MMR+ANPR REST Server 4.7.0

User Guide

Copyright © 2026, Eyedea Recognition s.r.o.

All rights reserved

Eyedea Recognition s.r.o. is not responsible for any damages or losses caused by incorrect or inaccurate results or unauthorized use of the software MMR+ANPR REST Server.

Thales and the Thales logo are trademarks and service marks of Thales Group and/or its subsidiaries and are registered in certain countries. Sentinel, Sentinel Admin Control Center and Sentinel Hardware Key are registered trademarks of Thales Group.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

Intel is a trademark of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

NVIDIA, the NVIDIA logo, GeForce®, GeForce® GTX, CUDA®, the CUDA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and/or other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Docker and the Docker logo are trademarks or registered trademarks of Docker, Inc. in the United States and/or other countries. Docker, Inc. and other parties may also have trademark rights in other terms used herein.

Contact:

Address:
Eyedea Recognition, s.r.o.
Vyšehradská 320/49
128 00, Prague 2
Czech Republic

web: http://www.eyedea.ai

email: info@eyedea.ai

Table of Contents

1 Introduction 1-4
2 Hardware requirements 2-5
2.1 Minimal requirements 2-5
2.2 Recommended requirements 2-5
3 Version history 3-6
4 Docker setup 4-14
4.1 GPU Support Prerequisites 4-14
4.2 Sentinel license protection system 4-14
4.3 Building the Docker image 4-15
4.4 Running the Docker container 4-17
4.5 Stopping the Docker container 4-20
4.6 Log files 4-21
5 REST API Documentation 5-23
5.1 SDK Engine Overview 5-23
5.2 Response Status Codes 5-24
5.3 Entry point 5-24
5.4 System Info 5-25
5.5 Recognition 5-31

1 Introduction

Eyedea Recognition's MMR+ANPR REST Server is a Java server application with REST interface running on Tomcat Docker container which allows to detect vehicles (or generally "road users") in input images and recognize the type and text of detected plates, as well as the view, category, make, model, generation, variation, color and tags (various traits) of the vehicle.

MMR+ANPR REST Server uses our state-of-the-art libraries, LPM and MMR SDK, with a possibility to easily switch to the latest modules for a different region. Both server REST interface and the simple web application built on it provide the detection and recognition used in various use cases, as well as server monitoring.

2 Hardware requirements

2.1 Minimal requirements

- Processor: Intel® Core™ i5 or equivalent, 4 cores / 8 threads or higher
- RAM: 8 GB
- Hard disk: 256 GB (optional SSD)
- GPU (optional): NVIDIA Driver version >= 410.48 compatible
- Operating system: Ubuntu 20.04 and higher – x86_64 platform

2.2 Recommended requirements

- Processor: Intel® Core™ i7 or equivalent, 8 cores / 16 threads or higher
- RAM: 16 GB
- Hard disk: 512 GB, SSD
- GPU: NVIDIA® GeForce RTX™ 30-series or equivalent
- Operating system: Ubuntu 20.04 and higher – x86_64 platform

3 Version history

Version 4.7.0

Released: 2026/07/03

- Added configurable pixel admission gateway for image decoding with controlled HTTP 503 backpressure when capacity is exhausted
- Updated LPM and MMR modules (precise MMR modules replaced with accelerated ones)
- Used LPM SDK: LPM-v7.9.1-2026-04-08-Ubuntu-20.04-x86_64-hasp10.2
- Updated MMR SDK: Eyedea-MMR-2.27.0-Ubuntu-20.04-x86_64-OpenCL-HASP
- Used Sentinel license protection system: aksusbd_108842-10.31.1

Version 4.6.0

Released: 2026/06/12

- Processing threads setting split into processing instances and internal CPU threads per instance
- Added optional computational constraints to recognition request
- Used LPM SDK: LPM-v7.9.1-2026-04-08-Ubuntu-20.04-x86_64-hasp10.2
- Used MMR SDK: Eyedea-MMR-2.26.2-Ubuntu-20.04-x86_64-OpenCL-HASP
- Updated Sentinel license protection system: aksusbd_108842-10.31.1

Version 4.5.3

Released: 2026/05/21

- Added characterConfidences to plate recognition response
- Updated LPM modules
- Updated LPM SDK: LPM-v7.9.1-2026-04-08-Ubuntu-20.04-x86_64-hasp10.2
- Used MMR SDK: Eyedea-MMR-2.26.2-Ubuntu-20.04-x86_64-OpenCL-HASP
- Used Sentinel license protection system: aksusbd_108842-10.21.1

Version 4.5.2

Released: 2026/04/08

- Updated LPM modules
- Used LPM SDK: LPM-v7.9.0-2025-12-17-Ubuntu-18.04-x86_64-hasp10.2
- Used MMR SDK: Eyedea-MMR-2.26.2-Ubuntu-20.04-x86_64-OpenCL-HASP
- Used Sentinel license protection system: aksusbd_108842-10.21.1

Version 4.5.1

Released: 2026/01/12

- Updated LPM and MMR modules
- Clarified documentation on USB dongle vs. cloud license usage
- Updated LPM SDK: LPM-v7.9.0-2025-12-17-Ubuntu-18.04-x86_64-hasp10.2
- Updated MMR SDK: Eyedea-MMR-2.26.2-Ubuntu-20.04-x86_64-OpenCL-HASP
- Updated Sentinel license protection system: aksusbd_108842-10.21.1

Version 4.5.0

Released: 2025/09/11

- Fixed license and resource release on Docker container shutdown
- Improved Docker container startup
- Used LPM SDK: LPM-v7.8.0-2025-02-04-Ubuntu-18.04-hasp10.0
- Used MMR SDK: Eyedea-MMR-2.25.0-Ubuntu-20.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-10.14.1

Version 4.4.5

Released: 2025/09/04

- Updated LPM modules
- Used LPM SDK: LPM-v7.8.0-2025-02-04-Ubuntu-18.04-hasp10.0
- Used MMR SDK: Eyedea-MMR-2.25.0-Ubuntu-20.04-x86_64-HASP
- Updated Sentinel license protection system: aksusbd_108842-10.14.1

Version 4.4.4

Released: 2025/08/13

- Updated LPM modules
- Used LPM SDK: LPM-v7.8.0-2025-02-04-Ubuntu-18.04-hasp10.0
- Used MMR SDK: Eyedea-MMR-2.25.0-Ubuntu-20.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-10.11.1

Version 4.4.3

Released: 2025/06/18

- Updated LPM and MMR modules
- Used LPM SDK: LPM-v7.8.0-2025-02-04-Ubuntu-18.04-hasp10.0
- Updated MMR SDK: Eyedea-MMR-2.25.0-Ubuntu-20.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-10.11.1

Version 4.4.2

Released: 2025/04/11

- Updated LPM modules
- Improved support for JPEG images with non-standard EXIF metadata
- Used LPM SDK: LPM-v7.8.0-2025-02-04-Ubuntu-18.04-hasp10.0
- Used MMR SDK: Eyedea-MMR-2.24.0-Ubuntu-20.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-10.11.1

Version 4.4.1

Released: 2025/03/28

- Removed isSingleLine plate attribute from recognition request and response
- Used LPM SDK: LPM-v7.8.0-2025-02-04-Ubuntu-18.04-hasp10.0
- Used MMR SDK: Eyedea-MMR-2.24.0-Ubuntu-20.04-x86_64-HASP

- Used Sentinel license protection system: aksusbd_108842-10.11.1

Version 4.4.0

Released: 2025/03/24

- Supported segmentation (LPM module 802, if explicitly enabled)
- Used LPM SDK: LPM-v7.8.0-2025-02-04-Ubuntu-18.04-hasp10.0
- Used MMR SDK: Eyedea-MMR-2.24.0-Ubuntu-20.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-10.11.1

Version 4.3.1

Released: 2025/03/17

- Updated LPM modules
- Used LPM SDK: LPM-v7.8.0-2025-02-04-Ubuntu-18.04-hasp10.0
- Used MMR SDK: Eyedea-MMR-2.24.0-Ubuntu-20.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-10.11.1

Version 4.3.0

Released: 2025/03/07

- Supported wheel detection (LPM module 802)
- Supported grouping of road users into combinations (LPM module 802)
- Refined description of recognition error messages
- Updated LPM SDK: LPM-v7.8.0-2025-02-04-Ubuntu-18.04-hasp10.0
- Used MMR SDK: Eyedea-MMR-2.24.0-Ubuntu-20.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-10.11.1

Version 4.2.0

Released: 2024/12/12

- Updated LPM and MMR modules, added view8 to MMR results
- Used LPM SDK: LPM-v7.7.0-2024-11-05-Ubuntu-18.04-hasp9.0
- Updated MMR SDK: Eyedea-MMR-2.24.0-Ubuntu-20.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-10.11.1

Version 4.1.5

Released: 2024/11/08

- Updated LPM modules
- Updated LPM SDK: LPM-v7.7.0-2024-11-05-Ubuntu-18.04-hasp9.0
- Used MMR SDK: Eyedea-MMR-2.23.0-Ubuntu-20.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-10.11.1

Version 4.1.4

Released: 2024/10/01

- Added unreadable plate attribute to recognition response
- Used LPM SDK: LPM-v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
- Used MMR SDK: Eyedea-MMR-2.23.0-Ubuntu-20.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-10.11.1

Version 4.1.3

Released: 2024/09/23

- Added support for AVIF image files
- Used LPM SDK: LPM-v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
- Used MMR SDK: Eyedea-MMR-2.23.0-Ubuntu-20.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-10.11.1

Version 4.1.2

Released: 2024/09/19

- Improved some Bad Request error messages
- Used LPM SDK: LPM-v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
- Used MMR SDK: Eyedea-MMR-2.23.0-Ubuntu-20.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-10.11.1

Version 4.1.1

Released: 2024/09/04

- Updated LPM modules
- Used LPM SDK: LPM-v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
- Used MMR SDK: Eyedea-MMR-2.23.0-Ubuntu-20.04-x86_64-HASP
- Updated Sentinel license protection system: aksusbd_108842-10.11.1

Version 4.1.0

Released: 2024/08/01

- Added possibility to start server with several MMR Box/Plate engines running
- Used LPM SDK: LPM-v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
- Used MMR SDK: Eyedea-MMR-2.23.0-Ubuntu-20.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-9.12.1

Version 4.0.1

Released: 2024/06/21

- Updated LPM and MMR modules
- Used LPM SDK: LPM-v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
- Used MMR SDK: Eyedea-MMR-2.23.0-Ubuntu-20.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-9.12.1

Version 4.0.0

Released: 2024/04/24

- Unified interface for road users detection and recognition
- Reorganized recognition request and response
- Used combined detector (detecting license plates, boxes, windshields)
- Added possibility to start server with several detection and OCR engines running
- Used LPM SDK: LPM-v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
- Used MMR SDK: Eyedea-MMR-2.22.0-Ubuntu-18.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-9.12.1

Version 3.0.0

Released: 2023/06/29

- Supported vehicle detection + recognition
- Changed POST endpoint for license plate recognition: recognition → lpRecognition
- Renamed optional license plate recognition data parameters: boundingBox → roi, lpDetection → lpDetections
- Changed default license plate LPM module to 801 (general)
- Moved LPM module and MMR models settings from Dockerfile to env-*.list files
- Used LPM module: LPM-v7.4.1-2023-01-12-Ubuntu-18.04-hasp
- Used MMR module: Eyedea-MMR-2.21.0-Ubuntu-18.04-x86_64-HASP
- Updated Sentinel license protection system: aksusbd_108842-9.12.1

