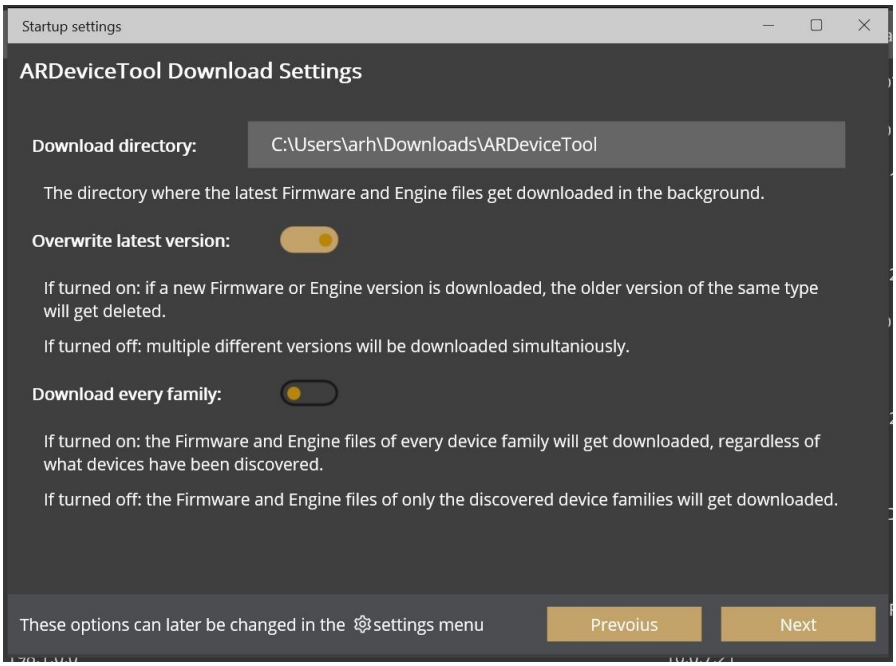
If turned on: the AR DeviceTool will automatically download firmware and engine files if a new version is available. This allows for faster device updating but takes up more storage space (0.5 – 10 GB)

If turned off: the AR DeviceTool will only download the necessary firmware and engine files when they are being used for a device update. This takes up less storage space but makes device updating slower.

Region: The DeviceTool will detect available updates only from the selected region. If you have devices from multiple regions select ALL! Note, that ALL selection will take more time download.

When you have made the settings, click Next.

These options can later be changed in the settings menu.



Download directory: The directory where the latest firmware and engine files det downloaded in the background.

Overwrite latest version:

If turned on: if a new firmware or engine version is downloaded, the older version of the same type will get deleted.

If turned off: multiple different versions will be downloaded simultaniously.

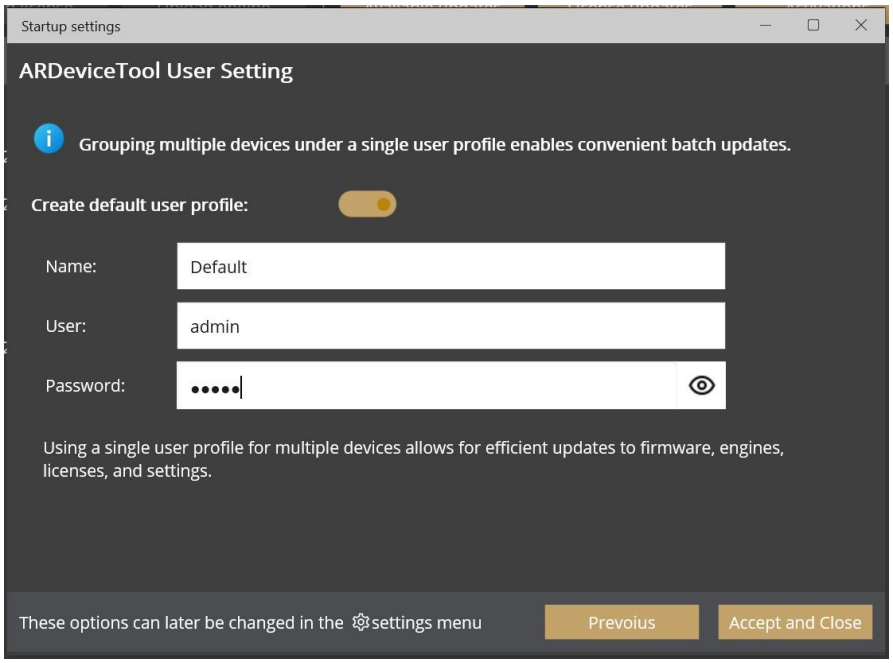
Download every family:

If turned on: the firmware and engine files of every device family will get downloaded, regardless of what devices have been discovered.

If turned off: the firmware and engine files of only discovered device families will get downloaded.

When you have made the settings, click Next.

These options can later be changed in the settings menu.



Create default user profile: Fill in the Name, User and Password fields.

Using a single user profile for multiple devices allows for efficient updates to firmware, engine, license, and settings.

When you have made the settings, click Accept and Close.

These options can later be changed in the settings menu.

9.4. FINDING CAMERAS

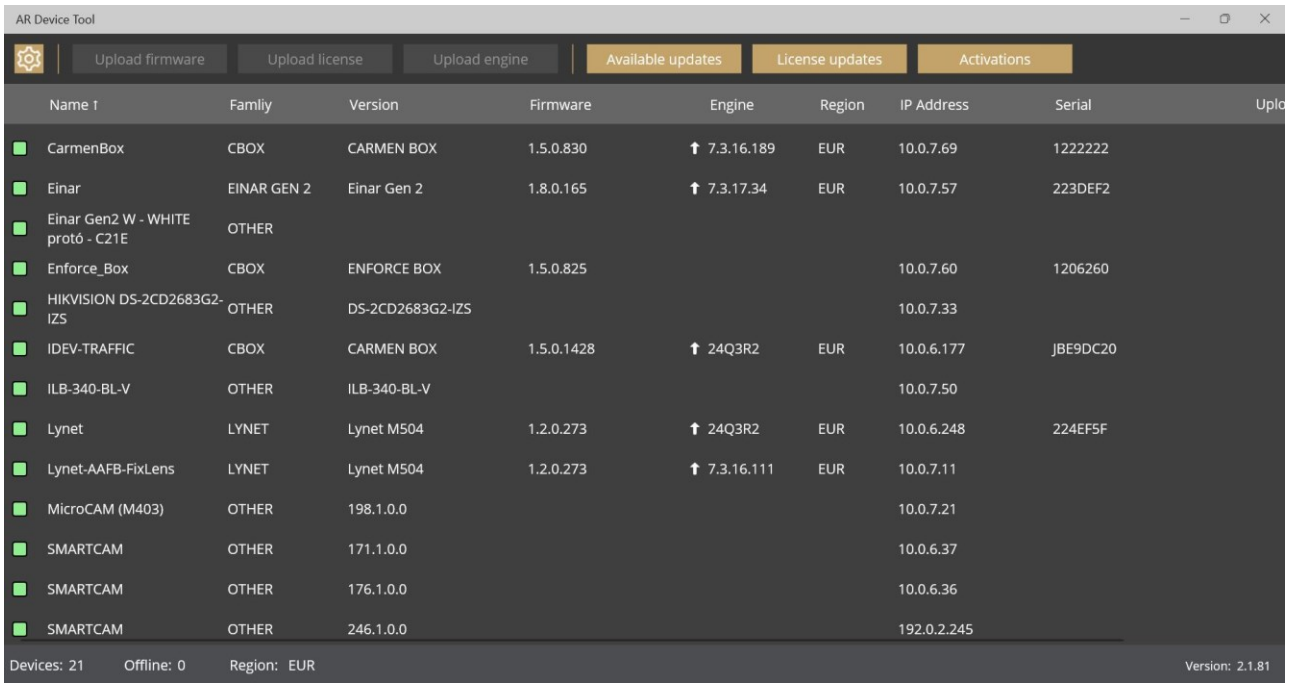
The program lists the AR devices/cameras detected on the local network if the devices/cameras are in the same network segment as the computer. The device/camera name, product family name, type, firmware version, IP address, MAC address and brief information about the current upload process will be displayed.

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, an up arrow will appear in the firmware/engine column.

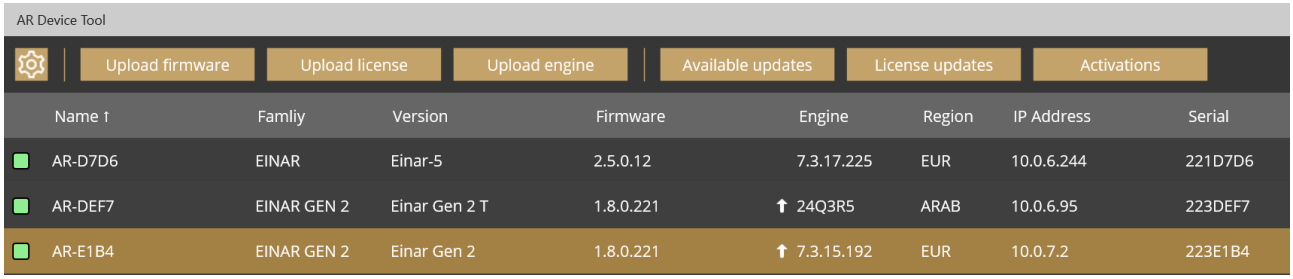
Double-click on the selected device/camera to open its web interface in the default browser.



For Vidar, MicroCAM, SmartCam cameras only Name, Version and IP Address data can be detected by AR DeviceTool, these cameras can not be upgraded using AR DeviceTool.

9.5. FIRMWARE, LICENSE, AND ENGINE UPLOAD - MANUALLY

In addition to find devices/cameras, you can also use the AR 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.

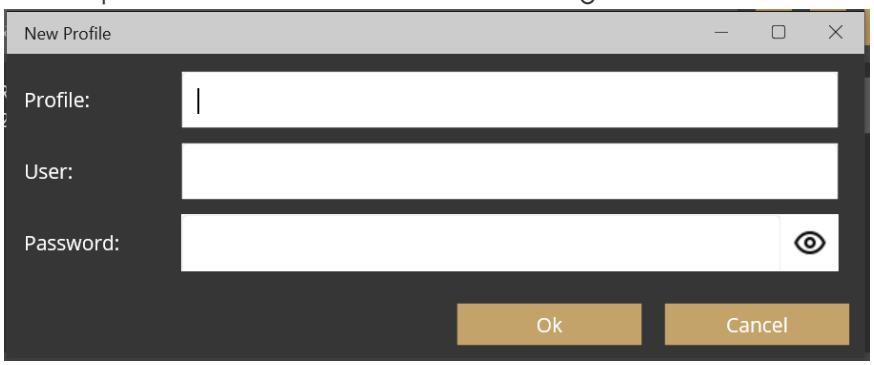


Select the device(s)/camera(s) you want to update and press the **[Upload firmware]**, **[Upload License]** or **[Upload engine]** buttons that become active.
In the window that appears, enter the username and password to access the camera, select the file you want to upload and click **[Start]** to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

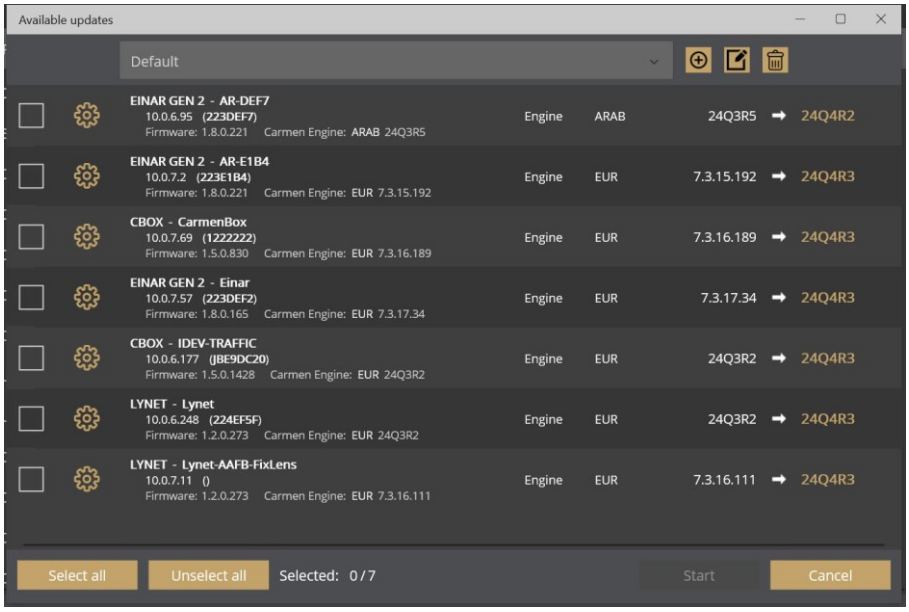
If you want to save the username and password to access the device(s)/camera(s), you can create user profiles. This way, you don't have to enter credentials before each upload.

