



MMR+ANPR REST Server 3.0.2

User Guide

Copyright © 2024, 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.

Gemalto, the Gemalto logo, are trademarks and service marks of Gemalto and are registered in certain countries. Safenet, Sentinel, Sentinel Local License Manager and Sentinel Hardware Key are registered trademarks of Safenet, Inc.

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 | 4-11 |
| 4.1 GPU Support Prerequisites | 4-11 |
| 4.2 Sentinel license protection system | 4-11 |
| 4.3 Building the Docker image | 4-13 |
| 4.4 Running the Docker image | 4-14 |
| 4.5 Log files | 4-16 |
| 5 REST API Documentation..... | 5-18 |
| 5.1 SDK Engine Overview..... | 5-18 |
| 5.2 Response Status Codes..... | 5-18 |
| 5.3 Entry point..... | 5-18 |
| 5.4 System Info | 5-19 |
| 5.5 Recognition | 5-24 |

1 Introduction

Eyedeia Recognition's MMR+ANPR REST Server is a Java server application with REST interface running on Tomcat Docker container which allows to detect vehicle license plates or vehicles in input images and recognize the text and type of detected license 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 models or 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, 2 cores (4 logical processors)
- RAM: 4 GB
- Hard disk: 256 GB (optional SSD)
- GPU (optional): NVIDIA Driver version \geq 410.48 compatible
- Operating system: Ubuntu 18.04 and higher – x86_64 platform

2.2 Recommended requirements

- Processor: Intel® Core™ i7, 4 cores (8 logical processors)
- RAM: 16 GB
- Hard disk: 512 GB, SSD
- GPU (optional): NVIDIA® GeForce® GTX 1050 Ti, 4GB GDDR5
- Operating system: Ubuntu 18.04 and higher – x86_64 platform

3 Version history

Version 3.0.2

Released: 2024/01/18

- Updated LPM modules
- Used LPM module: LPM-v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
- Used MMR module: Eyedea-MMR-2.22.0-Ubuntu-18.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-9.12.1
- Embedded modules with GPU support are using CUDA 10.0.
(*CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48*)

Version 3.0.1

Released: 2023/12/07

- Fixed detection angle: license plate detector now returns opposite values to the previous versions; also fixes input for MMR License Plate based
- Updated LPM module: LPM-v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
- Updated MMR module: Eyedea-MMR-2.22.0-Ubuntu-18.04-x86_64-HASP
- Used Sentinel license protection system: aksusbd_108842-9.12.1
- Embedded modules with GPU support are using CUDA 10.0.
(*CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48*)

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
- Embedded modules with GPU support are using CUDA 10.0.
(*CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48*)

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
- Embedded modules with GPU support are using CUDA 10.0.

(CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

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
- Embedded modules with GPU support are using CUDA 10.0.
(CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

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
- Embedded modules with GPU support are using CUDA 10.0.
(CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

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
- Embedded modules with GPU support are using CUDA 10.0.
(CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

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
- Embedded modules with GPU support are using CUDA 10.0.
(CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48)

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
- Embedded modules with GPU support are using CUDA 10.0.
(*CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48*)

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
- Embedded modules with GPU support are using CUDA 10.0.
(*CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48*)

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
- Embedded modules with GPU support are using CUDA 10.0.
(*CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48*)

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
- Embedded modules with GPU support are using CUDA 10.0.
(*CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48*)

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
- Embedded modules with GPU support are using CUDA 10.0.
(*CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48*)

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
- Embedded modules with GPU support are using CUDA 10.0.
(*CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48*)

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
- Embedded modules with GPU support are using CUDA 10.0.
(*CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48*)

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
- Embedded modules with GPU support are using CUDA 10.0.
(*CUDA 10.0 requires the Linux x86_64 Driver version >= 410.48*)

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
- Embedded modules with GPU support are using CUDA 9.1.85.
(*CUDA 9.1.85 requires the Linux x86_64 Driver version >= 390.46*)

Version 1.1.1

Released: 2019/07/01

- Added option to run the server with ANPR / MMR modules disabled.
- HTML documentation updated.
- Embedded modules with GPU support are using CUDA 9.1.85.
(*CUDA 9.1.85 requires the Linux x86_64 Driver version >= 390.46*)

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
- Embedded modules with GPU support are using CUDA 9.1.85.
(*CUDA 9.1.85 requires the Linux x86_64 Driver version >= 390.46*)

Version 1.1.0-BETA

Released: 2019/01/11

- Added support for descriptor batch computation.
- Added support for GPU computation (ANPR, MMR).
- Embedded modules with GPU support are using CUDA 9.1.85.
(*CUDA 9.1.85 requires the Linux x86_64 Driver version >= 390.46*)

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 Building the Docker image).

4. Install nvidia-container-runtime library

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

```
./installNvidiaContainerRuntime
```

4.2 Sentinel license protection system

Installation

The SDK engines used by MMR+ANPR REST Server are protected with a standard third-party software licensing solution, *Sentinel LDK* by *Gemalto*.

1. Install the 32-bit compatibility binaries

As a root, execute the following command to install the 32-bit compatibility binaries:

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

(Without compatibility binaries error “No such file or directory.” might appear.)

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

Uncompress the aksusbd*.tar.gz file.

3. Uninstall prior Sentinel LDK Run-time Environment version

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

```
./dunst
```

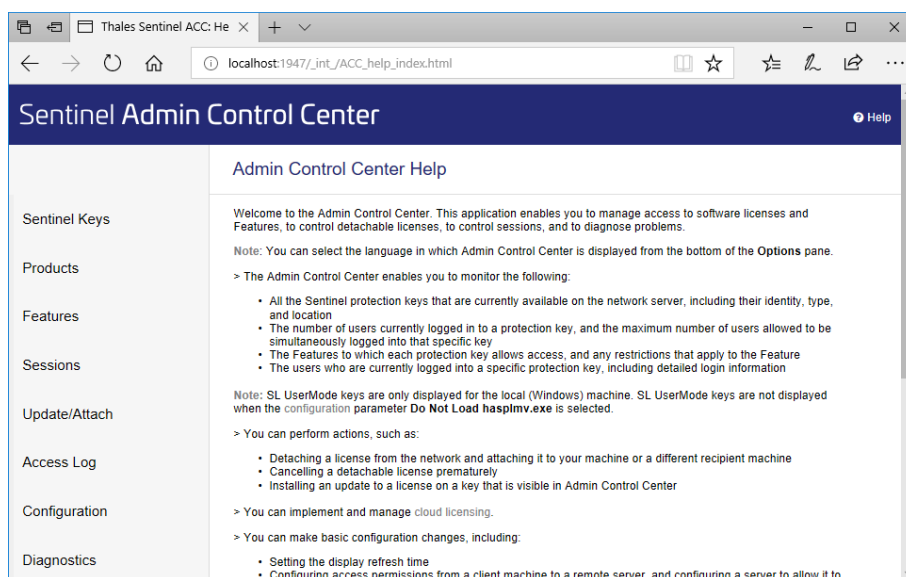
4. Install the Sentinel LDK Run-time Environment