Version 2.3.2

Released: 2023/05/23

- Added option to log errors, warnings and recognition statistics to files
- Added server start time to system info response
- Used LPM module: LPM-v7.4.1-2023-01-12-Ubuntu-18.04-hasp
- Used MMR module: Eyedea-MMR-2.20.0-Ubuntu-18.04-x86_64-HASP
- Updated Sentinel license protection system: aksusbd_108842-8.53.1

Version 2.3.1

Released: 2023/04/14

- Supported new LPM features: added occlusion, truncated and cluster detection attributes
- Modified layout of Recognition page
- Used LPM module: LPM-v7.4.1-2023-01-12-Ubuntu-18.04-hasp
- Used MMR module: Eyedea-MMR-2.20.0-Ubuntu-18.04-x86_64-HASP
- Updated Sentinel license protection system: aksusbd-8.51.1

Version 2.3.0

Released: 2023/01/05

- Supported new MMR features: added tags to response, all-in-one MMR engine
- Computing all MMR attributes (view, category, make, model, generation, variation, color and tags) by default
- Renamed MMR related Dockerfile and env.list variables
- Used LPM module: LPM-v7.4.0-2022-08-30-Ubuntu-18.04-hasp
- Used MMR module: Eyedea-MMR-2.20.0-Ubuntu-18.04-x86_64-HASP

Version 2.2.0

Released: 2022/09/27

- New product branding as "MMR+ANPR REST Server"
- Changed entry point to *SERVER_IP*:8080/RESTServer
- Used LPM module: LPM-v7.4.0-2022-08-30-Ubuntu-18.04-hasp
- Used MMR module: Eyedea-MMR-2.12.0-Ubuntu-18.04-x86_64-HASP

Version 2.1.1

Released: 2022/08/18

- Used LPM module: LPM-v7.3.1-2022-08-16-Ubuntu-18.04-hasp
- Used MMR module: Eyedea-MMR-2.12.0-Ubuntu-18.04-x86_64-HASP

Version 2.1.0

Released: 2022/07/28

- Fixed returning error message on SDK (usually license) error.
- Renamed Dockerfile and env.list variables.
- Limited number of LPM Detector CPU threads to 1 (optimized internally).
- Extended documentation.
- Updated to Java 12. Built with OpenJDK 12.0.1.
- Used LPM module: LPM-v7.3.0-2022-06-22-Ubuntu-18.04-hasp
- Used MMR module: Eyedea-MMR-2.12.0-Ubuntu-18.04-x86_64-HASP
- Updated Sentinel license protection system: aksusbd-8.41.1

Version 2.0.4

Released: 2022/04/11

- Fixed LPM memory leak.
- Used LPM module: LPM-v7.2-2021-10-08-Ubuntu-16.04-hasp
- Used MMR module: Eyedea-MMR-2.11.0-Ubuntu-18.04-x86_64-HASP

Version 2.0.3

Released: 2022/02/24

- Added support for GPU detection (currently available for LPM module 800).
- Reduced Sentinel license requirements.
- Fixed initialization of engines running on the CPU.
- LPM module selection moved to Dockerfile.
- Used LPM module: LPM-v7.2-2021-10-08-Ubuntu-16.04-hasp
- Used MMR module: Eyedea-MMR-2.11.0-Ubuntu-18.04-x86_64-HASP

Version 2.0.2

Released: 2021/12/20

- Split MMR-VCMMGV and MMR-Color in settings, renamed Dockerfile and env.list variables.
- Used LPM module: LPM-v7.2-2021-10-08-Ubuntu-16.04-hasp

- Used MMR module: Eyedea-MMR-2.11.0-Ubuntu-18.04-x86_64-HASP
- Updated Sentinel license protection system: aksusbd-8.31.1

Version 2.0.1

Released: 2021/07/16

- Added generation and variation attributes to MMR result.
- Used LPM module: LPM-v7.1-2020-04-16-Ubuntu-16.04-hasp
- Used MMR module: Eyedea-MMR-2.10.0-Ubuntu-18.04-x86_64-HASP
- Updated Sentinel license protection system: aksusbd-8.21.1

Version 2.0.0

Released: 2021/02/02

- Added more detailed SDK information to serverSystemInfo response.
- Removed null / NaN fields from response.
- Used LPM module: LPM-v7.1-2020-04-16-Ubuntu-16.04-hasp
- Used MMR module: Eyedentify-VCL-2.9.0-Ubuntu-16.04-x86_64-HASP

Version 2.0.0-BETA

Released: 2020/12/11

- Used LPM SDK for detection and OCR.
- Added option to specify a bounding box to reduce the input image area scanned by the detector.
- Simplified response for OCR / MMR module disabled (returning null).
- Removed countryID from anprResult response element.
- Used LPM module: LPM-v7.1-2020-04-16-Ubuntu-16.04-hasp
- Used MMR module: Eyedentify-VCL-2.9.0-Ubuntu-16.04-x86_64-HASP

Version 1.2.1

Released: 2020/04/14

- Used EyeScan module: EyeScanSDK-v3.8.3-DummyPack-Ubuntu-16.04-x86_64-HASP
- Used ANPR module: Eyedentify-ANPR-2.7.0-Ubuntu-16.04-x86_64-hasp
- Used MMR module: Eyedentify-VCL-2.7.2-Ubuntu-16.04-x86_64-hasp

Version 1.2.0

Released: 2019/10/08

- Added support for bypassing the internal detector by specifying license plates positions in JSON format.
- HTML documentation updated.
- Used EyeScan module: EyeScanSDK-v3.7.0-DummyPack-Ubuntu-16.04-x86_64-HASP
- Used ANPR module: Eyedentify-ANPR-2.5.1-Ubuntu-16.04-x86_64-hasp
- Used MMR module: Eyedentify-VCL-2.6.0-Ubuntu-16.04-x86_64-hasp

Version 1.1.1

Released: 2019/07/01

- Added option to run the server with ANPR / MMR modules disabled.
- HTML documentation updated.

Version 1.1.0

Released: 2019/04/17

- Web interface added: System Info page and Recognition page.
- Updated to the Java 11. Built with OpenJDK 11.0.2.
- HTML documentation added.
- Used EyeScan module: EyeScanSDK-v3.4.1-DummyPack-Ubuntu-16.04-x86_64-HASP
- Used ANPR module: Eyedentify-ANPR-2.5.1-Ubuntu-16.04-x86_64-hasp
- Used MMR module: Eyedentify-VCL-2.5.1-Ubuntu-16.04-x86_64-hasp

Version 1.1.0-BETA

Released: 2019/01/11

- Added support for descriptor batch computation.
- Added support for GPU computation (ANPR, MMR).

Version 1.0.0

Released: 2018/06/13

- First release version.
- MMR and ANPR recognition REST API server application with basic functionality for still photos.
- Docker image creation scripts included.

4 Docker setup

4.1 GPU Support Prerequisites

If you do not intend to use the GPU, you can skip this chapter.

1. Install GPU

Install GPU supporting NVIDIA CUDA® platform into your Docker host system. The Docker host is the system running the Docker daemon.

2. Linux x86_64

Using the GPU for computation in the Docker environment requires the Docker to be installed and running on the Linux x86_64 system. Use the distribution which is supported by both the Docker and the NVIDIA Driver.

3. Install NVIDIA Driver

Install NVIDIA Driver into the Docker host system. See the Version history for the minimal NVIDIA Driver version required by the CUDA® used in the current build. Write down the version of the NVIDIA Driver you are installing, the same driver version must be installed in the Docker image (see the chapter Important: When starting the container, you will need to configure the HASP_REMOTE_SERVERADDR application variable, which specifies the location of the license server. Details are provided in the Running the Docker container chapter.

Building the Docker image).

4. Install nvidia-container-runtime library

Install nvidia-container-runtime library into the Docker host system. As root, run the following script:

```
./installNvidiaContainerRuntime
```

5. Ensure nvidia_uvm kernel module is loaded

Verify that the /dev/nvidia-uvm device exists and that the nvidia_uvm kernel module is loaded:

```
lsmod | grep nvidia_uvm
```

If the command produces no output, the module is not loaded. Load it on the Docker host by running the following command as root:

```
modprobe nvidia-uvm
```

4.2 Sentinel license protection system

The SDK engines used by the MMR+ANPR REST Server rely on a standard third-party licensing solution, Sentinel LDK by Thales.

For **USB-dongle licensing**, install the Sentinel LDK Run-time Environment either on the host machine or on a separate license server.

If you are using an **Eyedea trial (cloud) license**, the license is validated by a machine where the Sentinel LDK Run-time Environment is already installed. In this case, installation is not required here; you will only need to set the HASP_REMOTE_SERVERADDR application variable to the full cloud license string, as described in the Running the Docker container chapter.

1. Uncompress the package containing the Run-time Environment installer

Uncompress the aksusbd*.tar.gz file.

2. Uninstall any existing Sentinel LDK Run-time Environment

If you have installed a prior version of the Sentinel LDK Run-time Environment, as root, run the following script from the uncompressed directory to uninstall it:

```
./dunst
```

3. Install the new Sentinel LDK Run-time Environment

As root, run the following script from the uncompressed directory to install the Sentinel LDK Run-time Environment:

```
./dinst
```

Note: If you encounter the "No such file or directory" error, you may need to install the 32-bit compatibility binaries. As root, execute the following command before installing the Sentinel LDK Run-time Environment:

```
apt-get install libc6:i386
```

4. Verify the installation

To verify the Sentinel LDK Run-time Environment installation, open the address http://localhost:1947/_int_/devices.html in a web browser or check the output of the following command:

```
service aksusbd status
```

Important: When starting the container, you will need to configure the HASP_REMOTE_SERVERADDR application variable, which specifies the location of the license server. Details are provided in the Running the Docker container chapter.

4.3 Building the Docker image

0. Prerequisites

Docker (https://www.docker.com) must be installed and Docker daemon must be running before creating the Docker image.

1. Set the Docker image variables (Dockerfile)

Instructions for building the Docker image are listed in Dockerfile which is located in the [PACKAGE]/ directory. Set the NVIDIA_DRIVER_VERSION variable (line 14) to install the proper version of the NVIDIA Driver (needed for GPU computation). The same version of the Driver must be installed on the Docker host system.

The NVIDIA Driver version may be found out by running the following command:

```
nvidia-smi
```

Set the NVIDIA_DRIVER_VERSION variable to empty to disable the NVIDIA Driver installation. In that case only CPU computation will be supported.

Example of line 14 in Dockerfile which enables GPU computation:

```
ENV NVIDIA_DRIVER_VERSION=410.104
```

Example of line 14 in Dockerfile for CPU computation only:

```
ENV NVIDIA_DRIVER_VERSION=
```

To update LPM or MMR SDK, change the LPM_VERSION (line 17) or MMR_VERSION (line 20) variables, respectively. The appropriate *.tar.gz archive must be in the [PACKAGE]/ directory.

Example of line 17 in Dockerfile defining the latest LPM SDK version:

```
ENV LPM_VERSION=v7.9.1-2026-04-08-Ubuntu-20.04-x86_64-hasp10.2
```

Example of line 20 in Dockerfile defining the latest Eyedea MMR SDK version:

```
ENV MMR_VERSION=2.27.0-Ubuntu-20.04-x86_64-OpenCL-HASP
```

2. Optionally: Enable segmentations

Segmentations are a new feature of the LPM module 802 detector, allowing it to return a binary mask for each detected object. This can be useful for visualization or precise anonymization of complex objects. However, enabling segmentations slows down detection by approximately 30%, so it is disabled by default.