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.



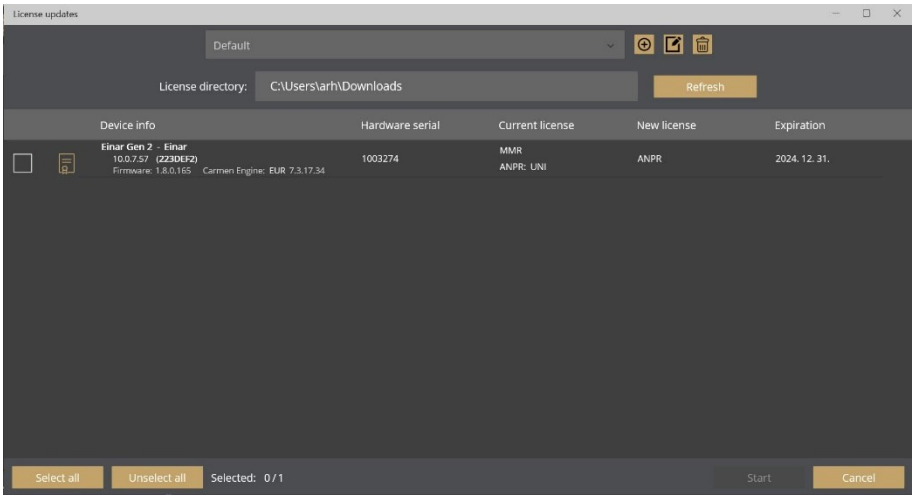
9.6. AVAILABLE UPDATES



AR DeviceTool downloads the appropriate firmware and/or engine from a central server, and uploads it to the selected device(s)/camera(s). A new folder will be created in your Download folder: ArDeviceTool.

Select the cameras you want to update, select the user associated with the cameras, click Start, and then confirm the upload. The cameras will restart during the update.

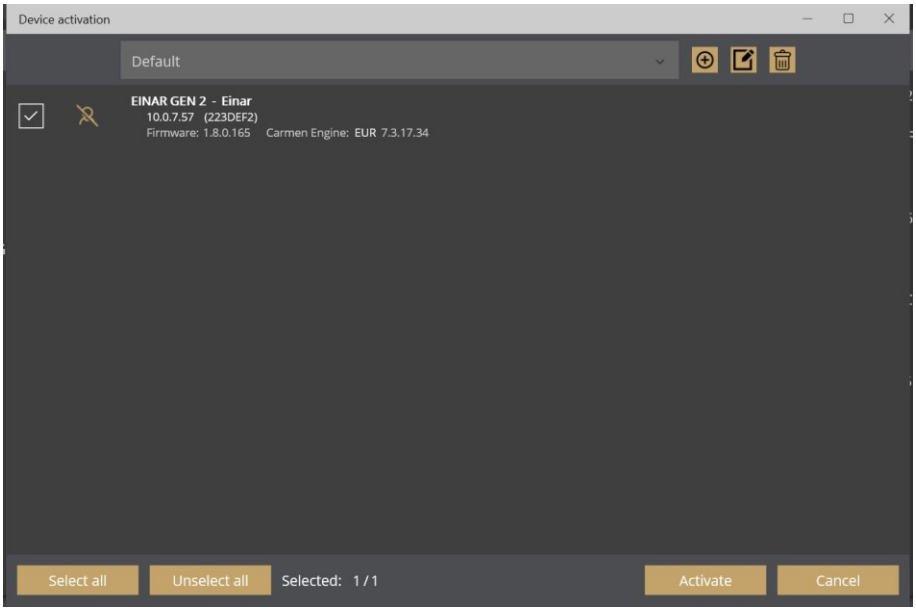
9.7. LICENSE UPDATES



Copy the license to a folder (unzipped) and ARDeviceTool will identify which camera it belongs to. Multiple licenses can be copied to the License directory folder.

Select the cameras you want to update, select the user associated with the cameras, click Start, and then confirm the upload. The cameras will restart during the update.

9.8. DEVICE ACTIVATION



Select the user you want to use, select the cameras you want to activate, then click on the Activate button. This will activate the selected cameras with this user. (This will be the default user.)

c. **}**.

d. You can access the web interface of the device with the **admin/admin** username/password pair. It is strongly recommended to change the default password.

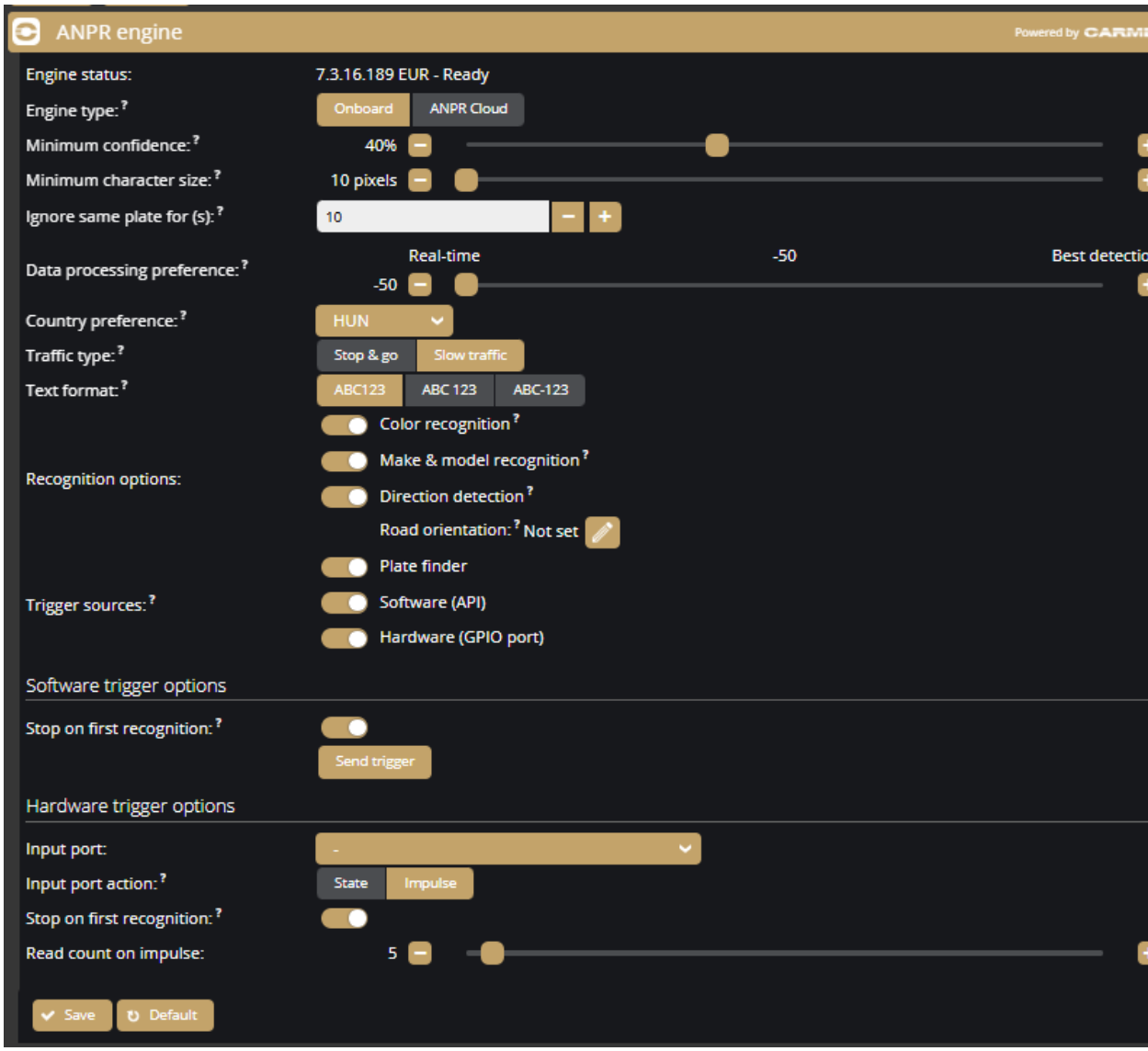
3. Mount your camera or adjust your existing camera as follows:
   a. The maximum angle of rotation and/or tilt of the camera should not exceed **25–30 degrees**.
   b. The optical axis of the camera should **face the license plate** of the oncoming or departing vehicle. The camera's optical axis shall be as small an angle as possible with the axis of travel of passing vehicles.
   c. The camera should be **at least one meter above the headlights** of vehicles. If possible, it should not be accessible without an assistive device.
   d. In the image, **the license plate should be horizontal**.
   e. 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 smaller 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 engine status" OSD layer in Live View and look at the "**Avg. char height:**" to see the character size ***{Overlay}***.
   f. 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.
   g. 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.
4. Some simple but important basic settings:
   a. Upload the **ANPR Engine file, and License Key** to your device ***{System / Device}*** according to the email received when you purchased the software. Check the ANPR Engine quarterly to ensure that the device 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" or „USB key - xxxxxxx") 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 **device has the latest firmware**. You can download the latest FW from Adatptive Recognition website or use Harbard DeviceTool. Upload it in ***{System / Device}***.

c. Check/set the **device time *{System / Device}***.
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 (3) in this chapter}***.
e. With the above steps and proper 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 device to reset internal statistics.

5. 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 your camera **image setting parameters** to as perfect as possible for ANPR. Take into account the location you are observing and, most importantly, make sure the license 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 your 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.
c. Check the settings of the **ANPR engine *{***



In the case of **Onboard ANPR**, you can adjust the primary functions of the ANPR engine:

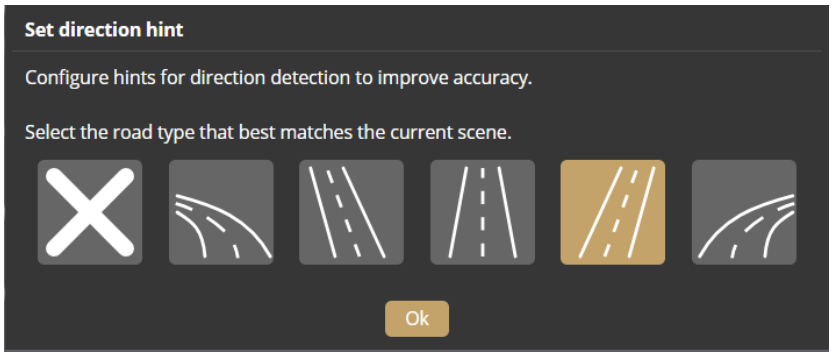
- **Engine status:** It shows the current ANPR version and region.
- **Minimum confidence:** Use the slider to define what reading confidence percentage should trigger an event. A low value potentially results in more reading errors. A high value potentially leads to missed reading cases. Therefore, 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 ***{****Hiba! A hivatkozási forrás nem található.****}***.
- **Minimum character size:** License plates with character size below this threshold are discarded.

