




Carmen[®]

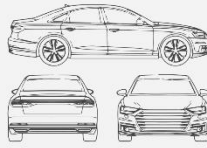
User Manual






ANPR RESULT:



CU 22 069 CU 22 069
COUNTRY: DENMARK
MAIN BACKG. COLOR: WHITE
MAIN FONT COLOR: BLACK
DATE: 16:36:21 07.03.25

MMR RESULT:



MAKE: AUDI 
MODEL: A8 
COLOR: GREY 
CATEGORY: CAR 
VIEWPOINT: FRONT SIDE 

```
31 // Load ANPR engine for european region
32 cmAnpr anpr = new cmAnpr("eur");
33
34 foreach (image in images)
```

User Manual for Carmen[®] related applications/tools/demos.

Carmen®

USER MANUAL

Document version: 2026.04.16.

Table of Contents

CARMEN LICENSING	8
LICENSE MANAGER	12
1. INTRODUCTION.....	12
2. WHAT IS THE PURPOSE OF THE LICENSE MANAGER.....	13
3. INSTALLATION OF THE LICENSE MANAGER	13
4. OPENING THE LICENSE MANAGER.....	14
4.1. ON WINDOWS	14
4.2. ON LINUX.....	14
5. STRUCTURE OF THE LICENSE MANAGER.....	15
5.1. STATIC LICENSES	15
5.2. DYNAMICALLY UPLOADABLE LICENSES	16
5.3. UPLOAD LICENSES.....	17
6. LICENSE INSTALLATION.....	19
7. LICMAN.....	21
8. TROUBLESHOOTING.....	28
ENGINE MANAGER.....	29
1. INTRODUCTION.....	29
2. ENGINE INSTALLATION	30
2.1. INSTALLATION ON WINDOWS	31

2.2.	INSTALLATION ON LINUX.....	33
3.	ENGINE MANAGER PRO APPLICATION.....	34
4.	UNINSTALLING ENGINE(S).....	38
4.1.	FROM WINDOWS.....	38
4.2.	FROM LINUX.....	38
5.	ENGMAN.....	39
LICENSE SERVER.....		43
1.	INTRODUCTION.....	43
2.	SERVER APPLICATION.....	44
2.1.	SERVER APPLICATION AS CONSOLE APPLICATION.....	45
2.1.1.	ON WINDOWS.....	45
2.1.2.	ON LINUX.....	45
2.2.	SERVER APPLICATION AS A SERVICE.....	47
2.2.1.	ON WINDOWS.....	47
2.2.2.	ON LINUX.....	47
2.3.	SERVER APPLICATION FOR MORE THAN ONE CLIENT.....	48
3.	CONFIGLSCLIENT.....	49
VIDEO SDK.....		51
1.	INTRODUCTION.....	51
2.	ENGINE AND LICENSE REQUIREMENTS.....	51
3.	VIDEO SDK INSTALLATION.....	52
3.1.	INSTALLATION ON LINUX.....	52
3.2.	INSTALLATION ON WINDOWS.....	54
4.	BUILD REQUIREMENTS.....	55
4.1.	LINUX.....	55
4.1.1.	C.....	55
4.1.2.	C++.....	55
4.1.3.	C#.....	56
4.2.	WINDOWS.....	58
4.2.1.	C.....	58
4.2.2.	C++.....	59
4.2.3.	C#.....	59
5.	VIDEO INPUT.....	60
5.1.	NETWORK STREAM.....	61
5.2.	VIDEO FILE.....	62
6.	A MINIMAL APPLICATION EXAMPLE.....	63
6.1.	CREATE ANPR.....	63
6.1.1.	C.....	63

6.1.2.	C++	63
6.1.3.	C#	63
6.2.	ONEVENTCALLBACK	64
6.2.1.	C	64
6.2.2.	C++	64
6.2.3.	C#	65
6.3.	CREATE STREAMPROCESSOR OBJECT	66
6.3.1.	C	66
6.3.2.	C++	66
6.3.3.	C#	66
6.4.	START STREAMPROCESSING	67
6.4.1.	C	67
6.4.2.	C++	67
6.4.3.	C#	67
7.	CONSTRUCTING COMPONENTS	68
7.1.	ANPR BUILDER	68
7.1.1.	TYPE	68
7.1.2.	LOCAL CONCURRENCY LIMIT	68
7.1.3.	REGISTERING ANPR PROFILE TO AN ALREADY BUILT ANPR OBJECT	69
7.2.	MMR BUILDER	70
7.2.1.	TYPE	70
7.2.2.	REGISTERING MMR PROFILE TO AN ALREADY BUILT MMR OBJECT	70
7.3.	ADAPTIVE RECOGNITION CLOUD BUILDER	71
7.3.1.	API KEY	71
7.3.2.	THE BUILT CLOUD OBJECT	71
7.4.	STREAMPROCESSOR BUILDER	71
7.4.1.	SOURCE	71
7.4.2.	NAME	72
7.4.3.	ANPR	72
7.4.4.	MMR	72
7.4.5.	CLOUD	72
7.4.6.	REGION	73
7.4.7.	LOCATION	73
7.4.8.	ANPR PROFILE ID AND MMR PROFILE ID	73
7.4.9.	AUTO RECONNECTION TO STREAM	73
7.4.10.	ANPR COLOR RECOGNITION	74
7.4.11.	MMR COLOR RECOGNITION	74
7.4.12.	EVENT DUPLICATION TIMEOUT	74