To enable segmentations, update the following configuration files in the [PACKAGE]/LPM-modules/802-generic.gen-gen-v7.16-2026-03-17-Ubuntu-18.04-x86_64-OpenCL-hasp10.2.tar.gz archive:

- **Modify modules-v7/x86_64/802-generic.gen-gen-v7.16-opencl/config-det.COMBINED.ini**

  The model_filename parameter in the [DETECTION MODELS] section specifies which data model is used for detection. By default, the faster models/CNN_DETECTTF2LITE_COMBINED_BGR_640x640_NONE_NONE_EXP32_enc.dat is used, but it does not support segmentations.

  To enable segmentations, comment out the default model (with #) and uncomment the segmentation-supported model model_filename="models/CNN_DETECTTF2LITE_COMBINED_BGR_640x640_NONE_NONE_EXP33_enc.dat".

  Model supporting segmentations:

  ```
  [DETECTION MODELS]
  # model_filename="models/CNN_DETECTTF2LITE_COMBINED_BGR_640x640_NONE_NONE_EXP32_enc.dat"
  # Model with segmentations
  model_filename="models/CNN_DETECTTF2LITE_COMBINED_BGR_640x640_NONE_NONE_EXP33_enc.dat"
  ```

  Model NOT supporting segmentations (default, faster):

  ```
  [DETECTION MODELS]
  model_filename="models/CNN_DETECTTF2LITE_COMBINED_BGR_640x640_NONE_NONE_EXP32_enc.dat"
  # Model with segmentations
  # model_filename="models/CNN_DETECTTF2LITE_COMBINED_BGR_640x640_NONE_NONE_EXP33_enc.dat"
  ```

- **Modify modules-v7/x86_64/802-generic.gen-gen-v7.16-opencl/config.ini**

  In the [DET PARAMETERS] section, configure the segmentation settings:

  - use_segmentation (True / False) – Enables or disables segmentations.
  - rescale_segmentation (True / False) – Determines whether segmentation masks are scaled to the input image size.

  Segmentations enabled, real size masks:

  ```
  use_segmentation    = True
  rescale_segmentation= True
  ```

  Segmentations enabled, reduced data transfer (masks require resizing by the user):

```
use_segmentation     = True
rescale_segmentation= False
```

Segmentations disabled, faster detection:

```
use_segmentation     = False
```

Note: These settings may change in future releases!

3. Run the build script (buildDocker)

Use the buildDocker script located in the [PACKAGE]/ directory to build the Docker image:

```
./buildDocker
```

The Docker image is built using the following command:

```
docker build -t mmr-anpr-rest-server .
```

The -t option specifies the name of the new Docker image.

4.4 Running the Docker container

1. Set the application variables (env-GPU.list or env-CPU.list)

The application reads its configuration (e.g., license server address, number of processing instances, etc.) from system environment variables at runtime. These variables are defined in two files:

- env-GPU.list – used when running the container with GPU support
- env-CPU.list – used when running the container with CPU-only processing

One of these files is passed to the container via the --env-file option during startup, ensuring that all required environment variables are correctly set inside the container.

In the following overview, <ENGINE> stands for: LPM_DETECTOR, LPM_PLATE_OCR, MMR_PLATE and MMR_BOX.

HASP_REMOTE_SERVERADDR	The address of the Sentinel License Manager server with a valid license. • For a USB-dongle license, specify the IP address of the machine where the dongle is plugged in. For the local host, this is typically 10.0.75.1 or 172.17.0.1. • For a cloud license (including Eyedea trial licenses), provide the full cloud license string (e.g.: ABCDEFG:aBCDeFgHij…wXyz12345@1.2.3.4).
LPM_DETECTOR_MODULE_ID	The three-digit identifiers of LPM modules used for the detection and the plate OCR, respectively. The appropriate *.tar.gz archives must be in the [PACKAGE]/LPM-modules directory. Multiple comma-separated module identifiers can be specified; the first one is used as the default. For the detection, the usage of all-in-one detection module 802 is recommended.
LPM_PLATE_OCR_MODULE_ID	
MMR_PLATE_MODEL	Specify the binary module names used for license plate based (MMR_*) and box based (MMRBOX_*) MMR, respectively. The default MMR modules are *_VCMMGVCT_* which recognize view, category, make, model, generation, variation, color and tags. If you do not have licenses for all these features or prefer fast versions (instead of default

MMR_BOX_MODEL	accelerated ones), select the appropriate data files from [PACKAGE]/Eyedea-MMR-${MMR_VERSION}.tar.gz/${MMR_PATH}/model directory. See the MMR documentation for details. Multiple comma-separated module names can be specified; the first one is used as the default.
<ENGINE>_NUM_INSTANCES	The number of engine instances to initialize (each uses one HASP license). Set the variable to 0 to disable the given engine.
<ENGINE>_NUM_THREADS	The number of internal CPU threads per engine instance. Has significant impact on performance in CPU mode, minimal in GPU mode. Setting the variable to 0 disables the given engine.
<ENGINE>_COMPUTATION_MODE	Specifies whether the given engine runs on CPU, or GPU.
<ENGINE>_GPU_ID	Relevant only for GPU computation. Specifies the 0-based index or indexes of GPU devices to compute on. If more GPUs are available, you can assign a comma separated list of indexes to a single engine (e.g.: MMR_BOX_GPU_ID=0,1) or distribute them among multiple engines (e.g.: LPM_PLATE_OCR_GPU_ID=0 MMR_PLATE_GPU_ID=1).
RECOGNITION_MAX_PIXEL_BUDGET	Maximum total number of image pixels allowed in-flight before decoding. Set 0 to disable this admission control. Use positive values to protect JVM heap when processing concurrent high-resolution images (for example, 300000000 for roughly 300 MPx in-flight).
RECOGNITION_PIXEL_BUDGET_TIMEOUT_MS	Maximum wait time in milliseconds for pixel-budget capacity. Set 0 for fail-fast behavior (requests rejected immediately with HTTP 503 when budget is exhausted), or a positive value to queue briefly during short bursts
LOG_STATS_PERIOD_REQUEST	Specifies whether and, if necessary, with what period recognition statistics should be logged to a file. Set a positive integer *N* to log recognition statistics to a new file every *N*-th request, or 0 to disable recognition statistics logging. See Log files chapter for details.
LOG_ERRORS	Specifies whether to log errors to a file. Set 1 to enable, or 0 to disable errors and warnings logging. See Log files chapter for details.

Check the SDK Engine Overview chapter to select the appropriate SDK engines for your use case.

Example:

Let's have the env-GPU.list file with the following configuration (only the relevant variables listed for brevity):

```
HASP_REMOTE_SERVERADDR=172.17.0.1
LPM_DETECTOR_MODULE_ID=802
LPM_DETECTOR_NUM_INSTANCES=1
LPM_DETECTOR_NUM_THREADS=1
LPM_DETECTOR_COMPUTATION_MODE=GPU
LPM_DETECTOR_GPU_ID=0
LPM_PLATE_OCR_MODULE_ID=801,800
LPM_PLATE_OCR_NUM_INSTANCES=1
LPM_PLATE_OCR_NUM_THREADS=1
LPM_PLATE_OCR_COMPUTATION_MODE=GPU
LPM_PLATE_OCR_GPU_ID=0
MMR_PLATE_MODEL=MMR_VCCT_FAST_2026Q2.dat
MMR_PLATE_NUM_INSTANCES=1
```

```
MMR_PLATE_NUM_THREADS=1
MMR_PLATE_COMPUTATION_MODE=GPU
MMR_PLATE_GPU_ID=0
MMR_BOX_NUM_INSTANCES=0
RECOGNITION_MAX_PIXEL_BUDGET=300000000
RECOGNITION_PIXEL_BUDGET_TIMEOUT_MS=30000
LOG_STATS_PERIOD_REQUEST=1000
LOG_ERRORS=1
```

The license key with the valid licenses must be plugged into the current machine.

There will be four running SDK engines: LPM Detector, two different modules of LPM Plate OCR, and License Plate based MMR, each of them running in one instance on the GPU. The Box based MMR engine will be disabled.

LPM Detector will use the all-in-one detection module 802; LPM Plate OCR will use the global module 801 (as the default one) and the European module 800 (used if requested); the MMR will use the fast module to recognize only view, category, color and tags.

The pixel admission gateway will allow up to 300 MPx of decoded images in-flight and wait up to 30 seconds for capacity before returning HTTP 503.

Recognition statistics will be saved to a file after every 1000th recognition request is processed. Any error and warning occurrences will be saved to files, too.

Notes:

- Use GPU computation whenever available for best performance.
- Accelerated (*_ACCE_*) MMR modules are optimized for GPU execution. When running in CPU mode, a fast (*_FAST_*) MMR module is generally recommended.
- Tune the CPU configuration based on measurements with representative data. As a starting point, keep the total number of active LPM and MMR processing threads below approximately twice the number of CPU cores, and allocate most threads to the detector.
- For deployments receiving concurrent high-resolution images (or running with tight JVM heap limits), enable pixel admission by setting RECOGNITION_MAX_PIXEL_BUDGET to a positive value and configure RECOGNITION_PIXEL_BUDGET_TIMEOUT_MS. This helps prevent decode-time memory spikes and provides controlled backpressure (HTTP 503) instead of unstable out-of-memory behavior.

2. Run the container (runDockerGPU or runDockerCPU)

To start the REST server, use one of the predefined scripts located in the [PACKAGE]/ directory:

- runDockerGPU – launches the container with GPU support.
- runDockerCPU – launches the container for CPU-only operation.

These scripts run the Docker container built from the mmr-anpr-rest-server image. You may modify them as needed to fit your environment.

Both runDockerGPU and runDockerCPU scripts manage the container lifecycle as follows:

- If the container does not exist: it is created and started.
- If the container exists but is stopped: it is removed and recreated.
- If the container is already running: a warning is displayed, and no action is taken.

The container is launched using the following commands:

- GPU version (runDockerGPU):

```
docker run \
  --name mmr-anpr-rest-server \
  --env-file env-GPU.list \
```

```
  -p 8080:8080 \
  --gpus device=0 \
  mmr-anpr-rest-server
```

- CPU version (runDockerCPU):

```
docker run \
  --name mmr-anpr-rest-server \
  --env-file env-CPU.list \
  -p 8080:8080 \
  mmr-anpr-rest-server
```

Docker options explained:

- --name: Assigns a name to the container.
- --env-file: Loads environment variables from the specified file.
- -p: Maps container port 8080 to the host port 8080.
- --gpus: Grants GPU access to the container (used only in GPU mode). Examples:
  - --gpus device=0 – use GPU with index 0
  - --gpus '"device=2,3"' – use GPUs 2 and 3
  - --gpus all – use all available GPUs

You can add other Docker options if needed (e.g., the -v option as described in the Log files chapter).

Important: If the USB dongle is connected to a remote license server, or if you are using a cloud-based license, you may need to add the --network host option to avoid the "Sentinel LDK Protection System: Sentinel protection key was not found (H0007)" error. However, do not use --network host when the USB dongle is attached directly to the local host machine, as host networking prevents the container from accessing USB dongles passed from the host.

The final argument (mmr-anpr-rest-server) specifies the Docker image used to create the container.

The server attempts to initialize all engine instances defined in the selected environment file (env-GPU.list or env-CPU.list). Check the error output for details if there are problems starting any engine. After launching the Docker container, the application is accessible at:

```
http://[MACHINE_IP]:8080/RESTServer/
```

You can check the status of the engines on the application home page or by requesting the serverSystemInfo. "DISABLED" means that no instance (or no thread) was requested for the given engine, "FAILED TO START" means that all requested instances of the given engine could not be started; in both cases, the server handles requests and returns data obtained by other running engines (unless the user request explicitly includes a task processed by an engine that is not running).

The engine status never changes when the container is started. If an engine encounters a license problem after the successful initialization, its status will still be "RUNNING", but the response to a request for that engine will have a status code of 500 with a message indicating the problematic engine (e.g.: "Error during OCR processing. Computation error. Please check your licenses.").