- **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.

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

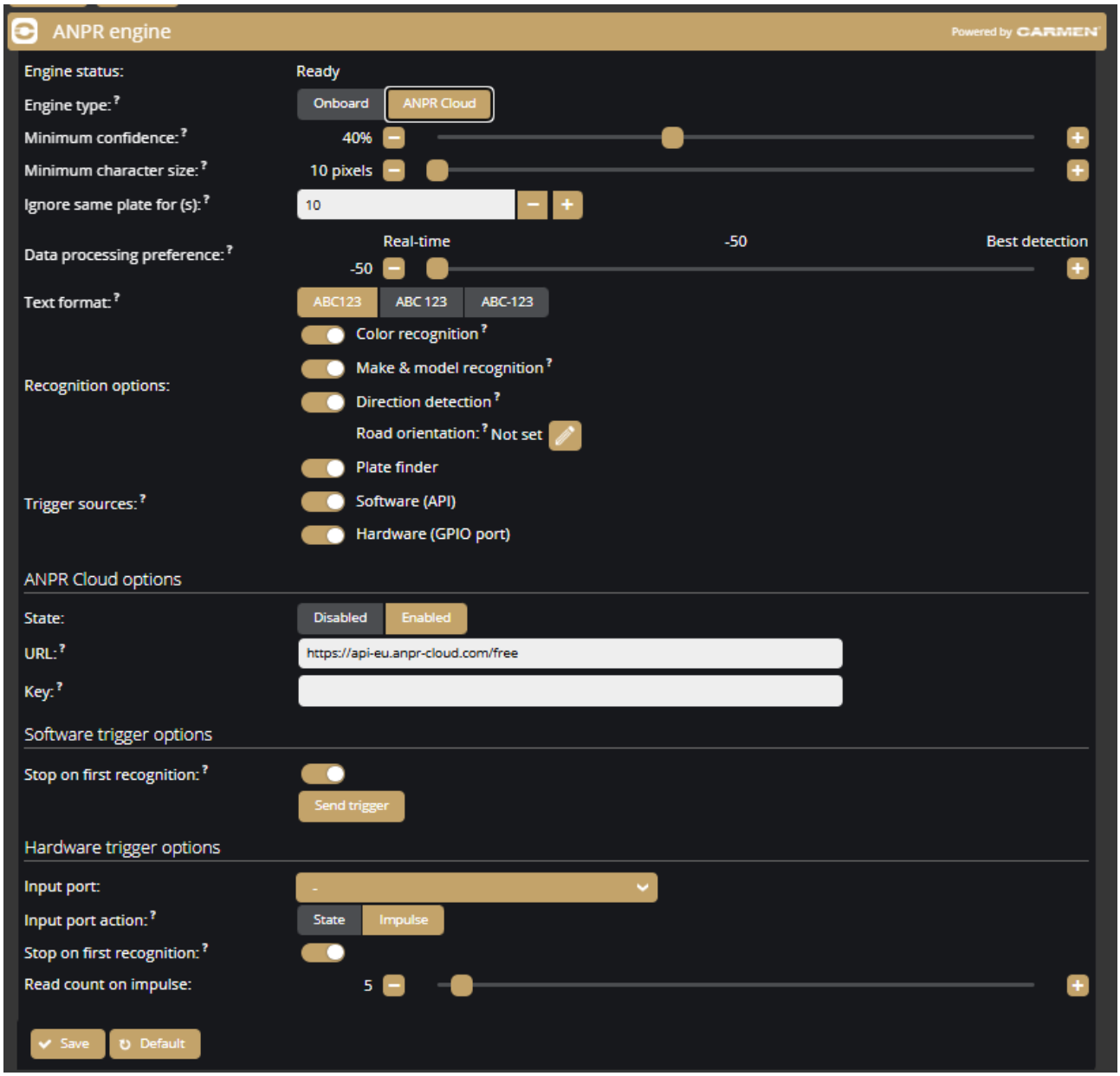
- **Data processing preference:** Configure how the engine processes license plate data.
  Setting the slider towards Real-time will decrease the delay between a vehicle appearing and an event being emitted but also decreases the accuracy of the license plate data. Recommended for quick license plate validation when device is used for controlling vehicle entry points.
  Setting the slider towards Best detection will increase the license plate data precision including vehicle direction and vehicle speed but will dramatically increase the delay between a vehicle appearing and an event being emitted as the engine will spend more time making sure all data are sufficiently validated.\ Recommended when data accuracy is important and events are not used for live triggers but processed later or in bulk.
- **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:** Type of traffic the device processes. Some engines contain more than one recognition possibility. If the uploaded engine supports „Quick" recognition mode, you can select it as „SlowTraffic" mode.
- **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.
- **Text format:** By default license plate characters are returned without spacing. Depending on the selected option symbol groups on a license plate may be separated by whitespaces and/or hypens.
- **Color recognition:** When color recognition is active, the device will attempt to identify the plate and/or vehicle colors. This may slow down the speed of recognition.
- **Make & Model recognition:** When make and model recognition is active, the device will attempt to identify the make, model and color of the vehicle. This may slow down the speed of recognition.

- **Direction detection:** Attempt to detect direction of the vehicle movement by license plate.
  - **Road orientation:** Click on the edit button and set the road type to improve direction detection accuracy.



Important!

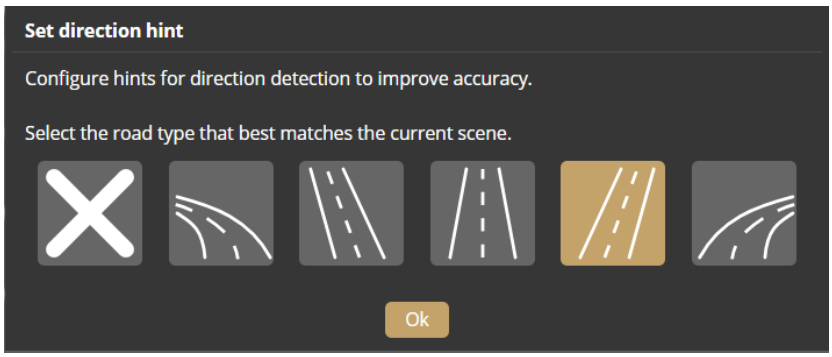
Direction detection is only performed if the same license plate number has been read at least twice by the camera and the height of the license plate number character has changed by at least 2 pixels in the image.


ANPR engine
Powered by CARMEN
Engine status:
Ready
Engine type:
Onboard
ANPR Cloud
Minimum confidence:
40%
Minimum character size:
10 pixels
Ignore same plate for (s):
10
Data processing preference:
Real-time
-50
Best detection
-50
Text format:
ABC123
ABC 123
ABC-123
Recognition options:
Color recognition
Make & model recognition
Direction detection
Road orientation:
Not set
Plate finder
Trigger sources:
Software (API)
Hardware (GPIO port)
ANPR Cloud options
State:
Disabled
Enabled
URL:
https://api-eu.anpr-cloud.com/free
Key:
Software trigger options
Stop on first recognition:
Send trigger
Hardware trigger options
Input port:
Input port action:
State
Impulse
Stop on first recognition:
Read count on impulse:
5
Save
Default

In the case of **Carmen Cloud**, the license plate recognition is not processed in the device. Instead, images with license plates are selected in the device 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/.

- **Engine status:** It shows the current ANPR 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 ***{****Hiba! A hivatkozási forrás nem található.****}***.
- **Minimum character size:** License plates with character size below this threshold are discarded.
- **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.
- **Data processing preference:** Configure how the engine processes license plate data.
  Setting the slider towards Real-time will decrease the delay between a vehicle appearing and an event being emitted but also decreases the accuracy of the license plate data. Recommended for quick license plate validation when device is used for controlling vehicle entry points.
  Setting the slider towards Best detection will increase the license plate data precision including vehicle direction and vehicle speed but will dramatically increase the delay between a vehicle appearing and an event being emitted as the engine will spend more time making sure all data are sufficiently validated.\ Recommended when data accuracy is important and events are not used for live triggers but processed later or in bulk.
- **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:** Type of traffic the device processes. Some engines contain more than one recognition possibility. If the uploaded engine supports „Quick" recognition mode, you can select it as „SlowTraffic" mode.
- **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.
- **Text format:** By default license plate characters are returned without spacing. Depending on the selected option symbol groups on a license plate may be separated by whitespaces and/or hypens.
- **Color recognition:** When color recognition is active, the device will attempt to identify the plate and/or vehicle colors. This may slow down the speed of recognition.
- **Make & Model recognition:** When make and model recognition is active, the device will attempt to identify the make, model and color of the vehicle. This may slow down the speed of recognition.
- **Direction detection:** Attempt to detect direction of the vehicle movement by license plate.
    - **Road orientation:** Click on the edit button and set the road type to improve direction detection accuracy.

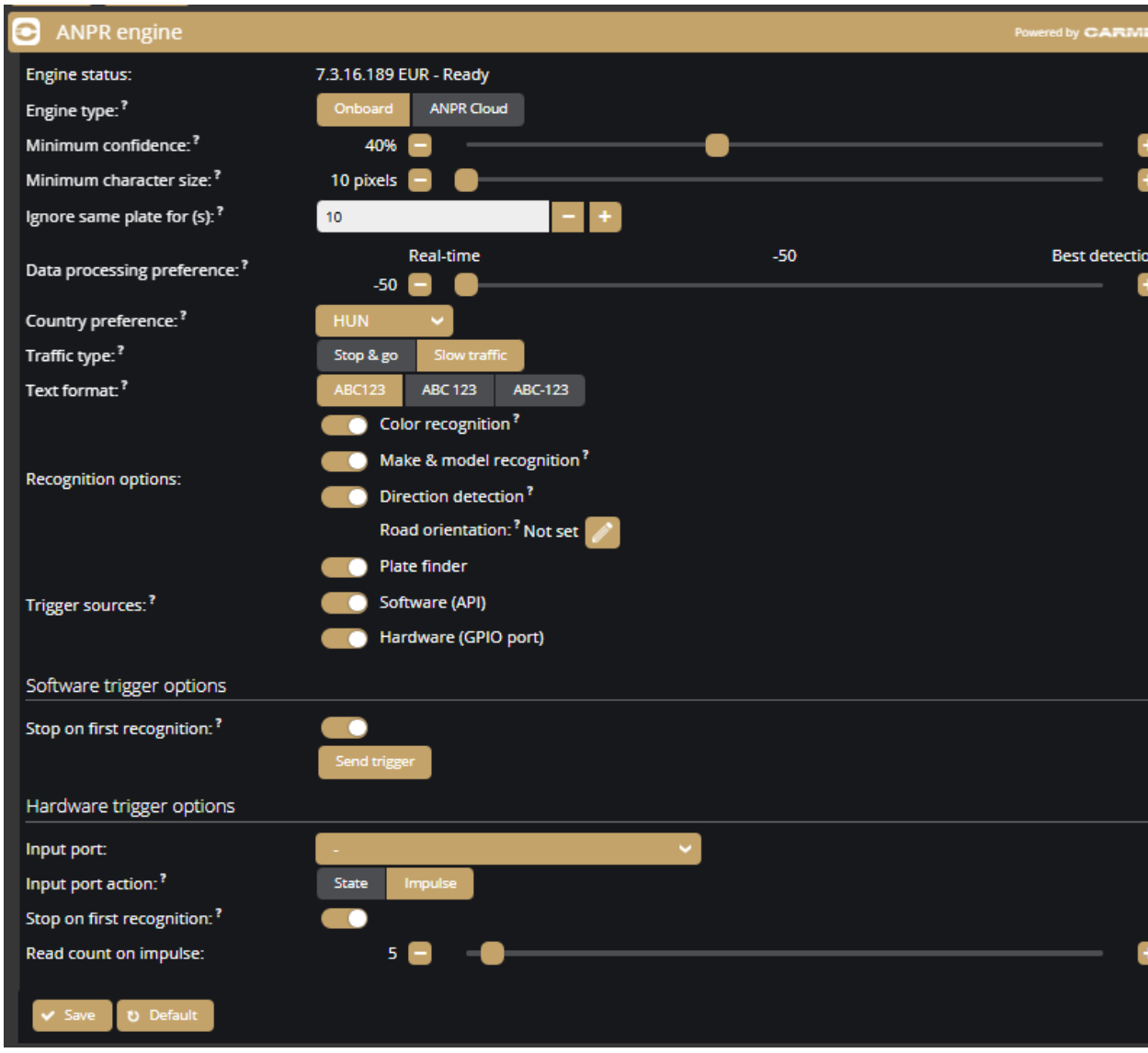


- **ANPR Cloud URL:** Paste the URL you received when subscribing to the field.
- **ANPR Cloud key:** Enter the individual key of the subscription