7.4.13.	ROI (REGION OF INTEREST).....	75
7.4.14.	EVENT CALLBACK.....	76
7.4.15.	ON FRAME CALLBACK.....	76
7.4.16.	STATUS CHANGE CALLBACK.....	77
7.4.17.	PROCESSING MODE.....	77
7.5.	LOGGER.....	78
7.5.1.	CHANGING LOGGING LEVEL.....	78
7.5.2.	SET LOGGING CALLBACK.....	78
7.6.	LICENSING.....	79
7.6.1.	LICENSING MODES.....	79
7.6.2.	SETTING THE LICENSING MODE.....	80
8.	RESULT CLASSES.....	81
8.1.	IMAGE CLASSES.....	81
8.1.1.	IMAGE.....	81
8.1.2.	IMAGEPROXY.....	81
8.2.	ANPR (AUTOMATIC NUMBER PLATE RECOGNITION) CLASSES.....	81
8.2.1.	PLATE.....	81
8.2.2.	PLATEDETECTION.....	81
8.3.	MMR (MAKE & MODEL RECOGNITION) CLASSES.....	81
8.3.1.	MMRDATA.....	81
8.3.2.	MMRDETECTION.....	82
8.4.	EVENT CLASSES.....	82
8.4.1.	PLATEONIMAGE.....	82
8.4.2.	MMRONIMAGE.....	82
8.4.3.	VEHICLE.....	83
8.4.4.	EVENT.....	83
9.	REGION LIST.....	84
10.	KNOWN ISSUES.....	85
	ADI DEMO.....	86
1.	INTRODUCTION.....	86
2.	MAIN SCREEN.....	87
3.	FILE MENU.....	89
3.1.	OPEN IMAGE(S).....	89
3.2.	OPEN DIRECTORY.....	89
3.3.	SAVE LOG.....	89
4.	EDIT MENU.....	90
4.1.	ANPR / MMR SETTINGS.....	90
4.2.	USING ADI DEMO WITH THE CLOUD ENGINE.....	91

- 4.3. CONFIGURING THE CLOUD ENGINE IN ADI DEMO 92
- 4.4. DEMO PREFERENCES 94
- 5. PROCESS MENU 97
- 6. HELP MENU 98
- 7. SET ROI/ROU ON THE IMAGE 99
- 8. MAGNIFIER 103
- 9. ANPR RESULT TABS 104
 - 9.1. LICENSE PLATE RESULT: 104
- 10. ANPR DATA 109
- 11. DATA LOGGING 110
- ADV DEMO 111
 - 1. INTRODUCTION 111
 - 2. MAIN SCREEN 113
 - 3. FILE MENU 115
 - 4. EDIT MENU 116
 - 4.1. TRIGGER CONFIGURATION 116
 - 4.2. ANPR THREADS 119
 - 4.3. ANPR SETTINGS 120
 - 4.4. ANPR PROCESSING PREFERENCES 121
 - 4.5. LOG SETTINGS 123
 - 5. VIEW 124
 - 5.1. FONTS 124
 - 5.2. RESULT IMAGE 125
 - 5.3. TABLE COLUMNS: 126
 - 5.4. TABLE ROW HEIGHT 126
 - 6. CAMERA MENU 127
 - 7. VIDEO PLAYER MENU 129
 - 8. PROCESS MENU 130
 - 9. HELP MENU 130
 - 10. ANPR RESULT TREE 131
 - 11. ANPR DATA 132
 - 12. DATA LOGGING 133
- ODI DEMO 134
 - 1. INTRODUCTION 134
 - 2. MAIN SCREEN 135
 - 3. FILE MENU 136
 - 3.1. OPEN DIRECTORY 136
 - 3.2. OPEN IMAGES 136

3.3. CLOSE.....	136
4. EDIT MENU.....	137
4.1. ENGINE SETTINGS.....	137
4.2. DEMO PREFERENCES.....	138
5. PROCESSING MENU.....	139
6. HELP MENU.....	140
7. OCR RESULT TREE.....	141
8. OCR DATA.....	143
9. DATA LOGGING.....	144
SDK DESCRIPTION.....	145
1. ANPR SAMPLE CODES.....	145
1.1. CMANPR01_LIST_ENGINES.....	145
1.2. CMANPR02_PLATE.....	145
1.3. CMANPR03_PLATES.....	145
1.4. CMANPR04_IMGLIST.....	145
1.5. CMANPR05_THREADS.....	145
1.6. CMANPR06_EMPTY_ADR.....	145
1.7. CMANPR07_K_LICENSE.....	146
1.8. CMANPR08_MULTI_STAGE.....	146
1.9. CMANPR09_MMR_AUTO.....	146
1.10. CMANPR10_MMR_MANUAL.....	146
1.11. CMANPR11_MMR_MANUAL_WO_PLATE.....	146
1.12. CMANPR12_PAXBELT.....	146
2. OCR SAMPLE CODES.....	147
2.1. CMOCR01_IMGSEQ.....	147
2.2. CMOCR02_MULTI_CODES.....	147
2.3. CMOCR03_ANPR_ADR_OCR.....	147
2.4. CMOCR04_THREADS.....	147
2.5. CMOCR05_MULTI_ENGINE.....	147
1. GX SAMPLE CODES.....	148
1.1. GXDEVICES01.....	148
1.2. GXDEVICES02.....	148
1.3. GXLICENSES01.....	148
2. EXAMPLE CODES AVAILABILITY.....	149
2.1. ON WINDOWS.....	149
2.2. ON LINUX.....	150
CONTACT INFORMATION.....	151

CARMEN LICENSING

Adaptive Recognition's Carmen® engines, known for their advanced ANPR, OCR, and document recognition capabilities, are designed with robust hardware key protection to ensure secure usage. These engines are embedded into products such as SDKs, middleware, service-type applications, and hardware devices like cameras and scanners, providing reliable, seamless performance in a variety of applications.