4.5 Stopping the Docker container

To stop the running container, you can use the stopDocker script located in the [PACKAGE]/ directory:

```
./stopDocker
```

This script gracefully stops the Docker container named mmr-anpr-rest-server using the following command:

```
docker stop mmr-anpr-rest-server
```

Important: Do not use the docker kill command to stop the container. It forcefully terminates the container without allowing the application to clean up resources properly. Always stop the container gracefully to ensure proper release of system resources and software licenses.

4.6 Log files

There is a possibility to enable logging of recognition statistics and errors in order to regularly monitor the server usage and report problems. Logging can be enabled by setting the corresponding variable in the selected environment file (env-GPU.list or env-CPU.list).

If enabled, each statistics record or problem produces a new file <Type>_<year>-<month>-<day>_<hour>-<minute>-<second>-<millisecond>.log in the Docker container's /usr/local/tomcat/webapps/RESTServer_data/logs/ folder.

Log file name

The following table lists possible values of <Type> in the log file name:

Type	Description
Stats	Recognition statistics.
BadRequest	400 Bad Request response status code is returned.
ErrorResponse	500 Internal Server Error or 503 Service Unavailable response status code is returned.
RuntimeError	More serious problem during server startup or recognition processing.
RuntimeWarning	Less serious problem during server startup or recognition processing.

Recognition statistics

Recognition statistics log files contain a JSON object which contains the following items:

Response item	Description	Data type
requestTimestamp	Date and time of the user request with a millisecond precision.	String
responseTimestamp	Date and time of the server response with a millisecond precision.	String
serverStartedTimestamp	Date and time when the server started with a millisecond precision.	String
totalTaskCount	Number of user recognition requests (including erroneous).	Integer
totalDetectionCount	Number of processed detection computation tasks.	Integer
totalOcrCount	Number of processed OCR computation tasks.	Integer
totalMmrCount	Number of processed MMR computation tasks.	Integer

Errors

Error log files contain a text message describing the problem.

Notes

To store log files on the host, mount a volume when running the Docker container by adding the following Docker options to the runDockerGPU / runDockerCPU script:

```
-v $PWD/logs:/usr/local/tomcat/webapps/RESTServer_data/logs
```

By default, the Docker container uses UTC for timestamps in log files. To use the host's time zone instead, add:

```
-v /etc/timezone:/etc/timezone:ro
```

5 REST API Documentation

5.1 SDK Engine Overview

Which SDK engines do you need? Use the combination of the SDK engines from the following table.

Use Case	SDK Engine
Plate detection	LPM Detector
Vehicle detection	LPM Detector *(module 802)*
Windshield detection	LPM Detector *(module 802)*
Wheel detection	LPM Detector *(module 802)*
Segmentation	LPM Detector *(module 802, if explicitly enabled)*
Plate OCR	LPM Plate OCR
Vehicle recognition	LPM License Plate based *or* MMR Box based

LPM Detector detects plates on vehicles (license plates, ADR and trash plates, etc.). Module 802 also detects boxes (whole vehicles and some other "road users" like pedestrians, kickbikes, etc.), windshields and wheels. Module 802 is currently the only module that supports grouping of road users into combinations; if explicitly enabled when building the Docker image, it also supports segmentation.

LPM Plate OCR recognizes the text and type of plates detected by LPM License Plate Detector or provided in the request.

MMR License Plate based can recognize the view, category, make, model, generation, variation, color and tags of vehicles specified by their license plate. Either LPM Detector detecting plates is needed, or the license plate detections must be provided in the request.

MMR Box based recognizes the view, category, make, model, generation, variation, color and tags of vehicles (or "road users") specified by their bounding box. Either LPM Detector detecting boxes (module 802) is needed, or the box detections must be provided in the request.

LPM modules

The following table lists the LPM modules shipped with the current version of MMR+ANPR REST Server.

LPM module	Detected objects	OCR region
553	Plates	Americas
555	Plates	Middle East and Africa (selected countries)
556	Plates	Asia (selected countries)
557	Plates	Oceania
800	Plates	Europe

801	Plates	Worldwide (selected countries)
802	Plates, boxes, windshields, wheels Segmentation (if explicitly enabled)	-

Whether an LPM module is available depends on the MMR+ANPR REST Server settings. See the chapter Running the Docker container for details.

MMR modules

The following table explains the meaning of each part of the MMR binary module names shipped with the current version of MMR+ANPR REST Server.

MMR module	**Description**
MMR_*	License plate based MMR module
MMRBOX_*	Box based MMR module
*_VCCT_*	Recognizes view, category, color and tags
*_VCMMCT_*	Recognizes view, category, make, model, color and tags
*_VCMMGVCT_*	Recognizes view, category, make, model, generation, variation, color and tags
*_FAST_*	Fast MMR module
*_ACCE_*	Accelerated MMR module

For a more detailed description, check the Eyedea MMR SDK documentation.

Whether an MMR module is available depends on the MMR+ANPR REST Server settings. See the chapter Running the Docker container for details.

5.2 Response Status Codes

The following table lists all response status codes MMR+ANPR REST Server returns.

Status Code	**Meaning**
200	Success.
400	Bad Request. The request was invalid. The error message provides the details.
500	Internal Server Error. Most of these errors are caused by a licensing issue. The error message provides the details.
503	Service Unavailable. The server is temporarily unable to accept the request (for example, pixel admission budget is exhausted or waiting timed out); retry later or use a smaller image.

5.3 Entry point

SERVER_IP:8080/RESTServer

5.4 System Info

System info contains information about system resources and SDK engines.

When pixel admission is enabled, the monitoring also reports admission pressure (in-flight pixels, max pixel budget, waiting requests) so bottlenecks before engine queues are visible.

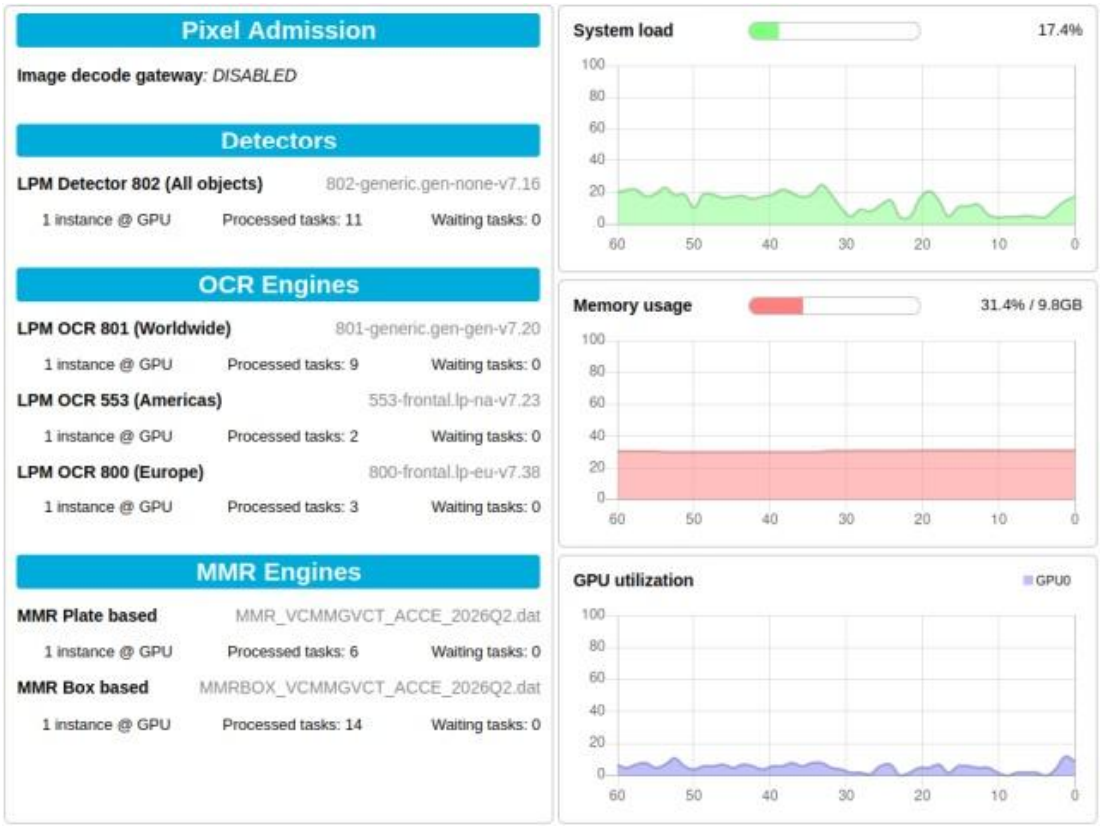
Main web page with real time server monitoring

Endpoint:	/
HTTP Method:	GET
Consumes Media Type:	-
Produces Media Type:	text/html
URL Parameters:	-
Data Parameters:	-

CURL Command Example:

```
curl -X GET -H "Accept: text/html" http://localhost:8080/RESTServer/
```

Response Example:



Text output with server monitoring information

Endpoint:	/
HTTP Method:	GET
Consumes Media Type:	-
Produces Media Type:	text/plain
URL Parameters:	-
Data Parameters:	-

CURL Command Example:

```
curl -X GET -H "Accept: text/plain" http://localhost:8080/RESTServer/
```

Response Example:

```
Current time: 2026-07-03 07:24:37
Started on: 2026-07-03 06:23:09

System:
 - Average load: 17.39%
 - Used memory: 31.47% (9.84 GB)

GPU 0 (NVIDIA GeForce GTX 1660):
 - GPU Utilization: 2%
 - Free memory: 4069 / 5746 MB

Pixel admission:
 - Enabled: NO

LPM Detector 802:
 - Engine version: v7.9.1-2026-04-08-Ubuntu-20.04-x86_64-hasp10.2
 - Module ID: 802
 - Module version: 802-generic.gen-none-v7.16
 - Computation mode: GPU
 - Status: RUNNING
 - Number of instances: 1
 - Number of internal threads: 1
 - Number of erroneous tasks: 0
 - Number of processed tasks: 11
 - Number of waiting tasks: 0

LPM OCR 801:
 - Engine version: v7.9.1-2026-04-08-Ubuntu-20.04-x86_64-hasp10.2
 - Module ID: 801
 - Module version: 801-generic.gen-gen-v7.20
 - Computation mode: GPU
 - Status: RUNNING
 - Number of instances: 1
 - Number of internal threads: 1
 - Number of erroneous tasks: 0
 - Number of processed tasks: 9
 - Number of skipped tasks: 0
 - Number of waiting tasks: 0

LPM OCR 553:
 - Engine version: v7.9.1-2026-04-08-Ubuntu-20.04-x86_64-hasp10.2
 - Module ID: 553
 - Module version: 553-frontal.lp-na-v7.23
 - Computation mode: GPU
```

```
 - Status: RUNNING
 - Number of instances: 1
 - Number of internal threads: 1
 - Number of erroneous tasks: 0
 - Number of processed tasks: 2
 - Number of skipped tasks: 0
 - Number of waiting tasks: 0

LPM OCR 800:
 - Engine version: v7.9.1-2026-04-08-Ubuntu-20.04-x86_64-hasp10.2
 - Module ID: 800
 - Module version: 800-frontal.lp-eu-v7.38
 - Computation mode: GPU
 - Status: RUNNING
 - Number of instances: 1
 - Number of internal threads: 1
 - Number of erroneous tasks: 0
 - Number of processed tasks: 3
 - Number of skipped tasks: 0
 - Number of waiting tasks: 0

MMR Plate based:
 - Engine version: 2.27.0-Ubuntu-20.04-x86_64-OpenCL-HASP
 - Module: MMR_VCMMGVCT_ACCE_2026Q2.dat
 - Module version: 20260523
 - Computation mode: GPU
 - Status: RUNNING
 - Number of instances: 1
 - Number of internal threads: 1
 - Number of erroneous tasks: 0
 - Number of processed tasks: 6
 - Number of skipped tasks: 0
 - Number of waiting tasks: 0

MMR Box based:
 - Engine version: 2.27.0-Ubuntu-20.04-x86_64-OpenCL-HASP
 - Module: MMRBOX_VCMMGVCT_ACCE_2026Q2.dat
 - Module version: 20260522
 - Computation mode: GPU
 - Status: RUNNING
 - Number of instances: 1
 - Number of internal threads: 1
 - Number of erroneous tasks: 0
 - Number of processed tasks: 14
 - Number of skipped tasks: 0
 - Number of waiting tasks: 0
```