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

d. }. Lowering the minimum confidence parameter will increase the chance of misreading license plates but will also reduce the probability of missing an unrecognized plate.

e. Set the **country** where the device 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 {



In the case of **Onboard ANPR**, you can adjust the primary functions of the ANPR engine:

- **Engine status:** It shows the current ANPR 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

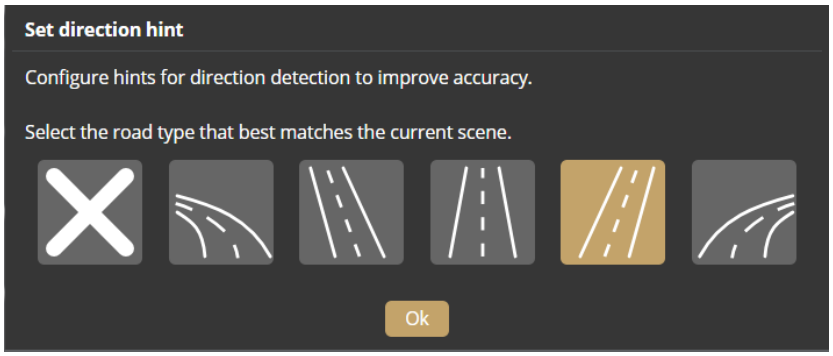
the "ANPR engine status" OSD layer to see the detected license plate numbers and their confidence values and then set the desired minimum confidence level based on the results. See also ***{****Hiba! A hivatkozási forrás nem található.****}***.

- **Minimum character size:** License plates with character size below this threshold are discarded.
- **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.

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

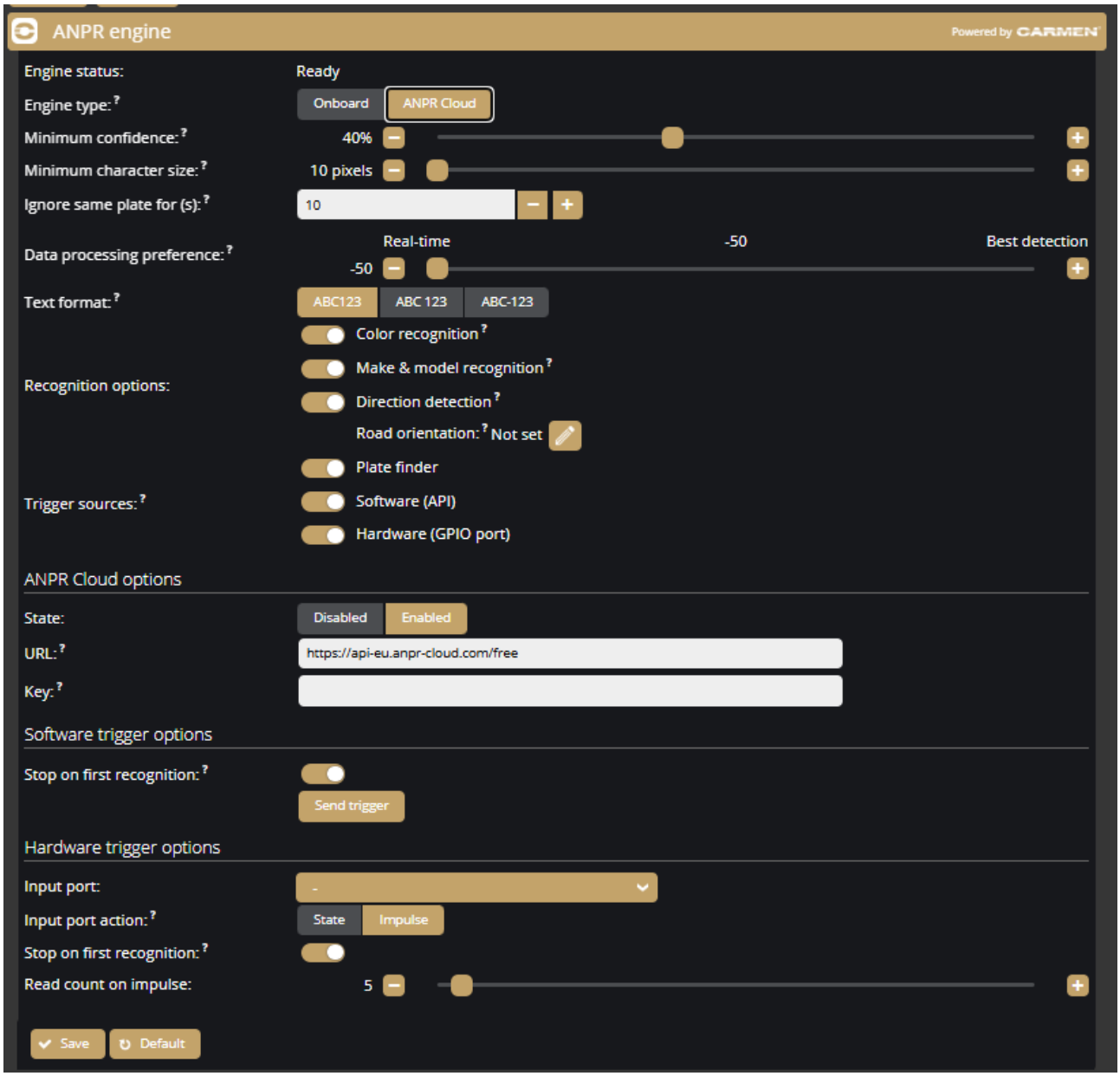
- **Data processing preference:** Configure how the engine processes license plate data.
  Setting the slider towards Real-time will decrease the delay between a vehicle appearing and an event being emitted but also decreases the accuracy of the license plate data. Recommended for quick license plate validation when device is used for controlling vehicle entry points.
  Setting the slider towards Best detection will increase the license plate data precision including vehicle direction and vehicle speed but will dramatically increase the delay between a vehicle appearing and an event being emitted as the engine will spend more time making sure all data are sufficiently validated.\ Recommended when data accuracy is important and events are not used for live triggers but processed later or in bulk.
- **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:** Type of traffic the device processes. Some engines contain more than one recognition possibility. If the uploaded engine supports „Quick" recognition mode, you can select it as „SlowTraffic" mode.
- **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.
- **Text format:** By default license plate characters are returned without spacing. Depending on the selected option symbol groups on a license plate may be separated by whitespaces and/or hypens.
- **Color recognition:** When color recognition is active, the device will attempt to identify the plate and/or vehicle colors. This may slow down the speed of recognition.
- **Make & Model recognition:** When make and model recognition is active, the device will attempt to identify the make, model and color of the vehicle. This may slow down the speed of recognition.

- **Direction detection:** Attempt to detect direction of the vehicle movement by license plate.
  - **Road orientation:** Click on the edit button and set the road type to improve direction detection accuracy.



Important!

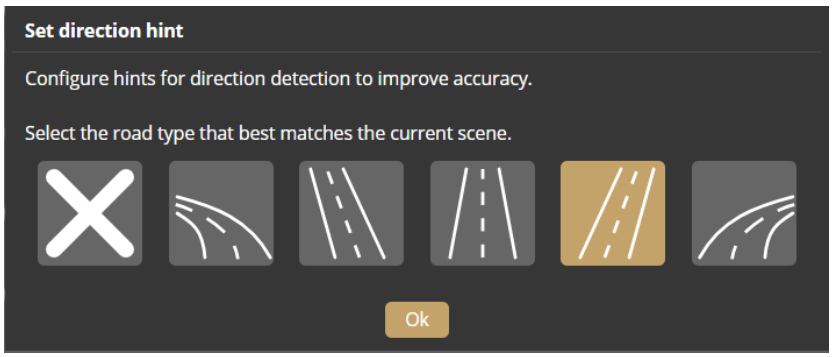
Direction detection is only performed if the same license plate number has been read at least twice by the camera and the height of the license plate number character has changed by at least 2 pixels in the image.


ANPR engine
Powered by CARMEN
Engine status:
Ready
Engine type:
Onboard
ANPR Cloud
Minimum confidence:
40%
Minimum character size:
10 pixels
Ignore same plate for (s):
10
Data processing preference:
Real-time
-50
Best detection
-50
Text format:
ABC123
ABC 123
ABC-123
Recognition options:
Color recognition
Make & model recognition
Direction detection
Road orientation:
Not set
Plate finder
Trigger sources:
Software (API)
Hardware (GPIO port)
ANPR Cloud options
State:
Disabled
Enabled
URL:
https://api-eu.anpr-cloud.com/free
Key:
Software trigger options
Stop on first recognition:
Send trigger
Hardware trigger options
Input port:
Input port action:
State
Impulse
Stop on first recognition:
Read count on impulse:
5
Save
Default

In the case of **Carmen Cloud**, the license plate recognition is not processed in the device. Instead, images with license plates are selected in the device 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/.

- **Engine status:** It shows the current ANPR 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 ***{****Hiba! A hivatkozási forrás nem található.****}***.
- **Minimum character size:** License plates with character size below this threshold are discarded.
- **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.
- **Data processing preference:** Configure how the engine processes license plate data.
  Setting the slider towards Real-time will decrease the delay between a vehicle appearing and an event being emitted but also decreases the accuracy of the license plate data. Recommended for quick license plate validation when device is used for controlling vehicle entry points.
  Setting the slider towards Best detection will increase the license plate data precision including vehicle direction and vehicle speed but will dramatically increase the delay between a vehicle appearing and an event being emitted as the engine will spend more time making sure all data are sufficiently validated.\ Recommended when data accuracy is important and events are not used for live triggers but processed later or in bulk.
- **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:** Type of traffic the device processes. Some engines contain more than one recognition possibility. If the uploaded engine supports „Quick" recognition mode, you can select it as „SlowTraffic" mode.
- **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.
- **Text format:** By default license plate characters are returned without spacing. Depending on the selected option symbol groups on a license plate may be separated by whitespaces and/or hypens.
- **Color recognition:** When color recognition is active, the device will attempt to identify the plate and/or vehicle colors. This may slow down the speed of recognition.
- **Make & Model recognition:** When make and model recognition is active, the device will attempt to identify the make, model and color of the vehicle. This may slow down the speed of recognition.
- **Direction detection:** Attempt to detect direction of the vehicle movement by license plate.
    - **Road orientation:** Click on the edit button and set the road type to improve direction detection accuracy.