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

```
./dinst
```

Verification

After the successful Sentinel License Server installation, open the address <http://localhost:1947> in a web browser and check whether the license server is running. If the Sentinel Admin Control Center web application is displayed, the license server is running. License server can be configured to allow to connect clients to use available licenses (configured as a server) or it can connect to another license server (configured as a client).



Server configuration

To set the license server as a license provider, please open the address in your web browser http://localhost:1947/int/_/config_from.html and choose the appropriate option from Allow Access from Remote Clients to allow other clients to connect to the network license key plugged in the server.

| Basic Settings | Users | Access to Remote License Managers | Access from Remote Clients | Client Identities | Detachable Licenses | Network |
|--|-------|-----------------------------------|----------------------------|-------------------|---------------------|---------|
| <p>Allow Access from Remote Clients</p> <p> <input type="radio"/> No one <input type="radio"/> Identifiable clients only. Non-cloud licenses cannot be accessed. <input type="radio"/> Cloud licenses require identity. Other licenses are accessible by all clients. <input checked="" type="radio"/> All licenses are accessible without need of identity </p> <p>Note: Regardless of the option selected, remote machines using a client identity cannot access non-cloud licenses.</p> | | | | | | |

Client configuration

To set the license server as a client, open the address http://localhost:1947/int/_/config_to.html in a web browser and choose the Allow Access to Remote Licenses option. If the license provider is in another network than the computer, put the server's IP address into the field Remote License Search Parameters.

| Basic Settings | Users | Access to Remote License Managers | Access from Remote Clients | Client Identities | Detachable Licenses | Network |
|---------------------------------------|-------------------------------------|--|----------------------------|-------------------|---------------------|---------|
| Allow Access to Remote Licenses | <input checked="" type="checkbox"/> | You may experience a delay of a few minutes before your changes take effect. | | | | |
| Broadcast Search for Remote Licenses | <input checked="" type="checkbox"/> | | | | | |
| Aggressive Search for Remote Licenses | <input type="checkbox"/> | | | | | |
| Remote License Search Parameters | <input type="text"/> | | | | | |

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=v7.6-2023-11-10-Ubuntu-18.04-hasp9.0
```

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

```
ENV MMR_VERSION=2.22.0-Ubuntu-18.04-x86_64-HASP
```

2. 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 image

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

The application settings (license server address, number of threads, ...) are loaded from the system environment variables by the application. The supported variables are defined in the env-GPU.list and env-CPU.list files (the first one is intended to use the GPU for computation, the other uses the CPU). One of these files is passed to the Docker during the image initialization and its variables are available in the Docker container as system environment variables.

In the following overview, <ENGINE> stands for: LPM_LP_DETECTOR, LPM_LP_OCR, MMR_LP, LPM_BOX_DETECTOR and MMR_BOX.

| | |
|---------------------------|--|
| HASP_REMOTE_SERVERADDR | The address of the Sentinel License Manager server with the valid license. If the licenses are on the Docker host, you can usually set the IP address to 10.0.75.1 or 172.17.0.1. |
| LPM_LP_MODULE_ID | The three-digit identifiers of LPM modules used for the license plate detection + OCR and the box detection, respectively. The appropriate *.tar.gz archives must be in the [PACKAGE]/LPM-modules directory. Currently, only LPM module 802 supports the box detection. |
| LPM_BOX_MODULE_ID | |
| MMR_LP_MODEL | Specify the binary models used for license plate based (MMR_*) and box based (MMRBOX_*) MMR, respectively. The default MMR models are *_VCMGVCT_* 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 precise ones), select the appropriate data models from the [PACKAGE]/Eyedea-MMR-\${MMR_VERSION}.tar.gz/\${MMR_PATH}/model directory. See the MMR documentation for details. |
| MMR_BOX_MODEL | |
| <ENGINE>_NUMTHREADS | The number of engine's threads to initialize. Set the variable to 0 to disable the given engine. Set -1 to select the number of threads automatically (1 for CPU, or 1 for each <ENGINE>_GPU_ID in case of GPU computation mode). Note: When running on CPU, the number of LPM Detector threads is limited to 1 because it internally uses the optimal number of CPU threads. |
| <ENGINE>_COMPUTATION_MODE | Specifies whether the given engine runs on CPU, or GPU. (Certain older LPM Detectors may support only CPU.) |
| <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_LP_OCR_GPU_ID=0 MMR_LP_GPU_ID=1). |
| LOG_STATS_PERIOD_REQUEST | Specifies whether and, if necessary, with what period recognition statistics should be logged to a file. Set a positive integer <i>N</i> to log recognition statistics to a new file every <i>N</i> -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_LP_MODULE_ID=800
LPM_LP_DETECTOR_NUMTHREADS=1
LPM_LP_DETECTOR_COMPUTATION_MODE=GPU
LPM_LP_DETECTOR_GPU_ID=0
LPM_LP_OCR_NUMTHREADS=1
LPM_LP_OCR_COMPUTATION_MODE=GPU
LPM_LP_OCR_GPU_ID=0
LPM_BOX_DETECTOR_NUMTHREADS=0
MMR_LP_MODEL=MMR_VCCT_FAST_2023Q4.dat
MMR_LP_NUMTHREADS=1
MMR_LP_COMPUTATION_MODE=GPU
MMR_LP_GPU_ID=0
MMR_BOX_NUMTHREADS=0
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 three running SDK engines: LPM License Plate Detector, LPM OCR and License Plate based MMR, each of them running in one thread on the GPU. The Box engines will be disabled.

LPM License Plate Detector + OCR will use the European module 800; the MMR will use the fast model to recognize only view, category, color and tags.

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.

2. Run the image (runDockerGPU or runDockerCPU)

Use the runDockerGPU or runDockerCPU script located in the [PACKAGE]/ directory to run the built Docker image with / without the support for GPU computation. Edit these scripts if needed.

runDockerGPU – support for GPU computation:

```
docker run --gpus device=0 --env-file env-GPU.list -p 8080:8080 mmr-anpr-rest-server
```

runDockerCPU – CPU computation only:

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

Notes:

The --env-file option specifies the file with the environment variables.

The -p option publishes port 8080 in the container and maps it to the host's port 8080.

The --gpus option specifies GPU device(s) used by the container. Possibilities:

- Use a GPU device using its index: --gpus device=0
- Use multiple GPU devices using their indexes: --gpus "device=2,3"
- Use all available GPU devices: --gpus all

The server tries to initialize all engine instances specified in the env-*.list file. 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 thread was requested for the given engine, "FAILED TO START" means that all requested threads of the given engine could not be started; in both cases, the server handles requests and returns data obtained by other running engines.

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 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 env-*.list file.

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 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 run the Docker container with a [volume](#) to store log files on the host, add `-v $PWD/logs:/usr/local/tomcat/webapps/RESTServer_data/logs` to the `runDocker*` script.

The time zone used for date and time associated with log files depends on the Docker container locale, which is UTC by default. To use the host time zone instead, add `-v /etc/timezone:/etc/timezone:ro` to the `runDocker*` script.

5 REST API Documentation

5.1 SDK Engine Overview

Which SDK engines do you need? Check the following table.

| Use Case | SDK Engines |
|---|--|
| License plate detection | LPM License Plate Detector |
| License plate detection + OCR | LPM License Plate Detector, LPM OCR |
| License plate detection + OCR + Vehicle recognition | LPM License Plate Detector, LPM OCR, MMR License Plate based |
| Vehicle detection | LPM Box Detector |
| Vehicle detection + Vehicle recognition | LPM Box Detector, MMR Box based |
| Vehicle recognition (without detection; <i>detections must be provided in the request</i>) | MMR License Plate based, <i>or</i> MMR Box based |

Note: License plate detection + OCR = ANPR

LPM License Plate Detector detects license plates.

LPM OCR recognizes the text and type of license plates detected by LPM License Plate Detector.

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 License Plate Detector and LPM OCR are needed, or the license plate detections must be provided in the request.

LPM Box Detector detects vehicles (and some other "road users" like pedestrians, kickbikes, etc.).

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 Box Detector is needed, or the box detections must be provided in the request.

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. |


5.3 Entry point

`SERVER_IP:8080/RESTServer`

5.4 System Info

System info contains information about system resources and SDK engines.

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: | <pre>curl -X GET -H "Accept: text/html" http://localhost:8080/RESTServer/</pre> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Response Example: |  <p>The screenshot displays the 'MMR+ANPR REST Server' monitoring interface. At the top, it shows the title 'MMR+ANPR REST Server' and the last info update time '2023-06-27 11:34:23'. Below this, there are two tabs: 'System Info' and 'Recognition'. The 'System Info' tab is active, showing a system load of 7.4% and memory usage of 59.1% / 18.5GB. There are two line graphs: one for system load (green) and one for memory usage (red). Below the graphs, there is a table of tasks:</p> <table border="1"> <thead> <tr> <th>Task Name</th> <th>Model</th> <th>Thread</th> <th>GPU</th> <th>Processed tasks</th> <th>Waiting tasks</th> </tr> </thead> <tbody> <tr> <td>LPM License Plate Detector</td> <td>801-generic.gen-gen-v7.8</td> <td>1</td> <td>@ GPU</td> <td>5</td> <td>0</td> </tr> <tr> <td>LPM OCR</td> <td>801-generic.gen-gen-v7.8</td> <td>1</td> <td>@ GPU</td> <td>8</td> <td>0</td> </tr> <tr> <td>MMR License Plate based</td> <td>MMR_VCMMGVCT_PREC_2023Q2.dat</td> <td>1</td> <td>@ GPU</td> <td>8</td> <td>0</td> </tr> <tr> <td>LPM Box Detector</td> <td>802-generic.gen-none-v7.1</td> <td>1</td> <td>@ GPU</td> <td>7</td> <td>0</td> </tr> <tr> <td>MMR Box based</td> <td>MMRBOX_VCMMGVCT_PREC_2023Q2.dat</td> <td>1</td> <td>@ GPU</td> <td>20</td> <td>0</td> </tr> </tbody> </table> <p>At the bottom right, there is a 'GPU utilization' graph showing 0% usage for GPU0.</p> | Task Name | Model | Thread | GPU | Processed tasks | Waiting tasks | LPM License Plate Detector | 801-generic.gen-gen-v7.8 | 1 | @ GPU | 5 | 0 | LPM OCR | 801-generic.gen-gen-v7.8 | 1 | @ GPU | 8 | 0 | MMR License Plate based | MMR_VCMMGVCT_PREC_2023Q2.dat | 1 | @ GPU | 8 | 0 | LPM Box Detector | 802-generic.gen-none-v7.1 | 1 | @ GPU | 7 | 0 | MMR Box based | MMRBOX_VCMMGVCT_PREC_2023Q2.dat | 1 | @ GPU | 20 | 0 |
| Task Name | Model | Thread | GPU | Processed tasks | Waiting tasks | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| LPM License Plate Detector | 801-generic.gen-gen-v7.8 | 1 | @ GPU | 5 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| LPM OCR | 801-generic.gen-gen-v7.8 | 1 | @ GPU | 8 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| MMR License Plate based | MMR_VCMMGVCT_PREC_2023Q2.dat | 1 | @ GPU | 8 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| LPM Box Detector | 802-generic.gen-none-v7.1 | 1 | @ GPU | 7 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| MMR Box based | MMRBOX_VCMMGVCT_PREC_2023Q2.dat | 1 | @ GPU | 20 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

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: | |
| <code>curl -X GET -H "Accept: text/plain" http://localhost:8080/RETServer/</code> | |
| Response Example: | |
| <pre> System: - Average load: 9.93% - Used memory: 58.99% (18.48 GB) - Started on: 2023-06-27 09:26:44 GPU 0 (Graphics Device): - Utilization: 6% - Free memory: 3455 / 5905 MB Detection: - Engine: LPM License Plate Detector - Version: v7.4.1-2023-01-12-Ubuntu-18.04-hasp - Module ID: 801 - Module version: 801-generic.gen-gen-v7.8 - Computation mode: GPU - Status: RUNNING - Processing batch size: 1 - Number of running threads: 1 - Number of waiting tasks: 0 - Number of processed tasks: 5 Classification: - Engine: LPM OCR - Version: v7.4.1-2023-01-12-Ubuntu-18.04-hasp - Module ID: 801 - Module version: 801-generic.gen-gen-v7.8 - Computation mode: GPU - Status: RUNNING - Processing batch size: 1 - Number of running threads: 1 - Number of waiting tasks: 0 - Number of processed tasks: 8 Classification: - Engine: MMR License Plate based - Version: 2.21.0-Ubuntu-18.04-x86_64-HASP - Module: MMR_VCMMGVCT_PREC_2023Q2.dat - Module version: 20230512 - Computation mode: GPU - Status: RUNNING - Processing batch size: 1 - Number of running threads: 1 - Number of waiting tasks: 0 - Number of processed tasks: 8 Detection: - Engine: LPM Box Detector - Version: v7.4.1-2023-01-12-Ubuntu-18.04-hasp - Module ID: 802 - Module version: 802-generic.gen-none-v7.1 - Computation mode: GPU - Status: RUNNING - Processing batch size: 1 - Number of running threads: 1 </pre> | |