Get detailed server system info

Endpoint:	/info
HTTP Method:	GET
Consumes Media Type:	-
Produces Media Type:	application/json
URL Parameters:	-
Data Parameters:	-
CURL Command Example:	

```
curl -X GET http://localhost:8080/RESTServer/info
```

Response Example:


```
{
  "timestamp": 1783063483919,
  "serverStartedTimestamp": 1783059789350,
  "system": {
    "cpuUsage": 17.555994,
    "memoryUsage": 31.490738,
    "memoryUsageBytes": 10569011200,
    "gpus": [
      {
        "id": 0,
        "name": "NVIDIA GeForce GTX 1660",
        "totalMemoryMB": 5746,
        "freeMemoryMB": 4068,
        "gpuUtilizationPerc": 2
      }
    ]
  },
  "pixelAdmission": {
    "enabled": false
  },
  "engines": [
    {
      "computationMode": "GPU",
      "engineName": "LPM",
      "engineType": "Detector",
      "engineVersion": "v7.9.1-2026-04-08-Ubuntu-20.04-x86_64-hasp10.2",
      "moduleId": 802,
      "moduleVersion": "802-generic.gen-none-v7.16",
      "numErroneousTasks": 0,
      "numInstances": 1,
      "numInternalThreads": 1,
      "numProcessedTasks": 11,
      "numSkippedTasks": 0,
      "numWaitingTasks": 0,
      "status": "RUNNING",
      "task": "DETECTION"
    },
    {
      "computationMode": "GPU",
      "engineName": "LPM",
      "engineType": "OCR",
      "engineVersion": "v7.9.1-2026-04-08-Ubuntu-20.04-x86_64-hasp10.2",
      "moduleId": 801,
      "moduleVersion": "801-generic.gen-gen-v7.20",
      "numErroneousTasks": 0,
      "numInstances": 1,
      "numInternalThreads": 1,
      "numProcessedTasks": 9,
      "numSkippedTasks": 0,
      "numWaitingTasks": 0,
      "status": "RUNNING",
      "task": "OCR"
    },
    {
      "computationMode": "GPU",
      "engineName": "LPM",
      "engineType": "OCR",
      "engineVersion": "v7.9.1-2026-04-08-Ubuntu-20.04-x86_64-hasp10.2",
      "moduleId": 553,
      "moduleVersion": "553-frontal.lp-na-v7.23",
      "numErroneousTasks": 0,
      "numInstances": 1,
      "numInternalThreads": 1,
```

```
      "numProcessedTasks": 2,
      "numSkippedTasks": 0,
      "numWaitingTasks": 0,
      "status": "RUNNING",
      "task": "OCR"
    },
    {
      "computationMode": "GPU",
      "engineName": "LPM",
      "engineType": "OCR",
      "engineVersion": "v7.9.1-2026-04-08-Ubuntu-20.04-x86_64-hasp10.2",
      "moduleId": 800,
      "moduleVersion": "800-frontal.lp-eu-v7.38",
      "numErroneousTasks": 0,
      "numInstances": 1,
      "numInternalThreads": 1,
      "numProcessedTasks": 3,
      "numSkippedTasks": 0,
      "numWaitingTasks": 0,
      "status": "RUNNING",
      "task": "OCR"
    },    {
      "computationMode": "GPU",
      "engineName": "MMR",
      "engineType": "Plate based",
      "engineVersion": "2.27.0-Ubuntu-20.04-x86_64-OpenCL-HASP",
      "moduleName": "MMR_VCMMGVCT_ACCE_2026Q2.dat",
      "moduleVersion": "20260523",
      "numErroneousTasks": 0,
      "numInstances": 1,
      "numInternalThreads": 1,
      "numProcessedTasks": 6,
      "numSkippedTasks": 0,
      "numWaitingTasks": 0,
      "status": "RUNNING",
      "task": "MMR"
    },
    {
      "computationMode": "GPU",
      "engineName": "MMR",
      "engineType": "Box based",
      "engineVersion": "2.27.0-Ubuntu-20.04-x86_64-OpenCL-HASP",
      "moduleName": "MMRBOX_VCMMGVCT_ACCE_2026Q2.dat",
      "moduleVersion": "20260522",
      "numErroneousTasks": 0,
      "numInstances": 1,
      "numInternalThreads": 1,
      "numProcessedTasks": 14,
      "numSkippedTasks": 0,
      "numWaitingTasks": 0,
      "status": "RUNNING",
      "task": "MMR"
    }
  ]
}
```


Response Definitions:

The following table describes each item in the response.

Response item	Description	Data type
timestamp	Current system time in milliseconds.	Integer
serverStartedTimestamp	System time in milliseconds when the server started.	Integer

system	System information	Object
system/cpuUsage	Recent CPU usage for the whole system in percent.	Decimal number
system/memoryUsage	Amount of physical memory used by any process in percent.	Decimal number
system/memoryUsageBytes	Amount of physical memory used by any process in bytes.	Integer
system/gpus	Information about available GPU devices.	Array of objects
system/gpus/id	GPU device ID.	Integer
system/gpus/name	GPU device name.	String
system/gpus/totalMemoryMB	Total amount of physical GPU device memory in megabytes.	Integer
system/gpus/freeMemoryMB	Amount of free physical GPU device memory in megabytes.	Integer
system/gpus/gpuUtilizationPerc	Utilization of the GPU device in percent.	Integer
pixelAdmission	Admission-control summary for image decoding pressure. Present in all responses.	Object
pixelAdmission/enabled	Indicates whether pixel admission is enabled (non-zero RECOGNITION_MAX_PIXEL_BUDGET).	Boolean
pixelAdmission/maxPixelBudget	Configured maximum number of in-flight pixels. Omitted when admission is disabled.	Integer
pixelAdmission/inFlightPixels	Current number of in-flight pixels that already acquired admission budget. Omitted when admission is disabled.	Integer
pixelAdmission/waitingRequests	Current number of requests waiting for admission budget. Omitted when admission is disabled.	Integer
pixelAdmission/admittedRequests	Total number of requests admitted through pixel budget control since server start. Omitted when admission is disabled.	Integer
pixelAdmission/rejectedRequests	Total number of requests rejected by pixel admission since server start (timeout, interruption, or image larger than budget). Omitted when admission is disabled.	Integer
engines	Information about the SDK engines.	Array of objects
engines/computationMode	Type of processing unit used for computation. Either "CPU", or "GPU".	String
engines/engineName	SDK engine name.	String
engines/engineType	SDK engine type. Either "Detector", "OCR", or "MMR".	String
engines/engineVersion	SDK engine version.	String
engines/moduleId	DK engine module ID.	Integer
engines/moduleName	SDK engine module name.	String

engines/moduleVersion	SDK engine module version.	String
engines/numInstances	Number of initialized engine instances.	Integer
engines/numInternalThreads	Number of internal CPU threads per engine instance.	Integer
engines/numErroneousTasks	Number of computational tasks which ended in an error.	Integer
engines/numProcessedTasks	Number of successfully processed computational tasks.	Integer
engines/numWaitingTasks	Number of queued computational tasks to be processed.	Integer
engines/status	SDK engine status. Either "RUNNING" (at least 1 instance successfully initialized), "DISABLED" (no instance or no thread requested), or "FAILED TO START" (all requested instances could not be started).	String
engines/task	The computational task associated with the SDK engine. Either "DETECTION", "OCR", or "MMR".	String

Notes:

If pixel admission is disabled, the **pixelAdmission** object contains only **enabled=false**; all other pixel-admission fields are omitted.

5.5 Recognition

Recognition provides information about recognized road users and their components in the input image.

Supported image formats: JPG, JPEG, PNG, BMP, TIFF, WEBP, AVIF.

Web page with image upload and recognition functionality

Endpoint:	/recognition
HTTP Method:	GET
Consumes Media Type:	-
Produces Media Type:	text/html
URL Parameters:	-
Data Parameters:	-
CURL Command Example:	`curl -X GET http://localhost:8080/RESTServer/recognition`
Response Examples: **All supported objects:**	

Boxes only:

Notes:

LPM detector, LPM Plate OCR and MMR (compatible with the detector) engines must be running for both ANPR and MMR results to be displayed.

The ANPR section displays the OCR result of *license* plates. If multiple license plates are detected on one vehicle, the OCR of the most confident detection is selected.

Basic information about the detected objects in the image is displayed as a tooltip when you hover the mouse cursor over them.

Occluded attribute is based on the value of **occlusion**: if occlusion is greater or equal to 0.25, the plate is considered to be occluded (marked with a green ✓); otherwise it is considered to be not occluded (marked with a red ✗). The plate is considered unreadable if unreadable is greater or equal to 0.98.

The score of an MMR tag (in the MMR tags section) is displayed as a tooltip when you hover the mouse cursor over it.

Detect and/or recognize road user objects in the input image

Endpoint:	/recognition
HTTP Method:	POST
Consumes Media Type:	multipart/form-data
Produces Media Type:	application/json
URL Parameters:	-

Data Parameters:

file – form data parameter containing input image

request – (optional) JSON object specifying the recognition request:

Request item	Description	Data type
tasks	An array of requested computational tasks: "DETECTION", "SEGMENTATION", "OCR", "MMR" (case insensitive). Optional; if absent, all available tasks are applied depending on other request items. The SEGMENTATION task is only allowed if the DETECTION task is included (SEGMENTATION cannot be used alone or with tasks other than DETECTION).	Array of strings; *optional*
combinations	An array containing combinations of detected road user objects (box and/or plates). Road users in a combination belong to each other and are usually physically connected. If absent, the detections must be obtained by the detector using the DETECTION task. Allowed only in combination with the [OCR, MMR], [OCR] and [MMR] tasks (without the DETECTION task).	Array of objects; *optional*
combinations/roadUsers	An array containing road users belonging to the combination. Road users sharing the same combination belong to each other and are usually physically connected.	Array of objects
combinations/roadUsers/box	Information about the detected vehicle.	Object; *optional*
combinations/roadUsers/box/position	Information about the detected vehicle bounding box location in the input image.	Object
combinations/roadUsers/box/position/topLeftCol	The horizontal coordinate of the top-left corner of the box within the image.	Decimal number
combinations/roadUsers/box/position/topLeftRow	The vertical coordinate of the top-left corner of the box within the image.	Decimal number
combinations/roadUsers/box/position/bottomRightCol	The horizontal coordinate of the bottom-right corner of the box within the image.	Decimal number
combinations/roadUsers/box/position/bottomRightRow	The vertical coordinate of the bottom-right corner of the box within the image.	Decimal number
combinations/roadUsers/plates	Information about the detected plates (license plate, ADR or trash plate, etc.).	Array of objects; *optional*