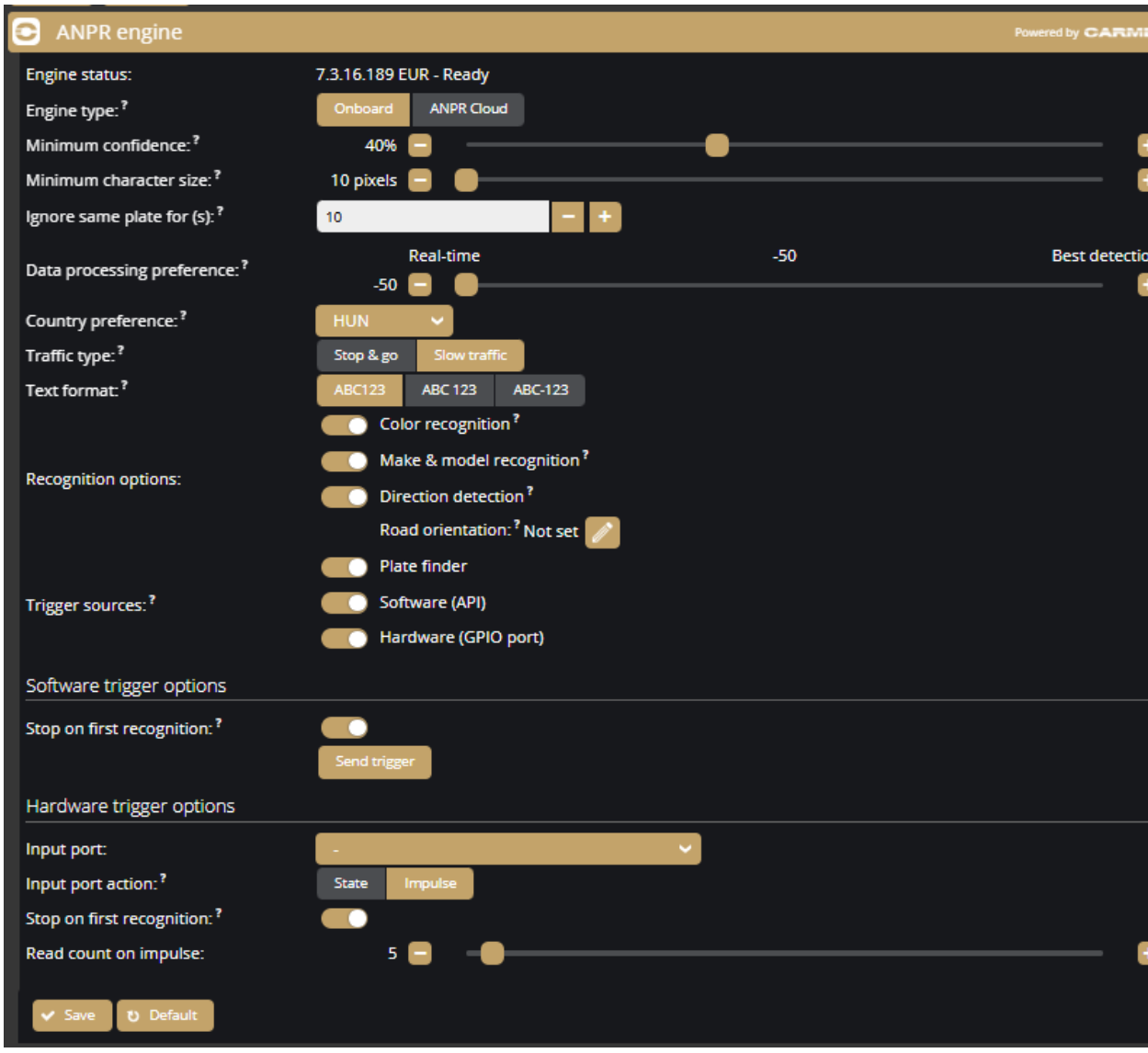


- **ANPR Cloud URL:** Paste the URL you received when subscribing to the field.
- **ANPR Cloud key:** Enter the individual key of the subscription

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

f. }.

g. Draw the area on the picture where you expect to see license plates - with the proper character size, and from the proper angle of view - using a mask. This will increase the possibility of speeding up the license plate recognition, and cars parked/passing by in an undesirable place will not trigger an event. {



In the case of **Onboard ANPR**, you can adjust the primary functions of the ANPR engine:

- **Engine status:** It shows the current ANPR 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

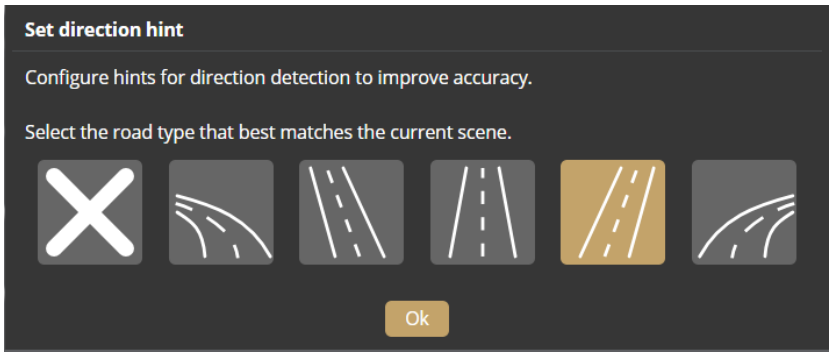
the "ANPR engine status" OSD layer to see the detected license plate numbers and their confidence values and then set the desired minimum confidence level based on the results. See also ***{****Hiba! A hivatkozási forrás nem található.****}***.

- **Minimum character size:** License plates with character size below this threshold are discarded.
- **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.

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

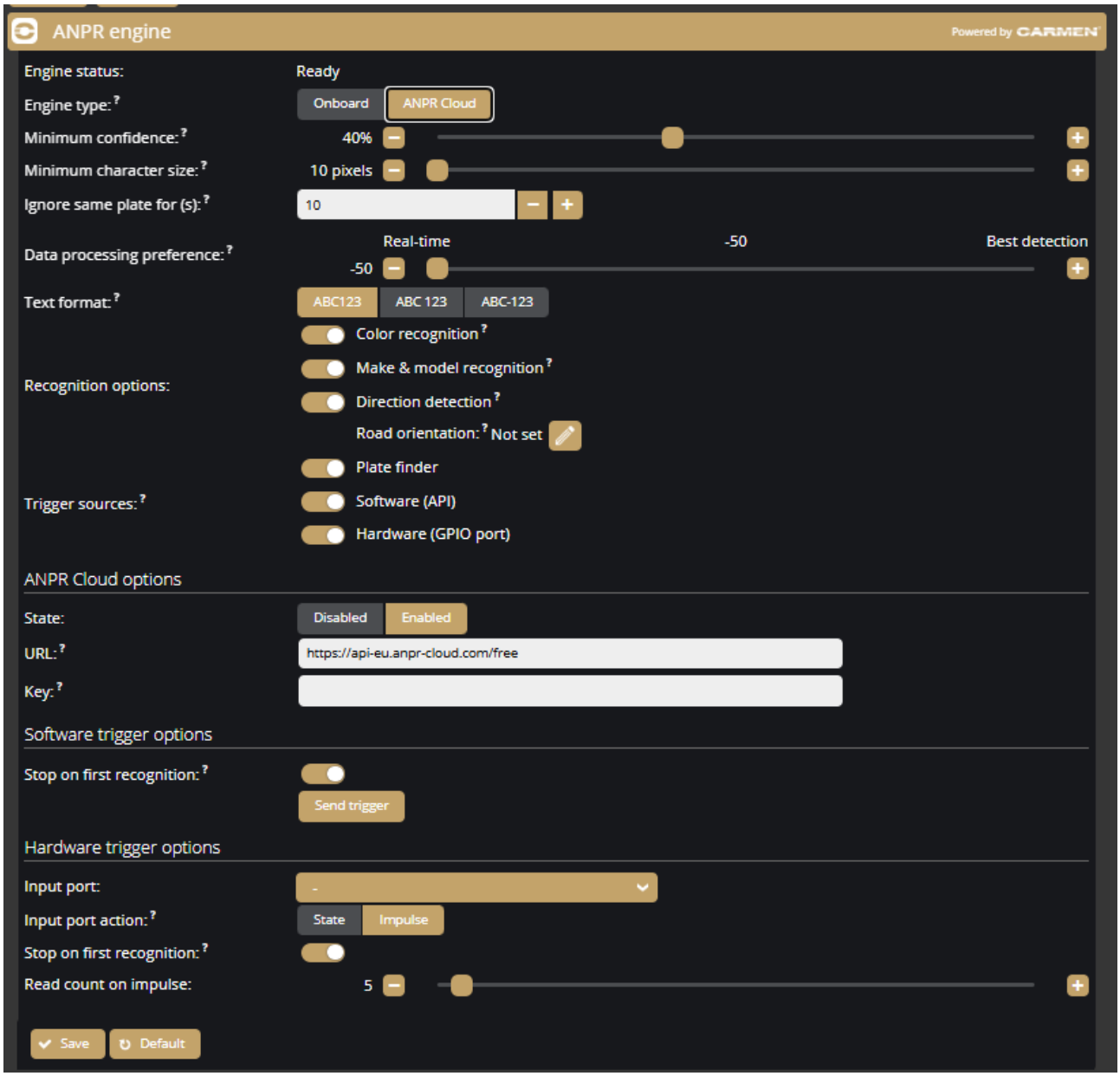
- **Data processing preference:** Configure how the engine processes license plate data.
  Setting the slider towards Real-time will decrease the delay between a vehicle appearing and an event being emitted but also decreases the accuracy of the license plate data. Recommended for quick license plate validation when device is used for controlling vehicle entry points.
  Setting the slider towards Best detection will increase the license plate data precision including vehicle direction and vehicle speed but will dramatically increase the delay between a vehicle appearing and an event being emitted as the engine will spend more time making sure all data are sufficiently validated.\ Recommended when data accuracy is important and events are not used for live triggers but processed later or in bulk.
- **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:** Type of traffic the device processes. Some engines contain more than one recognition possibility. If the uploaded engine supports „Quick" recognition mode, you can select it as „SlowTraffic" mode.
- **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.
- **Text format:** By default license plate characters are returned without spacing. Depending on the selected option symbol groups on a license plate may be separated by whitespaces and/or hypens.
- **Color recognition:** When color recognition is active, the device will attempt to identify the plate and/or vehicle colors. This may slow down the speed of recognition.
- **Make & Model recognition:** When make and model recognition is active, the device will attempt to identify the make, model and color of the vehicle. This may slow down the speed of recognition.

- **Direction detection:** Attempt to detect direction of the vehicle movement by license plate.
  - **Road orientation:** Click on the edit button and set the road type to improve direction detection accuracy.



Important!

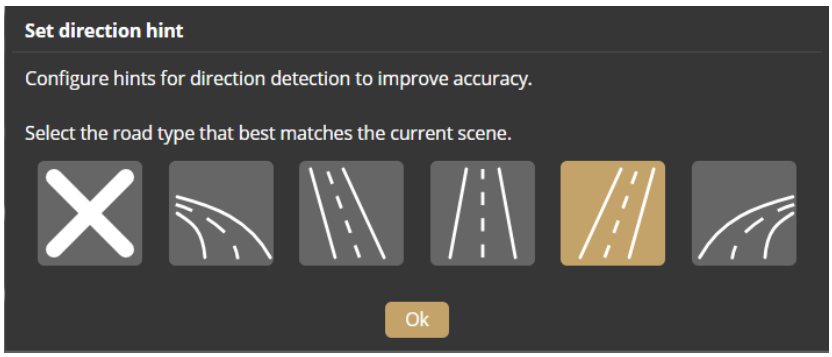
Direction detection is only performed if the same license plate number has been read at least twice by the camera and the height of the license plate number character has changed by at least 2 pixels in the image.