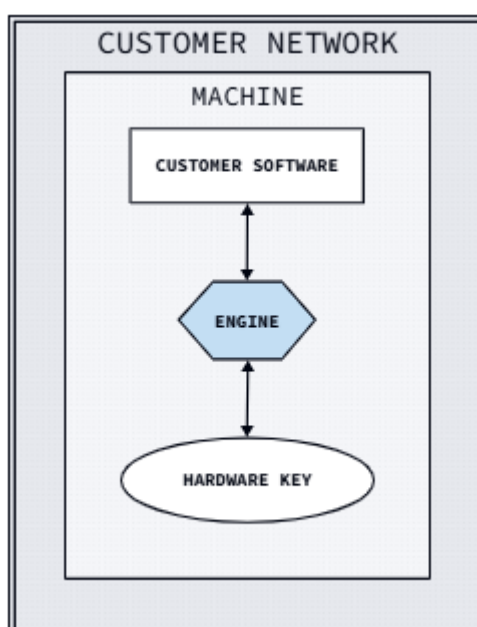
There are several methods available for utilizing these engines. One option involves using a hardware key, which can be connected directly to the computer running the engine. This ensures that the necessary licenses are applied effectively.

In addition to direct hardware key use, the License Server offers a flexible alternative. It allows licenses to be managed centrally, providing added convenience by removing the need for a direct hardware key connection to each device.

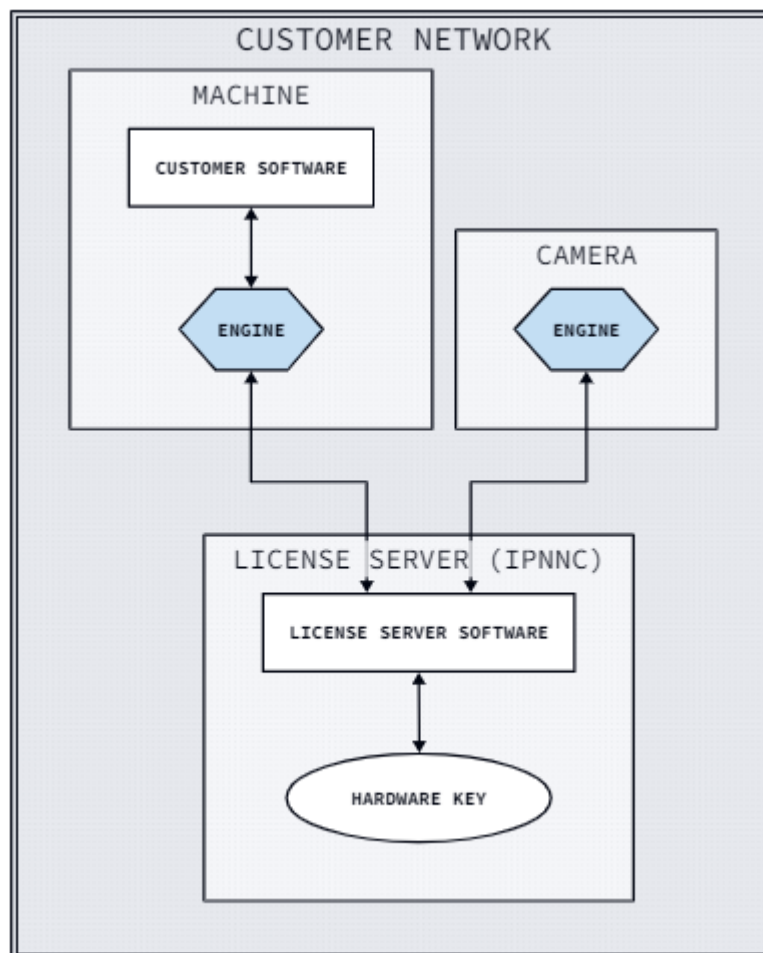
Additionally, a cloud-based licensing solution is available, offering even greater flexibility. This model requires only an internet connection, eliminating the need for physical hardware keys while providing the same reliable access to the engines and their capabilities.

In these setups, the Hardware Key, License Server, and License Server Client all work seamlessly to provide various options for managing and deploying the Carmen® engines based on specific needs.

As it was mentioned before, there are two methods that necessitate the purchase of a lifetime license file and its installation on the **hardware key**. The first method involves using the key directly by plugging it into the computer where the engine runs (with USB and PCI Express versions available).



The second method requires setting up a **License Server application** to share the licenses over the network, allowing multiple engine-using devices to work.



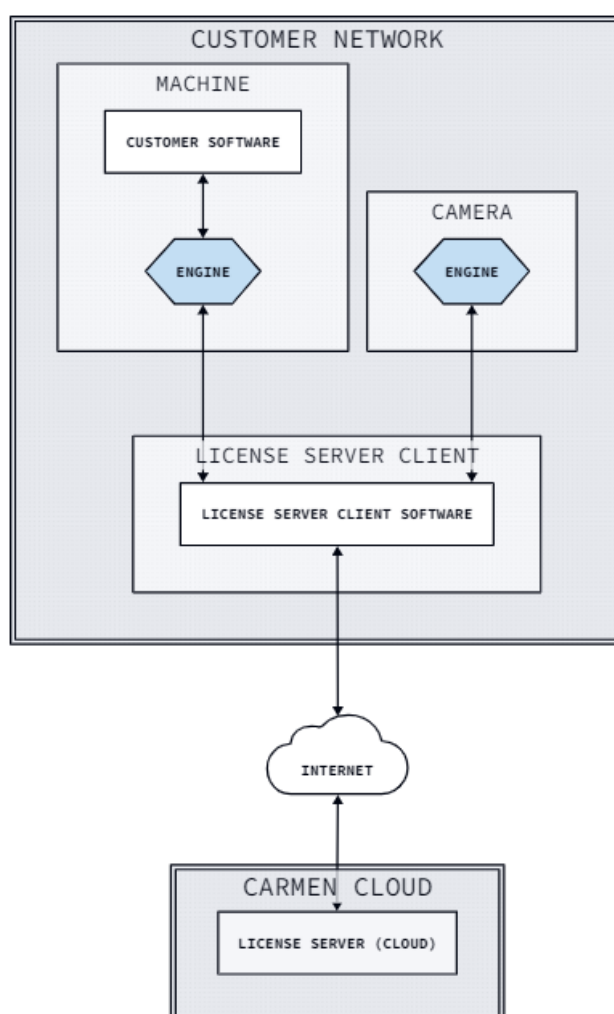
This model consists of a customer network with three main hosts:

1. A machine running customer software and the recognition engine.
2. A camera that also runs the engine.
3. A License Server application that serves both the machine and the camera, hosting the hardware key(s) for them.