```

- Number of waiting tasks: 0
- Number of processed tasks: 7

Classification:
- Engine: MMR Box based
- Version: 2.21.0-Ubuntu-18.04-x86_64-HASP
- Module: MMRBOX_VCMMGVCT_PREC_2023Q2.dat
- Module version: 20230513
- Computation mode: GPU
- Status: RUNNING
- Processing batch size: 1
- Number of running threads: 1
- Number of waiting tasks: 0
- Number of processed tasks: 20

```

Get detailed server system info

| | |
|--|-------------------|
| Endpoint: | /serverSystemInfo |
| HTTP Method: | GET |
| Consumes Media Type: | - |
| Produces Media Type: | application/json |
| URL Parameters: | - |
| Data Parameters: | - |
| CURL Command Example: | |
| <pre>curl -X GET http://localhost:8080/RESTServer/serverSystemInfo</pre> | |
| Response Example: | |
| <pre>{ "timestamp": 1687858635246, "serverStartedTimestamp": 1687858004078, "systemLoadAverage": 11.363636363636363, "systemUsedMemory": 58.99442995234615, "systemUsedMemoryBytes": 19843907584, "processingServersInfosList": [{ "batchSize": 1, "computationMode": "GPU", "engineName": "LPM", "engineType": "License Plate Detector", "engineVersion": "v7.4.1-2023-01-12-Ubuntu-18.04-hasp", "moduleId": 801, "moduleVersion": "801-generic.gen-gen-v7.8", "numComputingThreads": 1, "numProcessedTasks": 5, "numWaitingTasks": 0, "status": "RUNNING" }, { "batchSize": 1, "computationMode": "GPU", "engineName": "LPM", "engineType": "OCR", "engineVersion": "v7.4.1-2023-01-12-Ubuntu-18.04-hasp", "moduleId": 801, "moduleVersion": "801-generic.gen-gen-v7.8", </pre> | |