combinations/roadUsers/plates/**position**	Information about the detected plate location in the input image. Mandatory for OCR. Optional for MMR; if absent, the center coordinates, rotation and pixelsPerMeter must be provided.	Object; *optional*
combinations/roadUsers/plates/position/**topLeftCol**	The horizontal coordinate of the top-left corner of the plate within the image.	Decimal number
combinations/roadUsers/plates/position/**topLeftRow**	The vertical coordinate of the top-left corner of the plate within the image.	Decimal number
combinations/roadUsers/plates/position/**topRightCol**	The horizontal coordinate of the top-right corner of the plate within the image.	Decimal number
combinations/roadUsers/plates/position/**topRightCol**	The vertical coordinate of the top-right corner of the plate within the image.	Decimal number
combinations/roadUsers/plates/position/**bottomRightCol**	The horizontal coordinate of the bottom-right corner of the plate within the image.	Decimal number
combinations/roadUsers/plates/position/**bottomRightCol**	The vertical coordinate of the bottom-right corner of the plate within the image.	Decimal number
combinations/roadUsers/plates/position/**bottomLeftCol**	The horizontal coordinate of the bottom-left corner of the plate within the image.	Decimal number
combinations/roadUsers/plates/position/**bottomLeftRow**	The vertical coordinate of the bottom-left corner of the plate within the image.	Decimal number
combinations/roadUsers/plates/**centerCol**	The horizontal coordinate of the plate center within the image. Ignored for OCR. Optional for MMR; if absent, position (the coordinates of all four corners) must be provided.	Decimal number; *optional*
combinations/roadUsers/plates/**centerRow**	The vertical coordinate of the plate center within the image. Ignored for OCR. Optional for MMR; if absent, position (the coordinates of all four corners) must be provided.	Decimal number; *optional*
combinations/roadUsers/plates/**rotation**	Clockwise correction of the source image in degrees. Ignored for OCR. Optional for MMR; if absent, position (the coordinates of all four corners) must be provided.	Decimal number; *optional*
combinations/roadUsers/plates/**pixelsPerMeter**	The real-world scale in pixels per meter of the image computed on the plate. For example, plate of width = 300 millimeters, photographed to have 150 pixels, has 150 / 0.3 = 500 pixels per meter scale. Ignored for OCR. Optional for MMR; if absent, position (the coordinates of all four corners) must and dimension should be provided.	Decimal number; *optional*
combinations/roadUsers/plates/**dimension**	The physical dimensions of the plate. Ignored for OCR. Optional for MMR; if absent, pixelsPerMeter should be provided.	Object; *optional*
combinations/roadUsers/plates/dimension/**width**	The physical width of the plate in millimeters.	Integer
combinations/roadUsers/plates/dimension/**height**	The physical height of the plate in millimeters.	Integer
requestedDetectionTypes	An array of object types to be detected by the detector: "BOX", "PLATE", "WINDSHIELD", "WHEEL" (case insensitive). Optional; if absent, detections of all types supported by the detector will be returned. Allowed only if the tasks contain DETECTION.	Array of objects; *optional*

requestedSegmentationTypes	An array of object types for which segmentations are returned: "BOX", "PLATE", "WINDSHIELD", "WHEEL" (case insensitive). Optional; if absent, segmentations for all detected objects will be returned. Must be a subset of requestedDetectionTypes, if specified. Allowed only if the tasks contain SEGMENTATION.	Array of objects; *optional*
roi	The area of the input image to be scanned by the detector. Optional; if absent, the whole image will be scanned. Allowed only if the tasks contain DETECTION.	Object; *optional*
roi/topLeftCol	The horizontal coordinate of the top-left corner of the region of interest within the image.	Integer
roi/topLeftRow	The vertical coordinate of the top-left corner of the region of interest within the image.	Integer
roi/bottomRightCol	The horizontal coordinate of the bottom-right corner of the region of interest within the image.	Integer
roi/bottomRightRow	The vertical coordinate of the bottom-right corner of the region of interest within the image.	Integer
detectionModuleId	The three-digit LPM Detector identifier to be used for the detection (see the LPM modules table). If absent, the default one will be used. Allowed only if the tasks contain DETECTION.	Integer; *optional*
ocrModuleId	The three-digit LPM Plate OCR identifier to be used for the OCR computation (see the LPM modules table). Optional; if absent, the default one will be used. Allowed only for the requestedDetectionTypes containing PLATE and tasks containing OCR.	Integer; *optional*
mmrPreference	The preferred type of object for which the MMR is computed. Either "BOX", or "PLATE" (case insensitive). Optional; if absent, BOX is preferred. Allowed only if the tasks contain MMR. Must not be combined with the mmrModuleNames containing a single module name.	String; *optional*
mmrModuleNames	An object containing the names of the binary modules used for the MMR computation. Optional; if absent, the default ones (depending on the mmrPreference) will be used. Allowed only if the tasks contain MMR.	Object; *optional*
mmrModuleNames/box	The name of the binary module used for the MMR Box computation, e.g. "MMRBOX_VCMMGVCT_ACCE_2026Q2.dat" (case insensitive). Optional; if absent, the MMR Box engine will not be used.	String; *optional*
mmrModuleNames/plate	The name of the binary module used for the MMR Plate computation, e.g. "MMR_VCMMGVCT_ACCE_2026Q2.dat" (case insensitive). Optional; if absent, the MMR Plate engine will not be used.	String; *optional*
constraints	An object containing the user-supplied execution limits. If any constraint is reached, the server stops further processing and returns the partial result available. Optional; if absent, no execution limits are applied.	Object; *optional*
constraints/maxProcessingTimeMs	Maximum allowed cumulative processing time for the request, in milliseconds. Only actual task processing time is counted; queue waiting time is not included.	Integer; *optional*
constraints/maxOcrTasks	Maximum number of OCR tasks processed for this request. OCR tasks exceeding the limit are skipped.	Integer; *optional*

constraints/maxMmrTasks	Maximum number of MMR tasks processed for this request. MMR tasks exceeding the limit are skipped.	Integer; *optional*

Notes:

Request can contain manual detections. In that case, the **tasks** must not contain "DETECTION" (the detector will not be used). Each **roadUser** (vehicle) can contain one **box** and/or one or more **plates**. The encapsulating **combinations** are sets of one or more **roadUsers**, e.g. a tractor with a trailer. The **combinations** structure will be projected into the response.

The segmentation-related request fields will only take effect if they are supported by the detector (currently only module 802) and are explicitly enabled when building the Docker image.

For the OCR of a manually specified **plate**, its **position** (all four corners) must be specified.

If a **roadUser** contains more than one detected object (and MMR **task** is requested), the MMR is computed using only one of them based on the **mmrPreference**. If there are multiple plates and one of them is to be used for the MMR, the license plate with the highest detection score will be selected.

If the **mmrModuleNames** element is specified with both **box** and **plate**, the **box** module will be used for the MMR Box computation, and the plate module will be used for the MMR Plate computation. If only one of these modules is specified, only the corresponding engine will be used for the MMR computation. The **mmrPreference** is allowed only when **mmrModuleNames** include both **box** and **plate** elements or when **mmrModuleNames** are not specified at all (using the default MMR modules).

For the MMR of a manually specified **plate**, its center coordinates (**centerCol** and **centerRow**), **rotation** and **pixelsPerMeter** need to be known. If not specified, they can be estimated from the plate corners (**position**); however, for the more accurate estimation of pixelsPerMeter parameter, **dimension** should also be specified.

CURL Command Examples:

Default request: detection, possibly segmentation, plate OCR and MMR

```
curl -X POST \
     -H "Content-Type: multipart/form-data" \
     -F "file=@image.jpg" \
     http://localhost:8080/RESTServer/recognition
```

Request in CURL command: plate detection and possibly segmentation, OCR and MMR

```
curl -X POST \
     -H "Content-Type: multipart/form-data" \
     -F "file=@image.jpg" \
     -F "request={\"requestedDetectionTypes\":[\"PLATE\"]}" \
     http://localhost:8080/RESTServer/recognition
```

Request in JSON string (see JSON string examples below)

```
curl -X POST \
     -H "Content-Type: multipart/form-data" \
     -F "file=@image.jpg" \
     -F "request=@request.json" \
     http://localhost:8080/RESTServer/recognition
```

JSON string example: box and plate detection and (plate) OCR

```
{
  "tasks": [ "DETECTION", "OCR" ],
  "requestedDetectionTypes": [ "BOX", "PLATE" ]
```

```
}
```

JSON string example: box detection, possibly segmentation and MMR in the specified region of interest

```
{
  "requestedDetectionTypes": [ "BOX" ],
  "roi": {
    "topLeftCol": 176, "topLeftRow": 362,
    "bottomRightCol": 1234, "bottomRightRow": 1000
  }
}
```

JSON string example: detection, possibly segmentation, (plate) OCR and MMR for American region (LPM Plate OCR module 553)

```
{
  "ocrModuleId": 553
}
```

JSON string example: MMR based on the specified box detection

```
{
  "combinations": [{ "roadUsers": [{ "box": {
    "position": {
      "topLeftCol": 5, "topLeftRow": 69,
      "bottomRightCol": 954, "bottomRightRow": 642
    }
  } }] }]
}
```

JSON string example: MMR based on the specified plate detection

```
{
  "combinations": [{ "roadUsers": [{ "plates": [{
    "centerCol": 176.5,
    "centerRow": 362.0,
    "rotation": -5.1,
    "pixelsPerMeter": 214.3
  }] }] }]
}
```

JSON string example: OCR and MMR based on the specified plate detection

```
{
  "combinations": [{ "roadUsers": [{ "plates": [{
    "position": {
      "topLeftCol": 817.2, "topLeftRow": 892.5,
      "topRightCol": 1181.4, "topRightRow": 880.0,
      "bottomRightCol": 1184.0, "bottomRightRow": 956.5,
      "bottomLeftCol": 819.9, "bottomLeftRow": 969.0
    }
  }] }] }]
}
```

Notes:

Content type of the request is **multipart/form-data**, where the inputs are stored in the form data fields.

Input image is referenced with filepath (starting with @) in the form field **file**.

Optional parameter **request** can be referenced with filepath (starting with @, referencing a text file containing JSON string) or can contain the whole JSON as a string (special characters must be escaped).

There are many conditions limiting the use of optional **request** items. If the element combination is not allowed, the Bad Request error code 400 is returned with a message describing the problem. Some of the illegal combinations:

- **tasks** containing "DETECTION" with manual detections (if the **combinations** element is present, no detector is used).
- **mmrPreference** with **tasks** explicitly not containing "MMR" (the **mmrPreference** element is related to the MMR task).

If the requested **detectionModuleId**, **ocrModuleId** or **mmrModuleNames** engines are not available, the Bad Request error code 400 is returned with a message describing the problem.

If the requested engines (specified explicitly by **tasks**, **detectionModuleId**, **ocrModuleId** or **mmrModuleNames** elements) are available but not running, the Internal Server error code 500 is returned with a message describing the problem. This makes the difference between the following cases:

- If the request contains **tasks** containing "MMR" (e.g.: "tasks": ["DETECTION", "OCR", "MMR"]) and none of the MMR engines are running, the Internal Server error code 500 is returned.
- If the request does not contain **tasks** element at all and all MMR engines are disabled, the server runs just the detection and plate OCR.