This arrangement enables centralized management of licenses and offers flexibility in distributing the engine across multiple devices within the network.

A third, **cloud-based licensing option** has also been introduced, in addition to the existing two. With this model, customers are not required to purchase a hardware key or license, since it utilizes Carmen® Cloud's public license service. This service can be accessed by registering on the Carmen® Cloud website at <https://carmencloud.com/> and subscribing to the **Carmen® License Service**. For Carmen® Go, use the "Carmen® GO" option; for all other products (e.g. Carmen® FreeFlow SDK, Carmen® Video SDK) choose "Carmen® FreeFlow". The number of options you purchase determines the maximum number of license locks (image reads) in parallel. After obtaining an API key and running the **License Service Client** executable or deploying its lightweight docker image within your own infrastructure, you can begin using the product immediately. For more information, please visit our page here:

<https://carmencloud.com/docs/content/downloadable-products/license-service-client/tutorial>



In this cloud-based option, the customer network still comprises three hosts:

1. A machine running customer software and the recognition engine.
2. A camera running the engine.
3. The License Service Client serving both the machine and the camera, with a connection to the Carmen® Cloud public license service.

This modern approach leverages cloud technology to manage licenses, eliminating the need for physical hardware keys. The cloud-based infrastructure allows for scalable, remote management, and provides an accessible pathway to adopting newer licensing models.

In addition to the Carmen® License Service, another cloud-based subscription is available:

Carmen® Recognition Service.

This enables **cloud processing**, where recognition is performed in the Carmen® Cloud rather than by locally installed engines. In this mode, hardware keys, the License Server, and the LSC are not required. The ANPR, OCR and MMR processing is carried out by the Carmen® Cloud services, which are accessed via the Vehicle API and the Transportation & Cargo API using an API key. To obtain an API key, you must register on the Carmen® Cloud website at <https://carmencloud.com/> and subscribe to the Carmen® Recognition Service.

The Carmen® Recognition Service subscription also allows you to use **Carmen® Worker**, a containerized, self-hosted version of the Carmen® Cloud Vehicle and Transportation & Cargo APIs. With this method, recognition and data storage occur locally, while the container connects to the cloud-based License Service only for license verification.

More information about all available cloud-based subscription options can be found here:

<https://carmencloud.com/docs/content/tutorials/which-subscription.>



LICENSE MANAGER

1. INTRODUCTION

Thank you for choosing Adaptive Recognition's Optical Character Recognition technology. This short document will detail the use of the License Manager utility. If you have further questions after going through this manual, feel free to contact Adaptive Recognition's Support Team at <http://www.adaptiverecognition.com/support/>.

This application can be found in the following folder:

- On Windows: "C:\Program Files\Adaptive Recognition\Common Utils\LicenseManager\"
- On Linux: "/opt/gx64/LicenseManager/"

Note

The latest available version from License Manager: **7.3.2.1**

Important!

Please note, that this application is not available for ARM package on Linux.

Note

There are some videos on [YouTube](#) how to use our License Manager application. Feel free to check them as well.

2. WHAT IS THE PURPOSE OF THE LICENSE MANAGER

CARMEN® requires licenses to run. These licenses define which region-specific engines you are capable to run in what system configuration (single-core, dual-core, quad-core, etc.). The License Manager's main purpose is to help you install, update or delete these licenses by accessing your Hardware Key.

Note

Since your licenses are located on your Hardware Key, your already installed licenses will be visible in the License Manager application only if your Hardware Key is connected to your computer. Therefore, make sure that your Hardware Key (USB-dongle, PCI-e card, internal USB-key, etc.) is connected to your system.

3. INSTALLATION OF THE LICENSE MANAGER

The License Manager utility is part of all CARMEN® ANPR and OCR Software Package Installers above version 7.2.7.26 and installed automatically with your CARMEN® product. It is designed to provide the same look and features both under Windows and Linux operating systems. Further information about installation can be found in our [installation manual](#).



4. OPENING THE LICENSE MANAGER

4.1. ON WINDOWS

In order to launch the software, run the **LicenseManager_x64.exe**. You can start it either by locating the **LM** icon at the Adaptive Recognition > Common Utils > LicenseManager folder or Windows should be able to locate the application for you after pressing the 'Windows' key and typing 'License Manager'.

Note

The default file path for the License Manager is:
"C:\ProgramFiles\AdaptiveRecognition\CommonUtils\LicenseManager\LicenseManager_x64.exe".

Note

In older versions of CARMEN® (before 7.3.1.23) the application was in these folders:

- in case of 32bit OS: "C:\ProgramFiles (x86)\ARH\CommonUtils\LicenseManager"
- in case of 64bit OS: "C:\ProgramFiles\ARH\CommonUtils\LicenseManager"

4.2. ON LINUX

In case of Linux OS, the License Manager can be found under the following path:

/opt/gx64/LicenseManager/

Note

In older Carmen® versions (before 7.3.1.23), on 32bit OS you can find the License Manager here: */opt/gx32/LicenseManager/*

5. STRUCTURE OF THE LICENSE MANAGER

The screenshot shows the License Manager 7.3.2.0 (64 bit) interface. It is divided into three main sections, indicated by callouts 1, 2, and 3.