ANPR engine
Powered by CARMEN
Engine status:
Ready
Engine type:
Onboard
ANPR Cloud
Minimum confidence:
40%
Minimum character size:
10 pixels
Ignore same plate for (s):
10
Data processing preference:
Real-time
-50
Best detection
-50
Text format:
ABC123
ABC 123
ABC-123
Recognition options:
Color recognition
Make & model recognition
Direction detection
Road orientation:
Not set
Plate finder
Trigger sources:
Software (API)
Hardware (GPIO port)
ANPR Cloud options
State:
Disabled
Enabled
URL:
https://api-eu.anpr-cloud.com/free
Key:
Software trigger options
Stop on first recognition:
Send trigger
Hardware trigger options
Input port:
Input port action:
State
Impulse
Stop on first recognition:
Read count on impulse:
5
Save
Default

In the case of **Carmen Cloud**, the license plate recognition is not processed in the device. Instead, images with license plates are selected in the device 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/.

- **Engine status:** It shows the current ANPR 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 ***{****Hiba! A hivatkozási forrás nem található.****}***.
- **Minimum character size:** License plates with character size below this threshold are discarded.
- **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.
- **Data processing preference:** Configure how the engine processes license plate data.
  Setting the slider towards Real-time will decrease the delay between a vehicle appearing and an event being emitted but also decreases the accuracy of the license plate data. Recommended for quick license plate validation when device is used for controlling vehicle entry points.
  Setting the slider towards Best detection will increase the license plate data precision including vehicle direction and vehicle speed but will dramatically increase the delay between a vehicle appearing and an event being emitted as the engine will spend more time making sure all data are sufficiently validated.\ Recommended when data accuracy is important and events are not used for live triggers but processed later or in bulk.
- **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:** Type of traffic the device processes. Some engines contain more than one recognition possibility. If the uploaded engine supports „Quick" recognition mode, you can select it as „SlowTraffic" mode.
- **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.
- **Text format:** By default license plate characters are returned without spacing. Depending on the selected option symbol groups on a license plate may be separated by whitespaces and/or hypens.
- **Color recognition:** When color recognition is active, the device will attempt to identify the plate and/or vehicle colors. This may slow down the speed of recognition.
- **Make & Model recognition:** When make and model recognition is active, the device will attempt to identify the make, model and color of the vehicle. This may slow down the speed of recognition.
- **Direction detection:** Attempt to detect direction of the vehicle movement by license plate.
    - **Road orientation:** Click on the edit button and set the road type to improve direction detection accuracy.



- **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.

h. **}**.

i. 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 internal operation of the device, changes of device's parameters, etc. We suggest displaying at least the **General/ANPR engine status** layer during device setup ***{Overlay}***. Also, display these captions during playback and reduce the playback speed to facilitate more accurate evaluation.

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

API DOCUMENTATION

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

10. INTRODUCTION

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

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

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

> **Note:** API functions and properties not covered by this document may be changed or removed in the futurewithout notice

11. 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

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

12. AUTHENTICATION

Accessing resources on the device requires an authenticated session.

13. 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**.

13.1. 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.

13.2. 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

{}
```

13.3. 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

14. EXECUTING COMMANDS

14.1. 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**

14.2. 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

14.3. 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."
        }
}
```

14.4. 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.

14.4.1. BOOLEAN

7

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 aswell.

14.4.2. INTEGERS

8

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.

14.4.3. TIMESTAMPS

9

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
```

14.4.4. DOUBLE

10

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

14.4.5. GUID

11

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

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

14.4.6. ARRAYS OF INTEGERS

12

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.

14.4.7. UNNAMED KEYS

13

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:

```
14  {
5   "91" : "Unnamed entry with arbitrary numeric key",
6   "Test1" : "Named entry which will be the second in the list",
7   "28" : "Another unnamed entry",
8   "4" : "Third unnamed entry",
9   "Test" : "Another named entry which is the last in the list (5th)"
```

14.4.8. LISTS

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

14.4.9. MAP

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

14.5. 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

14.6. 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.

14.6.1. 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

15. DETECTORS & ENGINES

15.1. 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
AlarmDetectorIO *Input port monitor*	**DetectorConfigurationIOEventIO**
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*	**DetectorConfigurationEmergencyLan** **eEventEmergencyLane**
AlarmDetectorForbiddenZone *Forbidden zone violation*	**DetectorConfigurationForbiddenZo** **neEventForbiddenZone**
AlarmDetectorLane *Lane movement*	**DetectorConfigurationLane** **EventLane**
AlarmDetectorRedStop *Traffic light violation*	**DetectorConfigurationRedStop** **EventRedStop**
AlarmDetectorStoppedObject *Prohibited stop detection*	**DetectorConfigurationStoppedObje** **ctEventStoppedObject**
AlarmDetectorStopViolation *Stop sign violation*	**DetectorConfigurationStopViolatio** **nEventStopViolation**
AlarmDetectorTrafficLine *General line crossing*	**DetectorConfigurationTrafficLine** **EventTrafficLine**
AlarmDetectorUTurn *Illegal U-turn detection*	**DetectorConfigurationUTurn** **EventUTurn**
AlarmDetectorWhiteLineViolation *White line violation*	**DetectorConfigurationWhiteLineViolation** **EventWhiteLineViolation**
AlarmDetectorWrongTurn *Illegal turn violation*	**DetectorConfigurationWron** **gTurnEventWrongTurn**
AlarmDetectorWrongWay *Wrong-way driving detection*	**DetectorConfigurationWron** **gWayEventWrongWay**

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

15.2. GEOMETRY

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

15.2.1. 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
```

15.2.2. 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.

16. EVENTS

16.1. 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.

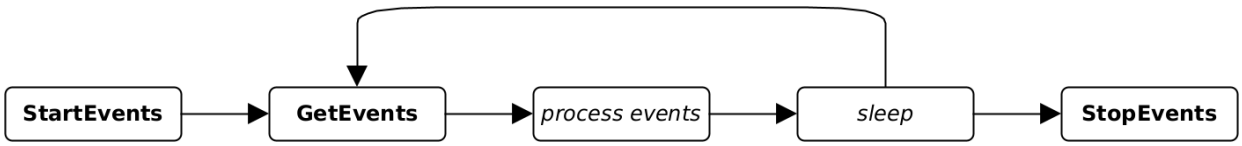
Pros	Cons
Event image and video available Event region of interest image available Compatible with most HTTP server implementation	Requires storage device Significant latency

16.2. 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.



16.3. 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**. 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.

16.3.1. 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

16.3.2. IMAGE ATTACHMENT

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

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

16.3.3. 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
```

16.3.4. 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}
```

16.3.5. 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
```

16.3.6. 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"
}
```

16.4. 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**).

16.4.1. 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.

16.4.2. 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.

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

16.5. 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.

16.5.1. 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.

16.5.2. 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.

16.5.3. 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/1.0Accept: */*
Cache-Control: no-cache
Content-Type: multipart/form-data; boundary=IntellioHttpPostUploaderBoundary
Content-Length: 4330662
Expect: 100-continue

HTTP/1.1 100 Continue

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

1631732906436
--IntellioHttpPostUploaderBoundary
Content-Disposition: form-data; name="event_video_0"
Content-Type: video/mp4

binary data
--IntellioHttpPostUploaderBoundary
Content-Disposition: form-data; name="event_image_0"
Content-Type: image/jpeg

binary data
--IntellioHttpPostUploaderBoundary
Content-Disposition: form-data; name="event_cropped_image_0"
Content-Type: image/jpeg

binary data
--IntellioHttpPostUploaderBoundary
Content-Disposition: form-data; name="event_descriptor"
Content-Type: application/json

{
```


Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
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
```

17. MISCELLANEOUS

17.1. 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)

17.1.1. 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).

17.1.2. 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
```

17.1.3. 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
```

17.1.4. 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
```

18. REFERENCE

18.1. 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

18.1.1. 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**.

18.1.2. ANALYTICS/DELETEALLDETECTORS

Deletes all detector instances except built-in detectors

Specification

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

18.1.3. 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.

18.1.4. 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.

18.1.5. 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

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

18.1.7. 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*

18.1.8. ANALYTICS/GETANPRENGINESTATE

Get the current state of the ANPR engine

Specification

User level	ADMINISTRATOR
Request data	none
Response data	AnprEngineState
Exceptions	none

18.1.9. 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

18.1.10. 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*

18.1.11. 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.

18.1.12. ANALYTICS/GETDETECTORS

Get the active detector instances on this device

Specification

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

18.1.13. 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

18.1.14. 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*

18.1.15. 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*

18.1.16. 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*

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

18.1.17. 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*

18.1.18. 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

18.1.19. 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*

18.1.20. 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*

18.1.21. ANALYTICS/STOPEVENTS

Stop the event buffering for the calling session

Specification

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

18.1.22. 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.

18.2. 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

18.2.1. 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

18.2.2. STORAGE/GETSTATISTICS

Get general statistics from the storage subsystem

Specification

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

18.3. 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.

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

Methods

Method	Description
System/ClearSecurityHistory	Release the block on all clients that are currently banned
System/FactoryReset	Factory reset the settings and reboot
System/GetDevice	Get general information about the device
System/GetSecurityHistory	List the active session and blocked clients
System/GetSecuritySettings	Get the security settings
System/GetVersion	Get the version of the JSON API
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/SetSecuritySettings	Change the security settings
Date & time	
System/GetNtpSettings	Get the NTP settings
System/GetTime	Get the current timestamp
System/SetNtpSettings	Change the NTP settings
System/SetTime	Change the current timestamp
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

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

18.3.1. 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

18.3.2. SYSTEM/CLEARSECURITYHISTORY

Release the block on all clients that are currently banned

Specification

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

18.3.3. 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

18.3.4. 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*

18.3.5. SYSTEM/GETCURRENTUSER

Get the user of the current session

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

Specification

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

18.3.6. SYSTEM/GETDEVICE

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

See also: System/SetDevice

Specification

User level	USER
Request data	none
Response data	SystemSettingsResponse
Exceptions	none

18.3.7. SYSTEM/GETGPIOSETTINGS

Get the available digital inputs and outputs on this device

Specification

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

18.3.8. 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*

18.3.9. SYSTEM/GETNTPSETTINGS

Get the NTP settings

> See also: System/SetNtpSettings

Specification

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

18.3.10. SYSTEM/GETSECURITYHISTORY

List the active session and blocked clients

Specification

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

18.3.11. 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*

18.3.12. SYSTEM/GETTIME

Get the current timestamp

> See also: System/SetTime

Specification

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

18.3.13. SYSTEM/GETUSERS

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

Specification

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

18.3.14. SYSTEM/GETVERSION

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

Specification

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

18.3.15. SYSTEM/MODIFYUSER

Modify the properties of a user account

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

Specification

User level	ADMINISTRATOR
Request data	**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

18.3.16. 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*

18.3.17. SYSTEM/RUNTEST

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

Specification

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

18.3.18. SYSTEM/SETDEVICE

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

> See also: System/GetDevice

Specification

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

18.3.19. SYSTEM/SETGPIOINPUTSETTINGS

Change the configuration of a digital input port

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

Specification

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

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

18.3.20. SYSTEM/SETGPIOOUTPUT

Change the state of a digital output port

See also: System/SetGpioInputSettings, System/SetGpioOutputSettings,

Specification

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

18.3.21. SYSTEM/SETGPIOOUTPUTSETTINGS

Change the configuration of a digital output port

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

Specification

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

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

18.3.23. SYSTEM/SETSECURITYSETTINGS

Change the security settings

See also: System/GetSecuritySettings

Specification

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

18.3.24. SYSTEM/SETTIME

Change the current timestamp

See also: System/GetTime

Specification

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

18.3.25. SYSTEM/TRIGGERGPIOOUTPUT

Send an impulse to a digital output port

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

Specification

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

18.4. STRUCTS

18.4.1. ACTIVESESSION

Active session information

Structure

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

Pseudo code

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

18.4.2. ANALYTICSENGINETRIGGER

Properties of a manual engine trigger.

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

See also: Analytics/TriggerEngine

Structure

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

Pseudo code

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

18.4.3. ANALYTICSENGINETRIGGER

Properties of a manual engine trigger.

> See also: Analytics/TriggerEngine

Structure

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

Pseudo code

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


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

18.4.4. ANPRENGINECONFIGURATION

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

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

Available trigger modes are:

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

The **HardwareTriggerSettings/TriggerMode** option controls how the activation of the input port triggers 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
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 Impulsemode
TriggerMode	string	Activation mode of the trigger
InterruptOnRecognition	bool	(deprecated) When enabled stops further recognition after a successful read. Ignored when InterruptOnRecognition is specifiedin HardwareTriggerSettings and SoftwareTriggerSettings.
MMR	bool	Set to enable MMR recognition on license plates
Masks	List/ Array/ int16	List of polygon coordinates that define the operating area of theengine
RecognitionMode	string	Type of traffic the device processes
SoftwareTriggerSettings		Advanced settings for Software trigger mode
InterruptOnRecognition	bool	When enabled stops further recognition after a successful read
TriggerMode	string	(deprecated) Source of triggers that activates the ANPR engine. Thissetting 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": ...,
            "HardwareTriggerSettings":
            {
                    "InterruptOnRecognition": ...,
                    "Port": "...",
                    "ReadCount": ...,
                    "TriggerMode": "..."
            },
              "InterruptOnRecognition": ...,
              "MMR": ...,
              "Masks":
              {
              "0": [ ..., ..., ... ],
              "1": [ ..., ..., ... ]
              },
              "RecognitionMode": "...",
              "SoftwareTriggerSettings":
              {
                    "InterruptOnRecognition": ...
              },
              "TriggerMode": "...",
              "TriggerModes":
              {
                    "0": "...",
                    "1": "..."
              },
              "Type": "...",
              "ValidInTimeWindow": ...
      }
 }
```