```

    "numComputingThreads": 1,
    "numProcessedTasks": 8,
    "numWaitingTasks": 0,
    "status": "RUNNING"
  },
  {
    "batchSize": 1,
    "computationMode": "GPU",
    "engineName": "MMR",
    "engineType": "License Plate based",
    "engineVersion": "2.21.0-Ubuntu-18.04-x86_64-HASP",
    "moduleName": "MMR_VCMMGVCT_PREC_2023Q2.dat",
    "moduleVersion": "20230512",
    "numComputingThreads": 1,
    "numProcessedTasks": 8,
    "numWaitingTasks": 0,
    "status": "RUNNING"
  },
  {
    "batchSize": 1,
    "computationMode": "GPU",
    "engineName": "LPM",
    "engineType": "Box Detector",
    "engineVersion": "v7.4.1-2023-01-12-Ubuntu-18.04-hasp",
    "moduleId": 802,
    "moduleVersion": "802-generic.gen-none-v7.1",
    "numComputingThreads": 1,
    "numProcessedTasks": 7,
    "numWaitingTasks": 0,
    "status": "RUNNING"
  },
  {
    "batchSize": 1,
    "computationMode": "GPU",
    "engineName": "MMR",
    "engineType": "Box based",
    "engineVersion": "2.21.0-Ubuntu-18.04-x86_64-HASP",
    "moduleName": "MMRBOX_VCMMGVCT_PREC_2023Q2.dat",
    "moduleVersion": "20230513",
    "numComputingThreads": 1,
    "numProcessedTasks": 20,
    "numWaitingTasks": 0,
    "status": "RUNNING"
  }
],
"gpuInfoList": [
  {
    "id": 0,
    "name": "Graphics Device",
    "totalMemoryMB": 5905,
    "freeMemoryMB": 3455,
    "gpuUtilizationPerc": 6
  }
]
}

```

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 |

| | | |
|--|--|------------------|
| systemLoadAverage | Recent CPU usage for the whole system in percent. | Decimal number |
| systemUsedMemory | Amount of physical memory used by any process in percent. | Decimal number |
| systemUsedMemoryBytes | Amount of physical memory used by any process in bytes. | Integer |
| processingServersInfosList | Information about the SDK engines. | Array of objects |
| processingServersInfosList/ batchSize | Size of the computation batch. | Integer |
| processingServersInfosList/ computationMode | Type of processing unit used for computation. Either "CPU", or "GPU". | String |
| processingServersInfosList/ engineName | SDK engine name. | String |
| processingServersInfosList/ engineType | SDK engine type. | String |
| processingServersInfosList/ engineVersion | SDK engine version. | String |
| processingServersInfosList/ moduleId | DK engine module ID. | Integer |
| processingServersInfosList/ moduleName | SDK engine module name. | String |
| processingServersInfosList/ moduleVersion | SDK engine module version. | String |
| processingServersInfosList/ numComputingThreads | Number of initialized engine's threads. | Integer |
| processingServersInfosList/ numProcessedTasks | Number of processed computation tasks. | Integer |
| processingServersInfosList/ numWaitingTasks | Number of queued computation tasks to be processed. | Integer |
| processingServersInfosList/ status | SDK engine status. Either "RUNNING" (at least 1 thread successfully initialized), "DISABLED" (no thread requested), or "FAILED TO START" (all requested threads could not be started). | String |
| gpuInfoList | Information about available GPU devices. | Array of objects |
| gpuInfoList/id | GPU device ID. | Integer |
| gpuInfoList/name | GPU device name. | String |
| gpuInfoList/totalMemoryMB | Total amount of physical GPU device memory in megabytes. | Integer |

| | | |
|--------------------------------|---|---------|
| gpuInfoList/freeMemoryMB | Amount of free physical GPU device memory in megabytes. | Integer |
| gpuInfoList/gpuUtilizationPerc | Utilization of the GPU device in percent. | Integer |

5.5 Recognition

Recognition provides information about recognized license plates and corresponding vehicles in the input image.

Web page with image upload and recognition functionality using the license plate, or box detector

| | |
|------------------------------|---|
| Endpoint: | /recognition |
| HTTP Method: | GET |
| Consumes Media Type: | - |
| Produces Media Type: | text/html |
| URL Parameters: | - |
| Data Parameters: | - |
| CURL Command Example: | <pre>curl -X GET http://localhost:8080/RESTServer/recognition</pre> |

Response Examples:

Using license plate detector:

MMR+ANPR REST Server

Recognition



System Info
Recognition



CV50070

ANPR

| | |
|----------|------------------------------------|
| OCR | CV50070 |
| | score: 1.000 |
| Country | N |
| | score: 0.962 |
| Occluded | x |
| | occlusion: 0.001 |

MMR

| | |
|------------|--------------------------------|
| View | frontal |
| | score: 1.000 |
| Category | CAR |
| | score: 1.000 |
| Make | VW |
| | score: 1.000 |
| Model | Passat |
| | score: 1.000 |
| Generation | Mk VI (2005) |
| | score: 1.000 |
| Variation | - |
| | score: - |
| Color | <input type="checkbox"/> WHITE |
| | score: 0.997 |

License plate detector (ANPR + MMR)

Box detector (MMR only)

Browse... image_03.jpg

MMR tags

law_enforcement

x: ambulance caravan fire_brigade push_bumper

Using box detector:

MMR+ANPR REST Server
Recognition

System Info Recognition

ANPR

OCR -
score: --

Country -
score: --

Occluded -
occlusion: --

MMR

View frontal
ID: 1
score: 1.000

Category CAR
ID: 2
score: 1.000

Make VW
ID: 43
score: 1.000

Model Passat
ID: 3569
score: 0.999

Generation Mk VI (2005)
ID: 3691
score: 0.999

Variation -
ID: -
score: --

Color WHITE
ID: 13
score: 0.942

License plate detector (ANPR + MMR)
 Box detector (MMR only)

Browse... image_03.jpg

MMR tags

law_enforcement

ambulance caravan fire_brigade pickup push_bumper towed wood_truck

Notes:

License plate detector, LPM OCR and license plate based MMR engines must be running for both ANPR and MMR results to be displayed.

When the Box detector is used, only MMR results are displayed (for LPM box detector and box based MMR engines running).

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

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

Detect and read license plates in the input image and recognize the corresponding vehicles

| | |
|-----------------------------|--|
| Endpoint: | /lpRecognition |
| HTTP Method: | POST |
| Consumes Media Type: | multipart/form-data |
| Produces Media Type: | application/json |
| URL Parameters: | - |
| Data Parameters: | <p>file – form data parameter containing input image</p> <p>roi – (optional) form data parameter containing a JSON object that defines the area of the input image to be scanned by the detector</p> |

The region of interest parameter is defined by its corners coordinates: `topLeftX`, `topLeftY`, `topRightX`, `topRightY`, `bottomLeftX`, `bottomLeftY`, `bottomRightX` and `bottomRightY` (decimal numbers).

Example:

```
{
  "topLeftX": 176,
  "topLeftY": 362,
  "topRightX": 1234,
  "topRightY": 362,
  "bottomLeftX": 176,
  "bottomLeftY": 1000,
  "bottomRightX": 1234,
  "bottomRightY": 1000
}
```

For the sides of a region of interest parallel to the sides of the image, a simplified form of *roi* defined by its sides can be used instead: `leftX`, `topY`, `rightX` and `bottomY` (decimal numbers).

Example:

```
{
  "leftX": 176,
  "topY": 362,
  "rightX": 1234,
  "bottomY": 1000
}
```

`lpDetections` – (optional) form data parameter containing a JSON object with an array of license plate positions (for MMR computation only)

The license plate detections parameter is defined as an array of "center, angle, scalePPM" structures which define the position of a license plate in the image for MMR computation. If this parameter is used, ANPR (automatic detection and OCR) will not run during the input image processing.

`center` – 2D point (structure of `x` and `y` decimal numbers) defining the center of the license plate. The *center* point must be inside the input image.

`angle` – Clockwise correction of the source image in degrees (decimal number).

`scalePPM` – The real-world scale in pixels per meter of the input image in the *center* position (decimal number). The *scalePPM* defines the real size of the license plate in the image and does not consider the margin added by the detector.

Example:

```
[
  {
    "center": {
      "x": 176.5,
      "y": 362.0
    },
    "angle": -5.1,
    "scalePPM": 214.3
  },
  ...
]
```

CURL Command Examples:

Use automatic detection:

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

```
http://localhost:8080/RESTServer/lpRecognition
```

Use automatic detection with a region of interest in JSON file:

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

Use automatic detection with a region of interest in JSON string:

```
curl -X POST \
-H "Content-Type: multipart/form-data" \
-F "file=@image.jpg" \
-F "roi={\"topLeftX\":176,\"topLeftY\":362,\"topRightX\":1234,
\"topRightY\":362,\"bottomLeftX\":176,\"bottomLeftY\":1000,
\"bottomRightX\":1234,\"bottomRightY\":1000}" \
http://localhost:8080/RESTServer/lpRecognition
```

Use automatic detection with a simplified form of region of interest in JSON string:

```
curl -X POST \
-H "Content-Type: multipart/form-data" \
-F "file=@image.jpg" \
-F "roi={\"leftX\":176,\"topY\":362,\"rightX\":1234,\"bottomY\":1000}" \
http://localhost:8080/RESTServer/lpRecognition
```

Detections in JSON file:

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

Detections in JSON string:

```
curl -X POST \
-H "Content-Type: multipart/form-data" \
-F "file=@image.jpg" \
-F "lpDetections=[{\"center\":{\"x\":176.5,\"y\":362.0},\"angle\":-5.1,
\"scalePPM\":214.3}]" \
http://localhost:8080/RESTServer/lpRecognition
```

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 parameters **roi** and **lpDetections** 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).

Response Examples:

Automatic license plate detection, OCR and vehicle recognition:

```
[
  {
    "lpDetection": {
      "corners": [
        {
          "x": 817.2425537109375,
          "y": 892.52294921875
        },
        {
```

```
        "x": 1181.4339599609375,
        "y": 879.98583984375
    },
    {
        "x": 1184.06640625,
        "y": 56.4561767578125
    },
    {
        "x": 819.875,
        "y": 968.9932861328125
    }
],
"width": 364.40714,
"height": 76.51563,
"angles": [ -1.9716004, 0.0, 0.0 ],
"score": 0.9058307,
"occlusion": 0.0011757431,
"position": {
    "center": {
        "x": 1000.6544799804688,
        "y": 700.78296
    },
    "angle": 1.9716004,
    "scalePPM": 700.78296
}
},
"anprResult": {
    "country": "N",
    "countryScore": 0.99532324,
    "ocrText": "CV50070",
    "ocrTextScore": 0.9993771
},
"mmrResult": {
    "view": "frontal",
    "viewID": 1,
    "viewScore": 0.9999852,
    "category": "CAR",
    "categoryID": 2,
    "categoryScore": 0.99994755,
    "make": "VW",
    "makeID": 43,
    "makeScore": 0.999934,
    "model": "Passat",
    "modelID": 3569,
    "modelScore": 0.9994862,
    "generation": "Mk VI (2005)",
    "generationID": 3691,
    "generationScore": 0.9979515,
    "color": "WHITE",
    "colorID": 13,
    "colorScore": 0.73097533,
    "tags": [
        {
            "name": "caravan",
            "value": "no",
            "id": 2,
            "score": 0.99945587
        },
        {
            "name": "law_enforcement",
            "value": "yes",
            "id": 3,
            "score": 1.0
        },
        {
            "name": "ambulance",
            "value": "no",
```

```

        "id": 6,
        "score": 0.999996
    },
    {
        "name": "fire_brigade",
        "value": "no",
        "id": 8,
        "score": 0.9999945
    },
    {
        "name": "push_bumper",
        "value": "no",
        "id": 28,
        "score": 0.99893206
    }
    ]
}
]

```

License plate detection provided in the request:

```

[
  {
    "lpDetection": {
      "angles": [ -1.9716004, 0.0, 0.0 ],
      "position": {
        "center": {
          "x": 1000.6544799804688,
          "y": 700.78296
        },
        "angle": 1.9716004,
        "scalePPM": 700.78296
      }
    },
    "mmrResult":{
      "view": "frontal",
      "viewID": 1,
      "viewScore": 0.9999852,
      "category": "CAR",
      "categoryID": 2,
      "categoryScore": 0.99994755,
      "make": "VW",
      "makeID": 43,
      "makeScore": 0.999934,
      "model": "Passat",
      "modelID": 3569,
      "modelScore": 0.9994862,
      "generation": "Mk VI (2005)",
      "generationID": 3691,
      "generationScore": 0.9979515,
      "color": "WHITE",
      "colorID": 13,
      "colorScore": 0.73097533,
      "tags": [
        {
          "name": "caravan",
          "value": "no",
          "id": 2,
          "score": 0.99945587
        },
        {
          "name": "law_enforcement",
          "value": "yes",
          "id": 3,
          "score": 1.0
        }
      ]
    }
  }
]

```