Section 1: Static licenses

Old license storage devices:

Serial num.	Device type

Licenses:

Lic. No.	Lic. type	Description

Section 2: Dynamically uploadable licenses

New license storage devices:

Serial num.	Device type	HW group
1190500	USB key	

Licenses:

Lic. ID	Lic. date	HWID/HWGRP	Lic. type	Expiry date	Description
1084459	2023.05.09	1190500	Normal	2030.12.31	CARMEN Go Anpr (UNI)
1084460	2023.05.09	1190500	Normal	2030.12.31	CARMEN Go Anpr (UNI)
1084461	2023.05.09	1190500	Normal	2030.12.31	CARMEN Go Anpr (UNI)
1084462	2023.05.09	1190500	Normal	2030.12.31	CARMEN Go Anpr (UNI)
1084463	2023.05.09	1190500	Normal	2030.12.31	CARMEN Go Anpr (UNI)
1084464	2023.05.09	1190500	Normal	2030.12.31	CARMEN Go Anpr (UNI)

Auto refresh devices and licenses
 Upload licenses
 Auto upload licenses for new devices
 Last licenses Best licenses
 Auto select and upload licenses

Refresh devices and licenses (2.f) Online manual (2.g)

Section 3: Upload licenses

License directory: G:/git/LicenseManager/bin/x64/licenses (3.a) Browse

Saved user licenses:

Lic. ID	Lic. date	HWID/HWGRP	Lic. type	Expiry date	Description

Upload licenses (3.c) Clear licenses (3.d)

Save changes (3.e) Summary (3.f)

Auto save after upload
 Auto summary after upload
 Create log file
 Log directory: Browse (3.h)

5.1. STATIC LICENSES

Static licenses were used in older systems, but for backwards compatibility we are keeping this segment in the License Manager. This part will be empty for you most probably. If not, please consider to contact your Sales manager and get an update. Many improvements have been made in our engines since static licenses were in use.

So, if you would like to use the latest engine, thereby increase the overall accuracy and achieve better recognition with the newer license plate types, a hardware key change and license update would be necessary.

5.2. DYNAMICALLY UPLOADABLE LICENSES

Here you could see all your connected Hardware Keys and the installed licenses on them.

2.a: New license storage devices

You will find your connected Hardware Keys here. The key's serial number, device type (e.g.: USB key or PCIe card) and hardware group is shown.

2.b: Licenses

Under licenses you will find the currently installed licenses on your selected Hardware Key. You can run engines which were released before the 'Expiry date' and having the same region as it is indicated in the 'Description'. For example, a CARMEN® ANPR (EUR) license with an "2020.12.31" expiry date will be able to run an EUR engine which was released before the end of December 2020.

2.c: Auto refresh devices

When enabled, License Manager refreshes your devices and licenses when a new Hardware Key is connected.

2.d: Upload licenses

Enable to upload new licenses on your Hardware Key.

2.e: Auto upload licenses

License manager will upload the selected Licenses automatically when this function is enabled. Last license means that when multiple licenses are available, the software will install the one with the latest 'License date'. When 'Best License' is selected, the program uploads the license with the latest Expiry date.

2.f Refresh devices and licenses

This button reloads your Hardware Keys and Licenses manually.

2.g Online manual

This button opens this manual.

5.3. UPLOAD LICENSES

Under this segment you are capable to upload licenses on your Hardware Key. Remember, this section is only visible when either the 'Upload licenses' or 'Auto upload licenses' is enabled.

3.a: License directory

Before you can select a license to upload, you have to choose the folder were that specific license is located.

Note

Please note, that you have to select the directory which contains the license and not the license itself!

3.b: Saved user licenses

License manager will show you the available licenses for the connected Hardware Key which are located in the *License directory* folder.

3.c: Upload licenses

This button will upload the selected licenses to your Hardware Key. If you would like to upload all of your licenses, select them all, but make sure that their 'License date' are the same, otherwise you will not be able to upload them!

Note

Please note, that in case you have some licenses uploaded to your Hardware Key, but you are trying to upload licenses with different "Lic. date", then it will delete all the current licenses from your Hardware Key and upload the newly selected ones on them.

3.d: Clear licenses

To delete all your licenses from your Hardware Key.

3.e: Save changes

Manually saves the previously made changes (when Auto save is not enabled).

3.f: Summary

To copy or save the current state of your Hardware Devices and Licenses.

3.g: Auto save after upload

Enable it to save the changes automatically after a license upload.

3.h: Auto summary after upload

Shows the summary after a license upload.

3.i: Create log file

When enabled the License Manager creates a log file to the selected directory.



6. LICENSE INSTALLATION

- 1) Make sure, that the Hardware Key, where you wish to upload your licenses is connected and visible in the License Manager!

Dynamically uploadable licenses

New license storage devices:			Licenses:					
Serial num.	Device type	HW group	Lic. ID	Lic. date	HWID/HWGRP	HW date	Expiry date	Description
1200005	USB key		111116	2020.07.01	1200005	2013.01.01	2021.06.30	CARMEN Go Anpr (NAM)
			111117	2020.07.01	1200005	2013.01.01	2021.06.30	CARMEN Accr
			111118	2020.07.01	1200005	2013.01.01	2021.06.30	CARMEN Accr
			111119	2020.07.01	1200005	2013.01.01	2021.06.30	CARMEN Accr
			111110	2020.07.01	1200005	2013.01.01	2021.06.30	CARMEN Accr
			111111	2020.07.01	1200005	2013.01.01	2021.06.30	CARMEN Anpr (NAM)

- 2) When **Upload Licenses** section not visible, enable **Upload licenses checkbox** (2.d).

Upload licenses

- 3) Hit **Browse** (3.a) and **Choose** the **directory** where your license files are stored.

As soon as you chose the folder, the available licenses should be visible in the **Saved user licenses** segment.

Saved user licenses:

Lic. ID	Lic. date	HWID/HWGRP	HW date	Expiry date	Description
111116	2020.07.01	1200005	2013.01.01	2021.06.30	CARMEN Go Anpr (NAM)
111117	2020.07.01	1200005	2013.01.01	2021.06.30	CARMEN Accr
111118	2020.07.01	1200005	2013.01.01	2021.06.30	CARMEN Accr
111119	2020.07.01	1200005	2013.01.01	2021.06.30	CARMEN Accr
111110	2020.07.01	1200005	2013.01.01	2021.06.30	CARMEN Accr
111111	2020.07.01	1200005	2013.01.01	2021.06.30	CARMEN Anpr (NAM)

Auto save after upload
 Auto summary after upload
 Create log file

Log directory:

Note

You cannot see the licenses listed? You may have chosen the license file, **not the folder itself!** **Select** only the **directory** and click on **Choose** again! Also, only those licenses will be visible which are dedicated to your connected Hardware Key. Check whether the Hardware Keys Serial number matches the serials indicated in the license (HWID/HWGRP).