18.4.5. ANPRENGINESTATE

Current state of the ANPR engine

> See also: Analytics/GetAnprEngineState

Structure

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

Pseudo code

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

18.4.6. APIVERSION

JSON API information

> See also: System/GetVersion

Structure

Parameter	Type	Description
Version	int32	Current version of the JSON API

Pseudo code

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

18.4.7. BUFFEREDEVENTS

Query collected events in a sessions buffer.

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

> See also: Analytics/GetEvents

Structure

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

Pseudo code

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

18.4.8. BUFFEREDEVENTSREQUEST

Parameters for starting event buffering on the current session.

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

> See also: Analytics/StartEvents

Structure

Parameter	Type	Description
Filter	List/guid	List of detector IDs

Pseudo code

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


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

18.4.9. DETECTOR

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

> See also: Analytics/GetDetector, Analytics/GetDetectorDefaults, Analytics/SetDetector

Structure

Parameter	Type	Description
Config	**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.

Pseudo code

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


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

18.4.10.DETECTORCLASSREQUEST

Property the uniquely identifies a detector type.

> See also: Analytics/GetDetectorDefaults

Structure

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

Pseudo code

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


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

18.4.11. DETECTORCONFIGURATION

Inherited by: DetectorConfigurationANPR, DetectorConfigurationIO,

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": ...
}
```

18.4.12.DETECTORCONFIGURATIONANPR → DETECTORCONFIGURATION

Configuration of the ANRP detector.

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

Structure

Parameter	Type	Description
Filter	string	New-line separated list of license plates to signal for
Whitelist	bool	Enable filter usage
Inherited from DetectorConfiguration:		
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": ...,
	"Filter": "...",
	"FpsLimit": ...,
	"RestoreDelayMs": ...,
	"Version": ...,
	"ViolationTimeMs": ...,
	"Whitelist": ...
}
```


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

18.4.13. DETECTORCONFIGURATIONEMERGENCYLANE→TRACKINGDETECTORCONFIGURATION

Structure

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

Pseudo code

```
{
	"BuiltIn": ...,
	"Center": ...,
	"Class": "...",
	"Confidence": ...,
	"ConfidenceEnabled": ...,
	"Description": "...",
	"DetectorClassID": ...,
	"DetectorID": "{...}",
	"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

18.4.14.DETECTORCONFIGURATIONFORBIDDENZONE →TRACKINGDETECTORCONFIGURATION

Structure

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

Pseudo code

```
{
	"BuiltIn": ...,
	"Center": ...,
	"Class": "...",
	"Confidence": ...,
	"ConfidenceEnabled": ...,
	"Description": "...",
	"DetectorClassID": ...,
	"DetectorID": "{...}",
	"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

18.4.15.DETECTORCONFIGURATIONIO → DETECTORCONFIGURATION

Configuration of the IO detector.

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

Structure

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

Pseudo code

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


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

18.4.16. DETECTORCONFIGURATIONLANE → TRACKINGDETECTORCONFIGURATION

Structure

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

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

18.4.17. DETECTORCONFIGURATIONREDSTOP → TRACKINGDETECTORCONFIGURATION

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

Structure

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

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

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":
	{
		"0":
		{
			"Id": ...,
			"X0": ...,
			"X1": ...,
			"Y0": ...,
			"Y1": ...
		}, "1":
		{
			"Id": ...,
			"X0": ...,
			"X1": ...,
			"Y0": ...,
			"Y1": ...
		}
	},
	"FpsLimit": ...,
	"GracePeriod": ...,
	"Lines":
	{
		"0":
		{
			"X0": ...,
			"X1": ...,
			"Y0": ...,
			"Y1": ...
		}, "1":
		{
			"X0": ...,
			"X1": ...,
			"Y0": ...,
			"Y1": ...
		}
	},
	"ObjectTypes":
```

```
        {
                "0": "...",
                "1": "..."
        },
        "RestoreDelayMs": ...,
        "TrafficLight":
        {
                "Type": "...",
                "X0": ...,
                "X1": ...,
                "Y0": ...,
                "Y1": ...
        },
        "Version": ...,
        "ViolationTimeMs": ...
}
```

18.4.18. DETECTORCONFIGURATIONSTOPVIOLATION →TRACKINGDETECTORCONFIGURATION

Structure

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

Pseudo code

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


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

18.4.19. DETECTORCONFIGURATIONSTOPPEDOBJECT →TRACKINGDETECTORCONFIGURATION

Structure

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

Pseudo code

```
{
	"BuiltIn": ...,
	"Center": ...,
	"Class": "...",
	"Confidence": ...,
	"ConfidenceEnabled": ...,
	"Description": "...",
	"DetectorClassID": ...,
	"DetectorID": "{...}",
	"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

18.4.20. DETECTORCONFIGURATIONTEST → DETECTORCONFIGURATION

Configure the test detector.

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

When Timeout is larger than zero the detector repeats the cycle of 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
Interval	int64	Duration of normal state in milliseconds
Timeout	int64	Duration of signalling state in milliseconds
Inherited from DetectorConfiguration:		
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": ...,
    "Interval": ...,
    "RestoreDelayMs": ...,
    "Timeout": ...,
    "Version": ...,
    "ViolationTimeMs": ...
}
```


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

18.4.21.DETECTORCONFIGURATIONTRAFFICLINE → TRACKINGDETECTORCONFIGURATION

Structure

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

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": ...,
	"Lines":
	{
		"0":
		{
			"X0": ...,
			"X1": ...,
			"Y0": ...,
			"Y1": ...
		}, "1":
		{
			"X0": ...,
			"X1": ...,
			"Y0": ...,
			"Y1": ...
		}
	},
	"ObjectTypes":
	{
		"0": "...",
		"1": "..."
	},
	"RestoreDelayMs": ...,
	"Version": ...,
	"ViolationTimeMs": ...
}
```


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

18.4.22. DETECTORCONFIGURATIONUTURN → TRACKINGDETECTORCONFIGURATION

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

Structure

Parameter	Type	Description
Direction	string	Direction of crossing that is monitored
Lines	GeometryPolygons	List of segments defining the line that is parallel and inbetween the two straights of the U path (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, ...).
Inherited from TrackingDetectorConfiguration:		
BuiltIn	bool	Automatically created detectors are marked built-in and cannot be deleted
Class	string	Detector type name
Description	string	Description of this detector instance for easier identification
DetectorClassID	int32	Detector type code
DetectorID	guid	Unique ID of the detector instance
DisplayName	string	Name of this detector instance displayed on user-facing interfaces
Enabled	bool	Controls the enabled state of the detector
FpsLimit	double	Limits the run speed of the detector to a specific FPS. Set to zero for no limit.
RestoreDelayMs	int64	unused
Version	int32	Detector type version
ViolationTimeMs	int64	Violations have to be present for this duration before an event is emitted. Not all detectors may use this field.
Center	bool	Set to true to operate using an object's center point instead of all corners

Confidence	int8	Minimum allowed object confidence when **ConfidenceEnabled** is set to true

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

Pseudo code

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

18.4.23. DETECTORCONFIGURATIONWHITELINEVIOLATION →TRACKINGDETECTORCONFIGURATION

Structure

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

Pseudo code

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


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

18.4.24. DETECTORCONFIGURATIONWRONGTURN → TRACKINGDETECTORCONFIGURATION

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

Structure

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

Confidence	int8	Minimum allowed object confidence when **ConfidenceEnabled** is set to true

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

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

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

18.4.25. DETECTORCONFIGURATIONWRONGWAY → TRACKINGDETECTORCONFIGURATION

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

Structure

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

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

Pseudo code

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

18.4.26. DETECTORCREATECONFIGURATION

Initial settings for a new detector instance.

> See also: Analytics/CreateDetector

Structure

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

Pseudo code

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

18.4.27. DETECTORINFO

Collection of properties defining an instance of a detector type.

A built-in detector is a special instance that is created by the device the first time it is booted and it cannot be 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": ...
}
```

18.4.28. 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

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

Pseudo code

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

18.4.29. DETECTORREQUEST

Collection of properties that uniquely identifies a detector instance.

> See also: Analytics/DeleteDetector, Analytics/DisableDetector, Analytics/EnableDetector, Analytics/ GetDetector, Analytics/GetDetectorState

Structure

Parameter	Type	Description
DetectorID	guid	Unique ID of the detector instance

Pseudo code

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

18.4.30. DETECTORSTATE

The detector state value

Numeric value	String value	Description
0	dsNotConfigured	Detector is not configured or the current configuration is invalid
1	dsInit	Detector is currently initializing the state machine and 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": ...
}
```

18.4.31. DETECTORTYPEINFO

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

Structure

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

Pseudo code

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


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

18.5. 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.

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

Structure

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

Pseudo code

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


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

18.5.1. EVENTANPR → EVENT

License plate detection event.