Response Example:

Automatic detection of all object types, segmentation, OCR and MMR (vehicle recognition):


```
{
  "combinations": [
    {
      "roadUsers": [
        {
          "box": {
            "position": {
              "topLeftCol": 156.95068,
              "topLeftRow": 275.35706,
              "bottomRightCol": 1555.0796,
              "bottomRightRow": 1220.435,
              "score": 0.99999845
            },
            "segmentation": {
              "image": {
                "base64": "iVBORw0KGgoAAAANSUhEUgAAAHEAAAB4CAAAAADgGk...",
                "format": "png"
              },
              "topLeftCol": 156,
              "topLeftRow": 275,
              "width": 1395,
              "height": 945
            },
            "occlusion": 0.07348359
          },
          "plates": [
            {
              "position": {
                "topLeftCol": 815.6459,
                "topLeftRow": 895.39453,
                "topRightCol": 1184.9958,
                "topRightRow": 876.7175,
                "bottomRightCol": 1186.2727,
                "bottomRightRow": 947.059,
                "bottomLeftCol": 819.57495,
                "bottomLeftRow": 969.176,
                "score": 0.9458341
              },
              "segmentation": {
                "image": {
                  "base64": "iVBORw0KGgoAAAANSUhEUgAAAB8AAAALCAAAAACigg...",
                  "format": "png"
```

```
            },
            "topLeftCol": 815,
            "topLeftRow": 876,
            "width": 370,
            "height": 92
          },
          "occlusion": 0.0016583027,
          "unreadable": 0.00016266086,
          "clusterScore": 1.0,
          "type": {
            "value": "N",
            "score": 0.99993
          },
          "text": {
            "value": "CV50070",
            "score": 0.997653
          },
          "characterConfidences": [
            0.99997425,
            0.9999752,
            0.99998605,
            0.99997497,
            0.9999715,
            0.9999931,
            0.99777764
          ],
          "dimension": {
            "width": 520,
            "height": 110,
            "score": 0.9999969
          }
        }
      ],
      "windshield": {
        "position": {
          "topLeftCol": 430.41376,
          "topLeftRow": 357.83295,
          "topRightCol": 1162.1718,
          "topRightRow": 332.93716,
          "bottomRightCol": 1300.4126,
          "bottomRightRow": 532.841,
          "bottomLeftCol": 390.6499,
          "bottomLeftRow": 573.5554,
          "score": 0.9342527
        },
        "segmentation": {
          "image": {
            "base64": "iVBORw0KGgoAAAANSUhEUgAAAEwAAAAfCAAAAABo5n...",
            "format": "png"
          },
          "topLeftCol": 390,
          "topLeftRow": 332,
          "width": 909,
          "height": 240
        },
        "occlusion": 0.0006347525,
        "clusterScore": 1.0
      },
      "mmr": {
        "view": {
          "value": "frontal",
          "id": 1,
          "score": 0.99802494
        },
        "view8": {
          "value": "frontal+right",
          "id": 8,
```

```
            "score": 0.9983039
          },
          "category": {
            "value": "CAR",
            "id": 2,
            "score": 0.9995504
          },
          "make": {
            "value": "VW",
            "id": 43,
            "score": 0.99904984
          },
          "model": {
            "value": "Passat",
            "id": 3569,
            "score": 0.99932593
          },
          "generation": {
            "value": "Mk VI (2005)",
            "id": 3691,
            "score": 0.9995584
          },
          "color": {
            "value": "WHITE",
            "id": 13,
            "score": 0.70475423
          },
          "tags": [
            {
              "name": "ambulance",
              "value": "no",
              "id": 14,
              "score": 0.9471349
            },
            {
              "name": "axle_count",
              "value": "2",
              "id": 23,
              "score": 0.939178
            },
            {
              "name": "caravan",
              "value": "no",
              "id": 10,
              "score": 0.9961877
            },
            {
              "name": "fire_brigade",
              "value": "no",
              "id": 16,
              "score": 0.9821997
            },
            {
              "name": "law_enforcement",
              "value": "yes",
              "id": 11,
              "score": 0.98277104
            },
            {
              "name": "pickup",
              "value": "no",
              "id": 38,
              "score": 0.9888988
            },
            {
              "name": "push_bumper",
              "value": "no",
```

```
                "id": 30,
                "score": 0.9862957
              },
              {
                "name": "roof_cargo",
                "value": "no",
                "id": 115,
                "score": 0.88583523
              },
              {
                "name": "taxi",
                "value": "no",
                "id": 18,
                "score": 0.64776176
              },
              {
                "name": "towed",
                "value": "no",
                "id": 51,
                "score": 0.99718744
              }
            ],
            "input": {
              "box": {
                "topLeftCol": 156.95068,
                "topLeftRow": 275.35706,
                "bottomRightCol": 1555.0796,
                "bottomRightRow": 1220.435
              }
            }
          }
        }
      ]
    }
  ],
  "engines": [
    {
      "task": "DETECTION",
      "moduleId": 802,
      "moduleVersion": "802-generic.gen-none-v7.16"
    },
    {
      "task": "OCR",
      "moduleId": 801,
      "moduleVersion": "801-generic.gen-gen-v7.20"
    },
    {
      "task": "MMR",
      "moduleName": "MMRBOX_VCMMGVCT_ACCE_2026Q2.dat",
      "moduleVersion": "20260522"
    }
  ]
}
```


Notes:

If the request contains **combinations** (manual detections), the corresponding elements with the input values are projected into the **combinations** response element (in the same order). However, the **position** element is copied from the request to the response only if its encapsulating object contains some information added by the analysis. This means that a manually detected **plate** will be returned as an empty element if it is not processed by the OCR; a manually detected **box** will never be returned (the MMR result is returned in the **mmr** element of the corresponding **roadUser**).

The **engines** element lists only SDK engines, which were used to obtain the results. For example, if the request contains **combinations** (manual detections), there is no "DETECTION" **task** in the **engines** element.

The **segmentation** elements can be returned only if this feature is supported by the detector (currently only module 802 if explicitly enabled before the server is started). The **base64** values ending with "..." in the example above are truncated.

Response Definitions:

The following table describes each item in the response.

Response item	Description	Data type
combinations	An array containing the detection and recognition results.	Array of objects
combinations/roadUsers	An array containing clusters of detected objects belonging to a single road user and recognition results.	Array of objects
combinations/roadUsers/box	Information about the detected vehicle bounding box.	Object
combinations/roadUsers/box/position	Information about the detected vehicle bounding box location in the input image.	Object
combinations/roadUsers/box/position/topLeftCol	The horizontal coordinate of the top-left bounding box corner within the image.	Decimal number
combinations/roadUsers/box/position/topLeftRow	The vertical coordinate of the top-left bounding box corner within the image.	Decimal number
combinations/roadUsers/box/position/bottomRightCol	The horizontal coordinate of the bottom-right bounding box corner within the image.	Decimal number
combinations/roadUsers/box/position/bottomRightRow	The vertical coordinate of the bottom-right bounding box corner within the image.	Decimal number
combinations/roadUsers/box/position/score	The vehicle detection confidence factor. Range 0 – 1.	Decimal number
combinations/roadUsers/box/segmentation	Information about the detected vehicle segmentation.	Object
combinations/roadUsers/box/segmentation/image	Segmentation image data.	Object
combinations/roadUsers/box/segmentation/image/base64	Base64-encoded segmentation image.	String
combinations/roadUsers/box/segmentation/image/format	Format of the segmentation image (e.g. "png").	String
combinations/roadUsers/box/segmentation/topLeftCol	The horizontal coordinate of the top-left corner of the segmentation image within the input image.	Integer
combinations/roadUsers/box/segmentation/topLeftRow	The vertical coordinate of the top-left corner of the segmentation image within the input image.	Integer
combinations/roadUsers/box/segmentation/width	Target width of the segmentation image in the input image. If the segmentation image width differs from this width, the segmentation image must be resized by the user.	Integer

combinations/roadUsers/box/segmentation/**height**	Target height of the segmentation image in the input image. If the segmentation image height differs from this height, the segmentation image must be resized by the user.	Integer
combinations/roadUsers/box/**occlusion**	Specifies how much the detected box is occluded. Range 0 (not occluded) – 1 (fully occluded).	Decimal number
combinations/roadUsers/**plates**	Information about the detected plates.	Array of objects
combinations/roadUsers/plates/**position**	Information about the detected plate location in the input image.	Object
combinations/roadUsers/plates/position/**topLeftCol**	The horizontal coordinate of the top-left plate corner within the image.	Decimal number
combinations/roadUsers/plates/position/**topLeftRow**	The vertical coordinate of the top-left plate corner within the image.	Decimal number
combinations/roadUsers/plates/position/**topRightCol**	The horizontal coordinate of the top-right plate corner within the image.	Decimal number
combinations/roadUsers/plates/position/**topRightCol**	The vertical coordinate of the top-right plate corner within the image.	Decimal number
combinations/roadUsers/plates/position/**bottomRightCol**	The horizontal coordinate of the bottom-right plate corner within the image.	Decimal number
combinations/roadUsers/plates/position/**bottomRightCol**	The vertical coordinate of the bottom-right plate corner within the image.	Decimal number
combinations/roadUsers/plates/position/**bottomLeftCol**	The horizontal coordinate of the bottom-left plate corner within the image.	Decimal number
combinations/roadUsers/plates/position/**bottomLeftRow**	The vertical coordinate of the bottom-left plate corner within the image.	Decimal number
combinations/roadUsers/plates/position/**score**	The plate detection confidence factor. Range 0 – 1.	Decimal number
combinations/roadUsers/plates/**segmentation**	Information about the detected plate segmentation.	Object
combinations/roadUsers/plates/segmentation/**image**	Segmentation image data.	Object
combinations/roadUsers/plates/segmentation/image/**base64**	Base64-encoded segmentation image.	String
combinations/roadUsers/plates/segmentation/image/**format**	Format of the segmentation image (e.g. "png").	String
combinations/roadUsers/plates/segmentation/**topLeftCol**	The horizontal coordinate of the top-left corner of the segmentation image within the input image.	Integer
combinations/roadUsers/plates/segmentation/**topLeftRow**	The vertical coordinate of the top-left corner of the segmentation image within the input image.	Integer
combinations/roadUsers/plates/segmentation/**width**	Target width of the segmentation image in the input image. If the segmentation image width differs from this width, the segmentation image must be resized by the user.	Integer

combinations/roadUsers/plates/segmentation/**height**	Target height of the segmentation image in the input image. If the segmentation image height differs from this height, the segmentation image must be resized by the user.	Integer
combinations/roadUsers/plates/**occlusion**	Specifies how much the detected plate is occluded. Range 0 (not occluded) – 1 (fully occluded).	Decimal number
combinations/roadUsers/plates/**unreadable**	Indicates the degree to which the plate is unreadable. Range 0 (readable) – 1 (unreadable).	Decimal number
combinations/roadUsers/plates/**clusterScore**	Confidence factor that the plate belongs to the road user. Range 0 – 1.	Decimal number
combinations/roadUsers/plates/**type**	Information about the recognized plate type.	Object
combinations/roadUsers/plates/type/**value**	The international country code for license plates; "ADR" or "TRASH" for ADR plates; "TRAILER" with the international country code for license plates on trailers. If the value is "UNK", then it was recognized as a false positive detection.	String
combinations/roadUsers/plates/type/**score**	The confidence factor for the type prediction. Range 0 – 1.	Decimal number
combinations/roadUsers/plates/**text**	Information about the recognized plate text.	Object
combinations/roadUsers/plates/text/**value**	The plate text recognized by the OCR. In case of multi-line text, the lines are separated by a semicolon (";").	String
combinations/roadUsers/plates/text/**score**	The confidence factor for the text prediction. Range 0 – 1.	Decimal number
combinations/roadUsers/plates/**characterConfidences**	Per-character confidence factors for the recognized plate text including separators ("-", ";"), in character order. Range 0 – 1. May not be supported by certain LPM modules.	Array of decimal numbers
combinations/roadUsers/plates/**dimension**	The predicted physical dimensions of the detected plate.	Object
combinations/roadUsers/plates/dimension/**width**	The predicted physical width of the detected plate in millimeters.	Integer
combinations/roadUsers/plates/dimension/**height**	The predicted physical height of the detected plate in millimeters.	Integer
combinations/roadUsers/plates/dimension/**score**	Confidence factor for the dimensions prediction. Range 0 – 1.	Decimal number
combinations/roadUsers/**windshield**	Information about the detected windshield.	Object
combinations/roadUsers/windshield/**position**	Information about the detected windshield location in the input image.	Object
combinations/roadUsers/windshield/position/**topLeftCol**	The horizontal coordinate of the top-left windshield corner within the image.	Decimal number
combinations/roadUsers/windshield/position/**topLeftRow**	The vertical coordinate of the top-left windshield corner within the image.	Decimal number
combinations/roadUsers/windshield/position/**topRightCol**	The horizontal coordinate of the top-right windshield corner within the image.	Decimal number