- 4) When you are uploading new licenses, your old licenses will be removed automatically from your Hardware Key unless the License dates are the same. Yet, if you wish to delete your old licenses manually, you may do now by pressing the **Clear licenses** (3.d) button.

- 5) **Select the licenses** you wish to upload to your Hardware Key by click and drag your mouse over the requested licenses. You can also select all the licenses with the Ctrl+A shortcut. By pressing Ctrl while selecting the licenses you are capable to select multiple licenses as well.

- 6) Click on **Upload licenses** (3.c) to upload the highlighted licenses. Your new licenses should be visible under the **Licenses** section.

Your new licenses are now installed on your Hardware Key!



7. LICMAN

CONSOLE APPLICATION

On Linux you can find this application in this folder: **/opt/gx64/LicMan/**

On Windows, this application can be downloaded from the following location:

https://adaptiverecognition.com/app/uploads/DOC/Software/Support/LicMan_x64.zip

If you run this application in Terminal, or Command Prompt: you will get the following help:

```
Use: LicMan_x64.exe [version | -h | -?] | [list [-s -d | -fw | -raw | -SN:serial {-t:type} ] | summary [-raw]
| add -p:path {-SN:serial {-t:type}} {-save} | clear {-SN:serial {-t:type}} {-save}] {-json} {-o:outputfile}]
Copyright © 2017-2022, Adaptive Recognition
```

Parameters:

=====

version	Displays the version information of the program.
-h, -?	Shows this help.
list	Lists the license storage devices and the licenses
-s	static licenses
-d	dinamically uploadable licenses
-fw	Shows the firmware code version
-raw	Lists licenses in raw data format
summary	Create the summary of the dinamically uploadable licenses.
-raw	Lists licenses in raw data format
add	Uploads the licenses from a specified directory.
-p:path	the path of the directory contains the licenses.
clear	Clear the licenses from the device.

Global parameter (all commands except version and help):

- json The program create json format output.
- o:outputfile The program redirects the output from display to the outputfile.

Common parameters (add and clear):

- SN:serial dinamically uploadable licenses selected by the serial number and/or device type
- t:type Device type to select a device.
Usable values: "USB key"
 "Combo Smart"
 "Combo Scan"
 "PRMc"
 "PCIe card"
 "CARMEN SPI"
 "CARMEN I2C"
- save After successfully uploading or clearing the program saves the changes.

Some examples:

Gets the summary about the devices and licenses into summary.txt:

Command: LicMan_x64.exe/LicMan_x86_64.out summary -o:summary.txt

Output: The "summary.txt" file in the same folder with the following content:

Summary of the devices and licenses

=====

Devices:

=====

[1]: Device Type: USB key (1), Serial: 1111111

Code version: 3.8

Code subversion: 0.1

Code type: USB key

Max. license: 32

Memory size: 128 K

Time credit: Supported

Licenses:

=====

Num.	License Type	Count	Description
1	99901004	1	CARMEN Core 4
2	1affffff	4	CARMEN Anpr (EUR)
3	30ffffff	1	MMR (EUR)
4	16ffffff	1	CARMEN Ocr (ISO)

Check the version of this application:

Command: LicMan_x64.exe/LicMan_x86_64.out version

Output: License Manager (console version)

Version: 7.3.1.18

Copyright © 2017-2022, Adaptive Recognition



Print the dynamically uploadable licenses in raw format:

Command: LicMan_x64.exe/LicMan_x86_64.out list -d -raw

Output: Licenses in the system:

[1]: HWID: 01111111, Device Type: USB key (1), Serial: 1111111 :

```
606376-20210914-01111111-20130101--16ffffff-ffffff-20220630-00000000--00000000-CARMEN Ocr
( ISO )
606380-20210914-01111111-20130101--99901004-ffffff-20220630-00000000--00000000-CARMEN
Core 4
606381-20210914-01111111-20130101--1affffff-ffffff-20220630-00000000--00000000-CARMEN Anpr
( EUR )
606382-20210914-01111111-20130101--1affffff-ffffff-20220630-00000000--00000000-CARMEN
Anpr ( EUR )
606383-20210914-01111111-20130101--1affffff-ffffff-20220630-00000000--00000000-CARMEN
Anpr ( EUR )
606384-20210914-01111111-20130101--1affffff-ffffff-20220630-00000000--00000000-CARMEN
Anpr ( EUR )
606385-20210914-01111111-20130101--30ffffff-ffffff-20220630-00000002--00000000-MMR ( EUR )
*****
```

Print the dynamically uploadable licenses, firmware (CodeVersion) in json format:

Command: LicMan_x64.exe/LicMan_x86_64.out list -d -fw -json

```
Output: {
    "NewLicdevs": [
        {
            "Num": 1,
            "HWID": "01111111",
            "DevType": "USB key",
            "DevTypeID": 1,
            "Serial": 1111111,
            "CodeVersion": "3.8",
            "Licenses": [
                {
```

```
"LicID": " 606376",
"LicDate": "2021.09.14",
"ExpDate": "2022.06.30",
"Desc": "CARMEN Ocr ( ISO )"
}
{
"LicID": " 606380",
"LicDate": "2021.09.14",
"ExpDate": "2022.06.30",
"Desc": "CARMEN Core 4"
},
{
"LicID": " 606381",
"LicDate": "2021.09.14",
"ExpDate": "2022.06.30",
"Desc": "CARMEN Anpr ( EUR )"
},
{
"LicID": " 606382",
"LicDate": "2021.09.14",
"ExpDate": "2022.06.30",
"Desc": "CARMEN Anpr ( EUR )"
},
{
"LicID": " 606383",
"LicDate": "2021.09.14",
"ExpDate": "2022.06.30",
"Desc": "CARMEN Anpr ( EUR )"
},
{
"LicID": " 606384",
"LicDate": "2021.09.14",
"ExpDate": "2022.06.30",
"Desc": "CARMEN Anpr ( EUR )"
},
}
```



```
    {
      "LicID": " 606385",
      "LicDate": "2021.09.14",
      "ExpDate": "2022.06.30",
      "Desc": "MMR ( EUR )"
    }
  ]
}
}
```