Structure

Parameter	Type	Description
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 **Approaching**, **Moving away** or **Unknown**.
MMR		Make and model recognition results
Category	string	Vehicle category
CategoryConfidence	double	Confidence of the category recognition
Color	string	Color of vehicle in #RRGGBB format
ColorConfidence	double	Confidence of the color recognition
Make	string	Make of the vehicle
MakeAndModelConfidence	double	Confidence of the make and model recognitions
Model	string	Model of the vehicle
Text	string	License plate text
TextColor	string	Text color of the license plate in #RRGGBB format
TriggerSource		Properties of the trigger that started the license plate recognition
Name	string	Unique name of the trigger

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

Pseudo code

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

18.5.2. 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 **Approaching**, **Moving** away or **Unknown**.
MMR		Make and model recognition results
Category	string	Vehicle category
CategoryConfidence	double	Confidence of the category recognition
Color	string	Color of vehicle in #RRGGBB format
ColorConfidence	double	Confidence of the color recognition
Make	string	Make of the vehicle
MakeAndModelConfidence	double	Confidence of the make and model recognitions
Model	string	Model of the vehicle
Text	string	License plate text
TextColor	string	Text color of the license plate in #RRGGBB format
TriggerSource		Properties of the trigger that started the license plate recognition
Name	string	Unique name of the trigger
Source	string	Type of the trigger (see **TriggerModes** at **AnprEngineConfiguration**)
Timestamp	int64	Timestamp of when the trigger was activated

Pseudo code

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


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

18.5.3. EVENTEMERGENCYLANE → EVENT

Structure

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

Pseudo code

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

18.5.4. EVENTFORBIDDENZONE → EVENT

Structure

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

Pseudo code

```
{
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "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

18.5.5. EVENTIO → EVENT

Input port activation event

Structure

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

Pseudo code

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


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

18.5.6. EVENTLANE → EVENT

Structure

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

Pseudo code

```
{
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "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

18.5.7. EVENTREDSTOP → EVENT

Structure

Parameter	Type	Description
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
StartTime	int64	Wall clock timestamp in milliseconds of the moment the object first appeared
State	string	State of object when the event was created
Type	string	Type of object
OrangeTimestamp	int64	Wall clock timestamp in milliseconds when the light entered orange state
RedTimestamp	int64	Wall clock timestamp in milliseconds when the light entered red state
Inherited from Event:		
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the event creation
EventTriggerTime	int64	Wall clock timestamp in milliseconds when the conditions for the event were met
State	string	State of the detector after the event was emitted (see DetectorState)

Pseudo code

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

18.5.8. EVENTSTOPVIOLATION → EVENT

Structure

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

Pseudo code

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

18.5.9. EVENTSTOPPEDOBJECT → EVENT

Structure

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

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

Pseudo code

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

18.5.10.EVENTTEST → EVENT

Basic test event

Structure

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

Pseudo code

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

18.5.11. EVENTTRAFFICLINE → EVENT

Structure

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

Pseudo code

```
{
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "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

18.5.12.EVENTUTURN → EVENT

Structure

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

Pseudo code

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

18.5.13. EVENTWHITELINEVIOLATION → EVENT

Structure

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

Pseudo code

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

18.5.14.EVENTWRONGTURN → EVENT

Structure

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

Pseudo code

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

18.5.15.EVENTWRONGWAY → EVENT

Structure

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

Pseudo code

```
{
    "DetectorClassID": ...,
    "DetectorEventType": "...",
    "DetectorID": "{...}",
    "DetectorVersion": ...,
    "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

18.5.16. GPSSETTINGS

Structure

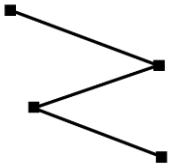
Parameter	Type	Description
Latitude	double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees

Pseudo code

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

18.5.17. 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
SequenceNumber	int32	Numeric id of this group for ordering

Pseudo code

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

18.5.18. 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/**GeometryLineGroup**	List of line group
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

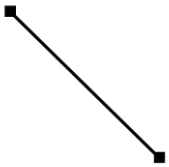
Adaptive Recognition **America**
Adaptive Recognition **Hungary**
Adaptive Recognition **Nordic**
Adaptive Recognition **Singapore**
REQUESTINFO@ADAPTIVERECOGNITION.COM
WWW.ADAPTIVERECOGNITION.COM

Pseudo code

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

18.5.19. 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

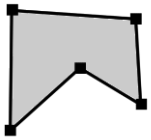
Pseudo code

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


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

18.5.20. GEOMETRYPOLYGONS

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



Structure

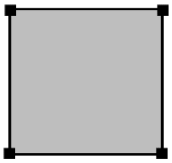
Parameter	Type	Description
Masks	List/Array/int16	List of masks. Each mask is a list of coordinates where odd and even indicies arex and y coordinates of a corner in the polygon (x0, y0, x1, y1, ...).

Pseudo code

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

18.5.21.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

18.5.22. GPIOINPUTPORT → GPIOPORT

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": "..."
}
```

18.5.23. GPIOOUTPUTPORT → GPIOPORT

Settings of a digital output port

> See also: System/SetGpioOutputSettings

Structure

Parameter	Type	Description
ActiveTime	int32	Duration of the active state after the output is triggered
DetectorList	List/ guid	List of detector IDs that can automatically trigger this output with an event
OutputMode	string	Output signal form. Only the "Impulse" mode is supported.
Inherited from 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": ...,
	"ActiveTime": ...,
	"DetectorList":
	{
		"0": "{...}",
		"1": "{...}"
	},
	"OutputMode": "...",
	"Port": "..."
}
```


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

18.5.24. GPIOOUTPUTPORTSTATE → GPIOPORTID

Settings for changing the state of a digital output port

> See also: System/SetGpioOutput

Structure

Parameter	Type	Description
Active	bool	New state of the digital output port
Inherited from GpioPortId:		
Port	string	Unique identifier of a digital input/output port

Pseudo code

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

18.5.25. GPIOPORT → GPIOPORTID

Settings of a digital input/output port

> Inherited by: GpioInputPort, GpioOutputPort
>
> See also: System/SetGpioInputSettings, System/SetGpioOutputSettings

Structure

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

Pseudo code

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

18.5.26. 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": "..."
}
```

18.5.27. GPIOPORTSTATE → GPIOPORTID

State of a digital port

> Inherited by: GpioPortStateChange

Structure

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

Pseudo code

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

18.5.28. GPIOPORTSTATECHANGE → GPIOPORTSTATE

Structure

Parameter	Type	Description
Type	string	Value of "Input" or "Output" indicating the port type
Inherited from GpioPortState:		
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

18.5.29. 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.

Pseudo code

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

18.5.30. GPIOSTATES

Last known state of all digital input/output ports

> See also: System/GetGpioStates

Structure

Parameter	Type	Description
Inputs	Map/**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	Map/**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":
    {
        "named_key0":
        {
            "Active": ...,
            "Port": "...",
            "Timestamp": ...
        },
        "named_key1":
        {
            "Active": ...,
            "Port": "...",
            "Timestamp": ...
        }
    },
    "Outputs":
    {
        "named_key0":
        {
            "Active": ...,
            "Port": "...",
            "Timestamp": ...
        },
        "named_key1":
        {
            "Active": ...,
            "Port": "...",
            "Timestamp": ...
        }
    }
}
```


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

18.5.31. INDEXEDTRACKINGDETECTORLINES → GEOMETRYLINESEGMENT

Structure

Parameter	Type	Description
Id	int8	Index of the line
Inherited from GeometryLineSegment:		
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

18.5.32. LOCATIONSETTINGS

Structure

Parameter	Type	Description
GPS	GPSSettings	Location as GPS coordinates
Latitude	double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees

Pseudo code

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

18.5.33. MODULEANALYTICS → SYSTEMSETTINGSMODULE

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

Tracker	Supports the iTracking tracker engine (see **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": "..."
}
```

18.5.34. MODULEIO → SYSTEMSETTINGSMODULE

Capabilities of the IO module

Structure

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

Pseudo code

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


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

18.5.35. 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": ...
}
```

18.5.36. 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

18.5.37. 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

18.5.38. 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": "..."
	}
}
```


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

18.5.39. 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": "..."
}
```

18.5.40. REDSTOPVIOLATIONINFO → TRACKEDOBJECTINFO

Structure

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

Pseudo code

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

18.5.41. 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":
    {
        "0":
        {
            "LastSeen": ...,
            "Source": "...",
            "User": "..."
        }, "1":
        {
            "LastSeen": ...,
            "Source": "...",
            "User": "..."
        }
    }
}
```

18.5.42. 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

18.5.43. STORAGEEVENTS → STORAGEEVENTSREQUEST

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
EventList	List/Event	List of events that match the search criteria
DetectorClassID	int32	Type ID of the detector
DetectorEventType	string	Type of this event
DetectorID	guid	Unique ID of the detector
DetectorVersion	int32	Version of the detector
EventCode	int32	Detector specific event code
EventID	guid	Unique ID of the event
EventTime	int64	Wall clock timestamp in milliseconds of the detectedevent
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)
Status	string	Final status of the query
Inherited from StorageEventsRequest:		
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 onlyexact 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 tomatch any characters. A question mark (?) indicatesone 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 begginingof the search range

Pseudo code

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

18.5.44. 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": ...
}
```

18.5.45. 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": "..."
}
```

18.5.46. 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": ...
}
```

18.5.47. 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":
    {
        "0":
        {
            "DetectorClass": "...",
            "InstanceCount": ...,
            "InstanceLimit": ...,
            "Version": ...
        }, "1":
        {
            "DetectorClass": "...",
            "InstanceCount": ...,
            "InstanceLimit": ...,
            "Version": ...
        }
    }
}
```

18.5.48. SYSTEMSETTINGS

Inherited by: SystemSettingsResponse

See also: System/GetDevice, System/SetDevice

Structure

Parameter	Type	Description
Description	string	User-specified description
Location	**LocationSettings**	User-specified location
GPS	**GPSSettings**	Location as GPS coordinates
Latitude	double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees
Name	string	User-specified name

Pseudo code

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

18.5.49. 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": "..."
}
```

18.5.50. 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

18.5.51.SYSTEMSETTINGSRESPONSE → SYSTEMSETTINGS

See also: System/GetDevice

Structure

Parameter	Type	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	Map/SystemSettingsModule	List of module specific entries that describeeach module's capabilities
Uptime	int64	Elapsed milliseconds since the system started
Inherited from SystemSettings:		
Description	string	User-specified description
Location	LocationSettings	User-specified location
GPS	GPSSettings	Location as GPS coordinates
Latitude	double	Latitude coordinate in decimal degrees
Longitude	double	Longitude coordinate in decimal degrees
Name	string	User-specified name

Pseudo code

```
{
    "Description": "...",
    "Device":
    {
        "Description": "...",
        "FirmwareVersion": "...",
        "ProductClass": "...",
        "ProductDisplayName": "...",
        "ProductName": "...",
        "ProductSubclass": "...",
        "RequiredFirmwareVersion": "...",
        "Serial": "..."
    },
    "InstanceId": ...,
    "Location":
    {
        "GPS":
        {
            "Latitude": ...,
            "Longitude": ...
        }
    },
    "Modules":
    {
        "named_key0":
        {

        },
        "named_key1":
        {

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


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

18.5.52. 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": ...
}
```

18.5.53. 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": "..."
}
```

18.5.54. TIMESETTINGS

Device time settings

> See also: System/GetTime, System/SetTime

Structure

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

Pseudo code

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

18.5.55. 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": "..."
}
```

18.5.56. TRACKERCONFIGURATION

Configuration of the iTracking engine.

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

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

Structure

Parameter	Type	Description
Config		
Masks	List/Array/int16	Mask defining the working area of the tracker (see **GeometryPolygons)**

Pseudo code

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

18.5.57. TRACKINGDETECTORCONFIGURATION → DETECTORCONFIGURATION

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

Structure

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

Pseudo code

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


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

18.5.58. USER → USERINFO

All user account information

> See also: System/AddUser, System/ModifyUser

Structure

Parameter	Type	Description
Password	string	User password (write only)
Inherited from UserInfo:		
Name	string	User name
Role	string	User role

Pseudo code

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


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

18.5.59. 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": "..."
}
```

18.5.60. USERINFO → USERID

User account information

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

Structure

Parameter	Type	Description
Role	string	User role
Inherited from UserId:		
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

18.5.61. USERS

Contains information about all user accounts available on the device

> See also: System/GetUsers

Structure

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

Pseudo code

```
{
    "Users":
    {
        "0":
        {
            "Name": "...",
            "Password": "...",
            "Role": "..."
        }, "1":
        {
            "Name": "...",
            "Password": "...",
            "Role": "..."
        }
    }
}
```


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.