```

        "name": "ambulance",
        "value": "no",
        "id": 6,
        "score": 0.999996
    },
    {
        "name": "fire_brigade",
        "value": "no",
        "id": 8,
        "score": 0.9999945
    },
    {
        "name": "push_bumper",
        "value": "no",
        "id": 28,
        "score": 0.99893206
    }
    ]
}
]

```

Notes:

When the **lpDetections** form field is provided in the request, the response does **not** contain the **anprResult** element and **lpDetection** element contains only the fields from the request.

If the **LPM License Plate Detector** engine is not running and the **lpDetections** form field is not provided in the request, an empty array is returned in the response.

The **anprResult** element is returned in the response only if the **LPM License Plate Detector** and the **LPM OCR** engines are running and the **lpDetections** form field is **not** provided in the request.

To obtain the **mmrResult** element in the response, running the **MMR License Plate based** engine is required. Additionally, either the **LPM License Plate Detector** and the **LPM OCR** engines must be running, or the **lpDetections** form field must be provided in the request.

Response Definitions:

The following table describes each item in the response.

| Response item | Description | Data type |
|-----------------------|--|------------------|
| lpDetection | Information about the detected license plate location in the input image. | Object |
| lpDetection/corners | An array of top-left, top-right, bottom-right and bottom-left corners of the detected license plate. | Array of objects |
| lpDetection/corners/x | The horizontal coordinate of the license plate corner within the image. | Decimal number |
| lpDetection/corners/y | The vertical coordinate of the license plate corner within the image. | Decimal number |
| lpDetection/width | The width of the detected license plate in pixels. | Decimal number |
| lpDetection/height | The height of the detected license plate in pixels. | Decimal number |

| | | |
|-------------------------------|--|--------------------------|
| lpDetection/angles | The license plate orientation - roll, pitch, yaw angles in degrees (clockwise). Currently, only the first array item, roll, is evaluated. | Array of decimal numbers |
| lpDetection/score | The license plate detection confidence factor. Range 0 – 1. | Decimal number |
| lpDetection/occlusion | Specifies how much the detection is occluded. Range 0 (not occluded) – 1 (fully occluded). | Decimal number |
| lpDetection/truncated | Specifies whether the detection is truncated (the bounding box does not cover the whole object). | Boolean |
| lpDetection/clusterID | ID of a cluster, to which this detection belongs. Not yet implemented in the current LPM modules. | Integer |
| lpDetection/clusterScore | Confidence factor for clusterID prediction. Range 0 – 1. | Decimal number |
| lpDetection/position | The "center, angle, scalePPM" structure defining the license plate position in the image for MMR computation. | Object |
| lpDetection/position/center | The coordinates of the license plate center within the image. | Object |
| lpDetection/position/center/x | The horizontal coordinate of the license plate center within the image. | Decimal number |
| lpDetection/position/center/y | The vertical coordinate of the license plate center within the image. | Decimal number |
| lpDetection/position/angle | Clockwise correction of the source image in degrees. | Decimal number |
| lpDetection/position/scalePPM | The real-world scale in pixels per meter of the image in the center position. The scalePPM defines the real size of the license plate in the image and does not consider the margin added by the detector. The value is calculated by the LPM OCR engine. | Decimal number |
| anprResult | Information about the recognized license plate. | Object |
| anprResult/country | The international license plate country code. If the value is "UNK", then it was recognized as a false positive detection. | String |
| anprResult/countryScore | The confidence factor for "country" prediction. Range 0 – 1. | Decimal number |
| anprResult/ocrText | The license plate text recognized by the OCR. | String |
| anprResult/ocrTextScore | The confidence factor for the OCR result. Range 0 – 1. | Decimal number |
| mmrResult | Information about the recognized vehicle attributes. | Object |
| mmrResult/view | The recognized vehicle view, either "FRONTAL", or "REAR". | String |

| | | |
|---------------------------|--|------------------|
| mmrResult/viewID | ID of the recognized vehicle view. | Integer |
| mmrResult/viewScore | The confidence factor for the view result. Range 0 – 1. | Decimal number |
| mmrResult/category | The recognized vehicle category, e.g., "BUS", "CAR", "HVT", ... For the full list of possible categories and their definition, check the Eyedea MMR SDK documentation. | String |
| mmrResult/categoryID | ID of the recognized vehicle category. | Integer |
| mmrResult/categoryScore | The confidence factor for the category result. Range 0 – 1. | Decimal number |
| mmrResult/make | The recognized vehicle manufacturer, e.g., "VW", "Ford", "Fiat", ... For the full list of possible categories, check the Eyedea MMR SDK documentation. | String |
| mmrResult/makeID | ID of the recognized vehicle manufacturer. | Integer |
| mmrResult/makeScore | The confidence factor for the make result. Range 0 – 1. | Decimal number |
| mmrResult/model | The recognized vehicle model (vehicle instance defined by a bodywork), e.g., "Golf", "Mondeo", "500", ... | String |
| mmrResult/modelID | ID of the recognized vehicle model. | Integer |
| mmrResult/modelScore | The confidence factor for the model result. Range 0 – 1. | Decimal number |
| mmrResult/generation | The recognized vehicle generation (vehicle mark and first model year), e.g., "Mk VI (2019)", "Mk I (2020)", ... | String |
| mmrResult/generationID | ID of the recognized vehicle generation. | Integer |
| mmrResult/generationScore | The confidence factor for the generation result. Range 0 – 1. | Decimal number |
| mmrResult/variation | The recognized vehicle variation (vehicle trim level and/or body type), e.g., "AMG", "AMG-Line SUV", "Coupe", ... | String |
| mmrResult/variationID | ID of the recognized vehicle variation. | Integer |
| mmrResult/variationScore | The confidence factor for the variation result. Range 0 – 1. | Decimal number |
| mmrResult/color | The recognized vehicle color, e.g., "BLUE", "GRAY", "RED", ... | String |
| mmrResult/colorID | ID of the recognized vehicle color. | Integer |
| mmrResult/colorScore | The confidence factor for the color result. Range 0 – 1. | Decimal number |
| mmrResult/tags | An array of recognized vehicle traits. | Array of objects |

| | | |
|----------------------|---|----------------|
| mmrResult/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 |
| mmrResult/tags/value | The value of the trait. Either "yes", or "no". | String |
| mmrResult/tags/id | ID of the trait value. | Integer |
| mmrResult/tags/score | The confidence factor for the tag value. Range 0 – 1. | Decimal number |

Detect and recognize vehicles (or “road users”) in the input image

| | |
|-----------------------------|---|
| Endpoint: | /boxRecognition |
| HTTP Method: | POST |
| Consumes Media Type: | multipart/form-data |
| Produces Media Type: | application/json |
| URL Parameters: | - |
| Data Parameters: | <p>file – form data parameter containing input image</p> <p>roi – (<i>optional</i>) form data parameter containing a JSON object that defines the area of the input image to be scanned by the detector</p> <p>The region of interest parameter is defined by its corners coordinates: topLeftX, topLeftY, topRightX, topRightY, bottomLeftX, bottomLeftY, bottomRightX and bottomRightY (decimal numbers).</p> <p><i>Example:</i></p> <pre>{ "topLeftX": 176, "topLeftY": 362, "topRightX": 1234, "topRightY": 362, "bottomLeftX": 176, "bottomLeftY": 1000, "bottomRightX": 1234, "bottomRightY": 1000 }</pre> <p>For the sides of a region of interest parallel to the sides of the image, a simplified form of <i>roi</i> defined by its sides can be used instead: leftX, topY, rightX and bottomY (decimal numbers).</p> <p><i>Example:</i></p> <pre>{ "leftX": 176, "topY": 362, "rightX": 1234, "bottomY": 1000 }</pre> <p>boxDetections – (<i>optional</i>) form data parameter containing a JSON object with an array of (vehicle) bounding box corners</p> |

The box detections parameter is defined as an array of vehicle's top-left and bottom-right corners coordinates: topLeftX, topLeftY, bottomRightX and bottomRightY (decimal numbers).

Example:

```
[
  {
    "topLeftX": 225,
    "topLeftY": 469,
    "bottomRightX": 1454,
    "bottomRightY": 1042
  },
  ...
]
```

CURL Command Examples:

Use automatic detection:

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

Use automatic detection with a simplified form of region of interest in JSON string:

```
curl -X POST \
  -H "Content-Type: multipart/form-data" \
  -F "file=@image.jpg" \
  -F "roi={\"leftX\":176,\"topY\":362,\"rightX\":1234,\"bottomY\":1000}" \
  http://localhost:8080/RESTServer/boxRecognition
```

Box detections in JSON file:

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

Box detections in JSON string:

```
curl -X POST \
  -H "Content-Type: multipart/form-data" \
  -F "file=@image.jpg" \
  -F "boxDetections=[{\"topLeftX\":225,\"topLeftY\":469,
  \"bottomRightX\":1454,\"bottomRightY\":1042}]" \
  http://localhost:8080/RESTServer/boxRecognition
```

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 parameters **roi** and **boxDetections** 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).

Response Example:

Automatic vehicle detection and recognition:

```
[
  {
    "boxDetection": {
      "topLeftX": 207.98,
```

```
"topLeftY": 254.33,
"bottomRightX": 1550.75,
"bottomRightY": 1209.36,
"score": 0.9650459,
"occlusion": 0.0010549
},
"mmrResult":{
  "view": "frontal",
  "viewID": 1,
  "viewScore": 0.9999329,
  "category": "CAR",
  "categoryID": 2,
  "categoryScore": 0.9997303,
  "make": "VW",
  "makeID": 43,
  "makeScore": 0.9999216,
  "model": "Passat",
  "modelID": 3569,
  "modelScore": 0.9990373,
  "generation": "Mk VI (2005)",
  "generationID": 3691,
  "generationScore": 0.9979515,
  "color": "WHITE",
  "colorID": 13,
  "colorScore": 0.94161636,
  "tags": [
    {
      "name": "ambulance",
      "value": "no",
      "id": 14,
      "score": 0.99998176
    },
    {
      "name": "caravan",
      "value": "no",
      "id": 10,
      "score": 0.99946773
    },
    {
      "name": "fire_brigade",
      "value": "no",
      "id": 16,
      "score": 0.9999783
    },
    {
      "name": "law_enforcement",
      "value": "yes",
      "id": 11,
      "score": 1.0
    },
    {
      "name": "pickup",
      "value": "no",
      "id": 38,
      "score": 0.670191
    },
    {
      "name": "push_bumper",
      "value": "no",
      "id": 30,
      "score": 0.9948259
    },
    {
      "name": "towed",
      "value": "no",
      "id": 51,
      "score": 0.99552554
    }
  ]
}
```

```

    },
    {
      "name": "wood_truck",
      "value": "no",
      "id": 46,
      "score": 0.9893941
    }
  ]
}
]

```

Notes:

When the **boxDetections** form field is provided in the request, the response elements **boxDetection** contain only the fields from the request: **topLeftX**, **topLeftY**, **bottomRightX** and **bottomRightY**.

If the **LPM Box Detector** engine is not running and the **boxDetections** form field is not provided in the request, an empty array is returned in the response.

Running the **MMR Box based** engine is required to obtain the **mmrResult** element in the response.

Response Definitions:

The following table describes each item in the response.

| Response item | Description | Data type |
|---------------------------|---|----------------|
| boxDetection | Information about the detected vehicle location in the input image. | Object |
| boxDetection/topLeftX | The horizontal coordinate of the top-left bounding box corner within the image. | Decimal number |
| boxDetection/topLeftY | The vertical coordinate of the top-left bounding box corner within the image. | Decimal number |
| boxDetection/bottomRightX | The horizontal coordinate of the bottom-right bounding box corner within the image. | Decimal number |
| boxDetection/bottomRightY | The vertical coordinate of the bottom-right bounding box corner within the image. | Decimal number |
| boxDetection/score | The vehicle detection confidence factor. Range 0 – 1. | Decimal number |
| boxDetection/occlusion | Specifies how much the detection is occluded. Range 0 (not occluded) – 1 (fully occluded). | Decimal number |
| boxDetection/truncated | Specifies whether the detection is truncated (the bounding box does not cover the whole object). | Boolean |
| boxDetection/clusterID | ID of a cluster, to which this detection belongs. Not yet implemented in the current LPM modules. | Integer |
| boxDetection/clusterScore | Confidence factor for clusterID prediction. Range 0 – 1. | Decimal number |

| | | |
|---------------------------|--|----------------|
| mmrResult | Information about the recognized vehicle attributes. | Object |
| mmrResult/view | The recognized vehicle view, either "FRONTAL", or "REAR". | String |
| mmrResult/viewID | ID of the recognized vehicle view. | Integer |
| mmrResult/viewScore | The confidence factor for the view result. Range 0 – 1. | Decimal number |
| mmrResult/category | The recognized vehicle category, e.g., "BUS", "CAR", "HVT", ... For the full list of possible categories and their definition, check the Eyedea MMR SDK documentation. | String |
| mmrResult/categoryID | ID of the recognized vehicle category. | Integer |
| mmrResult/categoryScore | The confidence factor for the category result. Range 0 – 1. | Decimal number |
| mmrResult/make | The recognized vehicle manufacturer, e.g., "VW", "Ford", "Fiat", ... For the full list of possible categories, check the Eyedea MMR SDK documentation. | String |
| mmrResult/makeID | ID of the recognized vehicle manufacturer. | Integer |
| mmrResult/makeScore | The confidence factor for the make result. Range 0 – 1. | Decimal number |
| mmrResult/model | The recognized vehicle model (vehicle instance defined by a bodywork), e.g., "Golf", "Mondeo", "500", ... | String |
| mmrResult/modelID | ID of the recognized vehicle model. | Integer |
| mmrResult/modelScore | The confidence factor for the model result. Range 0 – 1. | Decimal number |
| mmrResult/generation | The recognized vehicle generation (vehicle mark and first model year), e.g., "Mk VI (2019)", "Mk I (2020)", ... | String |
| mmrResult/generationID | ID of the recognized vehicle generation. | Integer |
| mmrResult/generationScore | The confidence factor for the generation result. Range 0 – 1. | Decimal number |
| mmrResult/variation | The recognized vehicle variation (vehicle trim level and/or body type), e.g., "AMG", "AMG-Line SUV", "Coupe", ... | String |
| mmrResult/variationID | ID of the recognized vehicle variation. | Integer |
| mmrResult/variationScore | The confidence factor for the variation result. Range 0 – 1. | Decimal number |
| mmrResult/color | The recognized vehicle color, e.g., "BLUE", "GRAY", "RED", ... | String |
| mmrResult/colorID | ID of the recognized vehicle color. | Integer |

| | | |
|----------------------|---|------------------|
| mmrResult/colorScore | The confidence factor for the color result. Range 0 – 1. | Decimal number |
| mmrResult/tags | An array of recognized vehicle traits. | Array of objects |
| mmrResult/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 |
| mmrResult/tags/value | The value of the trait. Either "yes", or "no". | String |
| mmrResult/tags/id | ID of the trait value. | Integer |
| mmrResult/tags/score | The confidence factor for the tag value. Range 0 – 1. | Decimal number |