Clear licenses from all attached devices and save this modification:

Command: LicMan_x64.exe/LicMan_x86_64.out clear -save

Output: [!] Device type: USB key, Serial: 1111111 clearing licenses...OK

! Important!

If you want to make this change as permanent do not forget to put '**-save**' at the end of the command, otherwise you will lose the modification. The HW key will hold the modification until the HW key is attached, but once it is detached it will fall back to the original configuration.

Add licenses from a folder which contains the ukeys file for an exact attached HW Key temporarily

Command: LicMan_x64.exe/LicMan_x86_64.out add -p: "c:\licenses" -SN:1111111

Output:

[!] Device type: USB key, Serial: 1111111:

Adding license: LicID: 606366, Lic. Date: 2021.09.14, Expiry Date: 2022.06.30, Description: CARMEN Core 8...OK

Adding license: LicID: 606367, Lic. Date: 2021.09.14, Expiry Date: 2022.06.30, Description: CARMEN Go 8 stream...OK

Adding license: LicID: 606368, Lic. Date: 2021.09.14, Expiry Date: 2022.06.30, Description: CARMEN Go Anpr (.NAM)...OK

Adding license: LicID: 606369, Lic. Date: 2021.09.14, Expiry Date: 2022.06.30, Description: CARMEN Go Anpr (NAM)...OK

Adding license: LicID: 606370, Lic. Date: 2021.09.14, Expiry Date: 2022.06.30, Description: CARMEN Go Anpr (NAM)...OK

Adding license: LicID: 606371, Lic. Date: 2021.09.14, Expiry Date: 2022.06.30, Description: CARMEN Go Anpr (NAM)...OK

Adding license: LicID: 606372, Lic. Date: 2021.09.14, Expiry Date: 2022.06.30, Description: CARMEN Go Anpr (NAM)...OK

Adding license: LicID: 606373, Lic. Date: 2021.09.14, Expiry Date: 2022.06.30, Description: CARMEN Go Anpr (NAM)...OK

Adding license: LicID: 606374, Lic. Date: 2021.09.14, Expiry Date: 2022.06.30, Description: CARMEN Go Anpr (NAM)...OK

Adding license: LicID: 606375, Lic. Date: 2021.09.14, Expiry Date: 2022.06.30, Description: CARMEN Go Anpr (NAM)...OK

! Important!

If you just forgot to put '**-save**' at the end of the command, just run the previous command again, but using the '**-save**' parameter. The command will show you ERRORS, because the licenses are already on the HW key, but at the end of the process the command will save successfully, and you will not lose the modifications, once the HW Key is detached.

! Important!

It is allowed to upload only those licenses to the HW Key which has the same Lic Date.

Note

If you would like to update your licenses then the process would be the following:

1. Clear the current licenses from the attached HW Key(s) or from an exact HW Key
2. Add the licenses from a folder which contains the ukeys file(s) and save the changes

8. TROUBLESHOOTING

Problem observed	Solution
Installation	
I cannot find the application!	<ul style="list-style-type: none"> • Please re-visit the Opening the License Manager section and make sure that you were thoroughly followed the instructions written there! • Make sure that Carmen® is installed on your computer! Please, run the Installer again. If it offers to install, choose that option. If it offers to Change, choose that option. Check whether the License Tools option is installed! If not, install it! Otherwise please contact support at https://adaptiverecognition.com/support/. • If you are using Carmen® 7.3.1.22 or older, maybe you have installed the 32-bit variant of the License Manager. In that case find them at either the Program Files (x86) folder [Windows] or at /opt/gx32/LicenseManager/ [Linux].
License upload	
The choose button is greyed out! I cannot select my license file!	<p>Select the directory which contains the license and not the license file itself. When the folder is selected the Choose button will be active!</p>
I chose the directory where my license is located, but it is not visible under the Saved User Licenses part, therefore I cannot upload!	<ul style="list-style-type: none"> • Only those licenses will be visible which are dedicated to your connected Hardware Key. The first 8 numbers from the name of the license file should be the dedicated Hardware Key's ID. If they don't match, it means that license is generated for another Hardware Key! • Are you sure that your Hardware Key is connected? The present of the dedicated Hardware Key is a must, otherwise the License Manager will not show your licenses!
I uploaded some licenses to my Hardware Key, but my old licenses are gone! I want both the old and the new licenses on my Hardware Key!	<ul style="list-style-type: none"> • The Hardware Key can only hold licenses which are having the same Lic. Date. If your licenses are having different license dates, then unfortunately you cannot have them both on your Hardware Key. Please contact your sales person or our support team to request a new license with the same date as your old one! • Your 'deleted' licenses are not lost! You can reupload them anytime by locating the other license file and upload those licenses again.
Licenses	
My licenses are highlighted with yellow! What does it mean?	<ul style="list-style-type: none"> • It means that your licenses currently updated on your Hardware Key are passed their 'Expiry Date'. Your license is still functional; all your ANPR or OCR processes will run indefinitely, however you will not be able to run engines which were released later then the Expiry Date. • Please contact your sales person to request a new license with an extended Expiry date if you would wish to use recently released up-to-date engines!
My licenses are highlighted with red! What does it mean?	<p>When "K"-license is in use, the License Manager is constantly monitoring the date and time of your system. If it finds an inconsistent change it will block your license. Please contact our support to resolve this issue!</p>

ENGINE MANAGER

1. INTRODUCTION

Engine Manager Pro is a utility for Windows and Linux based systems, that enables the management of all kinds of OCR engines for the CARMEN® ANPR and CARMEN® OCR Software's main module. This utility is part of the CARMEN® ANPR / OCR Software's download package, and it is automatically installed along with the software.