combinations/roadUsers/windshield/position/topRightCol	The vertical coordinate of the top-right windshield corner within the image.	Decimal number
combinations/roadUsers/windshield/position/bottomRightCol	The horizontal coordinate of the bottom-right windshield corner within the image.	Decimal number
combinations/roadUsers/windshield/position/bottomRightCol	The vertical coordinate of the bottom-right windshield corner within the image.	Decimal number
combinations/roadUsers/windshield/position/bottomLeftCol	The horizontal coordinate of the bottom-left windshield corner within the image.	Decimal number
combinations/roadUsers/windshield/position/bottomLeftRow	The vertical coordinate of the bottom-left windshield corner within the image.	Decimal number
combinations/roadUsers/windshield/position/score	The windshield detection confidence factor. Range 0 – 1.	Decimal number
combinations/roadUsers/windshield/segmentation	Information about the detected windshield segmentation.	Object
combinations/roadUsers/windshield/segmentation/image	Segmentation image data.	Object
combinations/roadUsers/windshield/segmentation/image/base64	Base64-encoded segmentation image.	String
combinations/roadUsers/windshield/segmentation/image/format	Format of the segmentation image (e.g. "png").	String
combinations/roadUsers/windshield/segmentation/topLeftCol	The horizontal coordinate of the top-left corner of the segmentation image within the input image.	Integer
combinations/roadUsers/windshield/segmentation/topLeftRow	The vertical coordinate of the top-left corner of the segmentation image within the input image.	Integer
combinations/roadUsers/windshield/segmentation/width	Target width of the segmentation image in the input image. If the segmentation image width differs from this width, the segmentation image must be resized by the user.	Integer
combinations/roadUsers/windshield/segmentation/height	Target height of the segmentation image in the input image. If the segmentation image height differs from this height, the segmentation image must be resized by the user.	Integer
combinations/roadUsers/windshield/occlusion	Specifies how much the detected windshield is occluded. Range 0 (not occluded) – 1 (fully occluded).	Decimal number
combinations/roadUsers/windshield/clusterScore	Confidence factor that the windshield belongs to the road user. Range 0 – 1.	Decimal number
combinations/roadUsers/wheels	Information about the detected wheels.	Array of objects
combinations/roadUsers/wheels/position	Information about the detected wheel location in the input image.	Object
combinations/roadUsers/wheels/position/topLeftCol	The horizontal coordinate of the top-left wheel corner within the image.	Decimal number
combinations/roadUsers/wheels/position/topLeftRow	The vertical coordinate of the top-left wheel corner within the image.	Decimal number

combinations/roadUsers/wheels/position/**bottomRightCol**	The horizontal coordinate of the bottom-right wheel corner within the image.	Decimal number
combinations/roadUsers/wheels/position/**bottomRightCol**	The vertical coordinate of the bottom-right wheel corner within the image.	Decimal number
combinations/roadUsers/wheels/position/**score**	The wheel detection confidence factor. Range 0 – 1.	Decimal number
combinations/roadUsers/wheels/**segmentation**	Information about the detected wheel segmentation.	Object
combinations/roadUsers/wheels/segmentation/**image**	Segmentation image data.	Object
combinations/roadUsers/wheels/segmentation/image/**base64**	Base64-encoded segmentation image.	String
combinations/roadUsers/wheels/segmentation/image/**format**	Format of the segmentation image (e.g. "png").	String
combinations/roadUsers/wheels/segmentation/**topLeftCol**	The horizontal coordinate of the top-left corner of the segmentation image within the input image.	Integer
combinations/roadUsers/wheels/segmentation/**topLeftRow**	The vertical coordinate of the top-left corner of the segmentation image within the input image.	Integer
combinations/roadUsers/wheels/segmentation/**width**	Target width of the segmentation image in the input image. If the segmentation image width differs from this width, the segmentation image must be resized by the user.	Integer
combinations/roadUsers/wheels/segmentation/**height**	Target height of the segmentation image in the input image. If the segmentation image height differs from this height, the segmentation image must be resized by the user.	Integer
combinations/roadUsers/wheels/**occlusion**	Specifies how much the detected wheel is occluded. Range 0 (not occluded) – 1 (fully occluded).	Decimal number
combinations/roadUsers/wheels/**clusterScore**	Confidence factor that the wheel belongs to the road user. Range 0 – 1.	Decimal number
combinations/roadUsers/**mmr**	Information about the recognized road user's attributes.	Object
combinations/roadUsers/mmr/**view**	The recognized road user's positional information.	Object
combinations/roadUsers/mmr/view/**value**	The value of the road user's view, either "frontal", "side" or "rear".	String
combinations/roadUsers/mmr/view/**id**	ID of the recognized road user's view.	Integer
combinations/roadUsers/mmr/view/**score**	The confidence factor for the view result. Range 0 – 1.	Decimal number
combinations/roadUsers/mmr/**view8**	The recognized road user's positional information with increased specificity. Only available for Box-based MMR.	Object
combinations/roadUsers/mmr/view8/**value**	The value of the road user's view8. Possible values: "frontal exact", "frontal+left", "frontal+right", "left", "right", "rear exact", "rear+left", "rear+right".	String

combinations/roadUsers/mmr/view8/**id**	ID of the recognized road user's view8.	Integer
combinations/roadUsers/mmr/view8/**score**	The confidence factor for the view8 result. Range 0 – 1.	Decimal number
combinations/roadUsers/mmr/**category**	The recognized road user's category.	Object
combinations/roadUsers/mmr/category/**value**	The value of the road user's category, e.g., "BUS", "CAR", "HVT", ... For the full list of possible categories and their definition, check the Eyedea MMR SDK documentation.	String
combinations/roadUsers/mmr/category/**id**	ID of the recognized road user's category.	Integer
combinations/roadUsers/mmr/category/**score**	The confidence factor for the category result. Range 0 – 1.	Decimal number
combinations/roadUsers/mmr/**make**	The recognized vehicle manufacturer. Only available for vehicles.	Object
combinations/roadUsers/mmr/make/**value**	The recognized vehicle manufacturer, e.g., "VW", "Ford", "Fiat", ... For the full list of possible makes, check the Eyedea MMR SDK documentation.	String
combinations/roadUsers/mmr/make/**id**	ID of the recognized vehicle manufacturer.	Integer
combinations/roadUsers/mmr/make/**score**	The confidence factor for the make result. Range 0 – 1.	Decimal number
combinations/roadUsers/mmr/**model**	The recognized vehicle model. Only available for vehicles.	Object
combinations/roadUsers/mmr/model/**value**	The recognized vehicle model (vehicle instance defined by a bodywork), e.g., "Golf", "Mondeo", "500", ...	String
combinations/roadUsers/mmr/model/**id**	ID of the recognized vehicle model.	Integer
combinations/roadUsers/mmr/model/**score**	The confidence factor for the model result. Range 0 – 1.	Decimal number
combinations/roadUsers/mmr/**generation**	The recognized vehicle generation. Only available for vehicles.	Object
combinations/roadUsers/mmr/generation/**value**	The recognized vehicle generation (vehicle mark and first model year), e.g., "Mk VI (2019)", "Mk I (2020)", ...	String
combinations/roadUsers/mmr/generation/**id**	ID of the recognized vehicle generation.	Integer
combinations/roadUsers/mmr/generation/**score**	The confidence factor for the generation result. Range 0 – 1.	Decimal number
combinations/roadUsers/mmr/**variation**	The recognized vehicle variation. Only available for vehicles.	Object
combinations/roadUsers/mmr/variation/**value**	The recognized vehicle variation (vehicle trim level and/or body type), e.g., "AMG", "AMG-Line SUV", "Coupe", ...	String

combinations/roadUsers/mmr/variation/**id**	ID of the recognized vehicle variation.	Integer
combinations/roadUsers/mmr/variation/**score**	The confidence factor for the variation result. Range 0 – 1.	Decimal number
combinations/roadUsers/mmr/**color**	The recognized vehicle color. Only available for selected categories.	Object
combinations/roadUsers/mmr/color/**value**	The recognized vehicle color, e.g., "BLUE", "GRAY", "RED", ...	String
combinations/roadUsers/mmr/color/**id**	ID of the recognized vehicle color.	Integer
combinations/roadUsers/mmr/color/**score**	The confidence factor for the color result. Range 0 – 1.	Decimal number
combinations/roadUsers/mmr/**tags**	An array of recognized road user's (usually vehicle) traits. A tag may typically recognize whether a vehicle belongs to a certain group. Some tag tasks can only be recognized from a certain view.	Array of objects
combinations/roadUsers/mmr/tags/**name**	The name of the trait, e.g., "caravan", "ambulance", ... For the full list of possible tags and their definition, check the Eyedea MMR SDK documentation.	String
combinations/roadUsers/mmr/tags/**value**	The value of the trait. Either "yes", or "no".	String
combinations/roadUsers/mmr/tags/**id**	ID of the trait value.	Integer
combinations/roadUsers/mmr/tags/**score**	The confidence factor for the tag value. Range 0 – 1.	Decimal number
combinations/roadUsers/mmr/**input**	The source object within the cluster used for the MMR computation. Contains either box, or plate object.	Object
combinations/roadUsers/mmr/input/**box**	The parameters of the bounding box used for the MMR computation.	Object
combinations/roadUsers/mmr/input/box/**topLeftCol**	The horizontal coordinate of the top-left bounding box corner within the image.	Decimal number
combinations/roadUsers/mmr/input/box/**topLeftRow**	The vertical coordinate of the top-left bounding box corner within the image.	Decimal number
combinations/roadUsers/mmr/input/box/**bottomRightCol**	The horizontal coordinate of the bottom-right bounding box corner within the image.	Decimal number
combinations/roadUsers/mmr/input/box/**bottomRightRow**	The vertical coordinate of the bottom-right bounding box corner within the image.	Decimal number
combinations/roadUsers/mmr/input/**plate**	The parameters of the plate used for the MMR computation.	Object
combinations/roadUsers/mmr/input/plate/**index**	The 0-based index of detected plate used for the MMR computation.	Integer
combinations/roadUsers/mmr/input/plate/**centerCol**	The horizontal coordinate of the plate center within the image.	Decimal number

combinations/roadUsers/mmr/input/plate/centerRow	The vertical coordinate of the plate center within the image.	Decimal number
combinations/roadUsers/mmr/input/plate/rotation	Clockwise correction of the source image in degrees.	Decimal number
combinations/roadUsers/mmr/input/plate/pixelsPerMeter	The real-world scale in pixels per meter of the image computed on the plate. For example, plate of width = 300 millimeters, photographed to have 150 pixels, has 150 / 0.3 = 500 pixels per meter scale.	Decimal number
engines	An array containing the detection and recognition results.	Array of objects
engines/task	The computation task processed by the SDK engine. Either "DETECTION", "OCR", or "MMR".	String
engines/moduleId	SDK engine module ID.	Integer
engines/moduleName	SDK engine module name.	String
engines/moduleVersion	SDK engine module version.	String