This document will detail the following:

- How to install engines on Windows and Linux systems
- How to use the Engine Manager PRO application
- How to remove/uninstall engines from Windows and Linux systems

This application can be found in the following folder:

- On Windows:
"C:\ProgramFiles\AdaptiveRecognition\CARMENsoftwares\EngineManagerPro\"
- On Linux: "/opt/gx64/EngineManagerPro/"

Note

The latest available version from Engine Manager Pro: **7.3.2.1**

Important!

Please note, that this application is not available for ARM package on Linux.

2. ENGINE INSTALLATION

! Important!

Before installing the engine(s) make sure that no other process on the PC is using CARMEN® ANPR / OCR.

Engines are available directly from Adaptive Recognition's documentation page: [Carmen® FreeFlow documentation — "Downloads" → "Engines" section](#)

Engines arrive in a zip file. For example, the 2025 Q2 general engine: **cmanpr-gen-7.3.19.103_25Q2.zip**. This zip file contains 2 different folders:

- linux
- windows

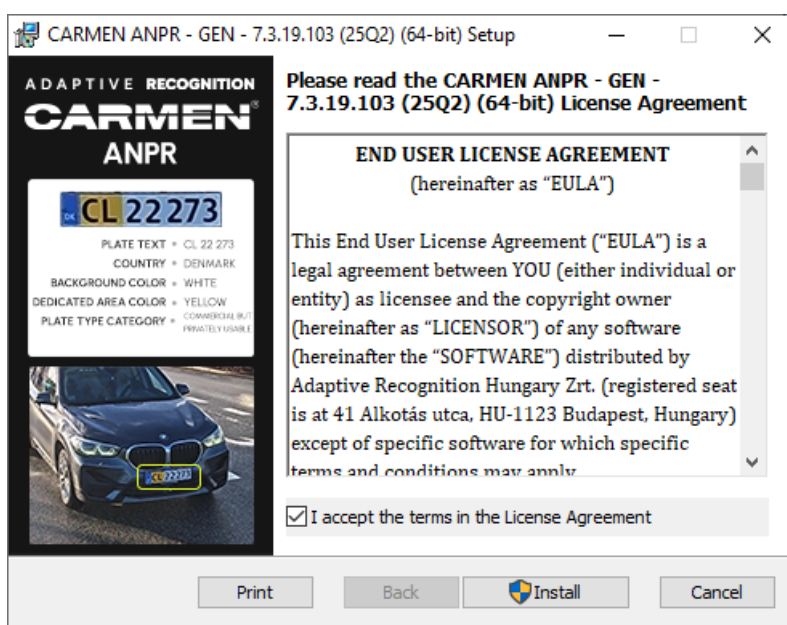
📄 Note

CARMEN® ANPR and CARMEN® OCR engines are the same from installation point of view so all the rest is true for the CARMEN® OCR engines as well.

2.1. INSTALLATION ON WINDOWS

In the "windows" folder in the zip file which you have downloaded from ATSS you can find the installer files. For example, the 2025 Q2 general engine: ***cmanpr-gen-7.3.19.103_25Q2-x86.msi*** or ***cmanpr-gen-7.3.19.103_25Q2-x64.msi*** – (32-bit and 64-bit versions). This folder also contains a text file which refers to this document and the previous version of it.

After locating the engine(s) in the selected download folder, simply double click on it to begin installation.



As a first step, you need to **accept** the EULA by checking the "**I accept the terms in the License Agreement**" box and click on **Install button**.

Note

The newly installed engine will be the default engine.

To check the installed engine(s), open the **Engine Manager Pro** application (see [chapter 2](#)).

Note

You can locate **GXSD.DAT** in this folder: **c:/ProgramData/gx/**

This file contains all of the installed engines and their properties. It is NOT RECOMMENDED to change this file manually, please use **Engine Manager Pro**, or the **Demo Applications** to do that.

Important!

From 20Q3 engines vcredist (Microsoft Visual C++ Redistributable packages for Visual Studio 2015, 2017, 2019, and 2022) is required on Windows systems. You can download it from [here](#).



2.2. INSTALLATION ON LINUX

In the "linux" folder in the zip file which you have downloaded from ATSS you can find the installer files separated in folders by architecture (arm, arm64, x64 and x86). For example, the 2020 Q3 general engine for x64 contains the following files:

Files in the package:

```
cmanpr-gen-7.3.12.169_20Q3-x86_64.tar.gz
_install_cmanpr-gen-7.3.12.169_20Q3-x86_64.sh
_uninstall_cmanpr-gen-7.3.12.169_20Q3-x86_64.sh
```

Where "**`_install_cmanpr-gen-7.3.12.169_20Q3-x86_64.sh`**" is the script for installing the engine and "**`_uninstall_cmanpr-gen-7.3.12.169_20Q3-x86_64.sh`**" is the script for uninstalling the engine.

The install script does all the necessary file copies to the relevant folders and inserts the engine properties into the gxsd.dat. The engine will be **set as the default engine**.

The uninstall script does the opposite. (For further information please check [this](#) chapter)

Note

You can locate **GXSD.DAT** in this folder: **`/var/gx`**

This file contains all of the installed engines and their properties. It is NOT RECOMMENDED to change this file manually, please use **Engine Manager Pro**, or the **Demo Applications** to do that.

Important!

Starting from CARMEN® ANPR 7.3.1.28 and CARMEN® OCR 7.3.1.19, the software package does not include the Intel "iclib" package, which used to be required for handling CARMEN® ANPR / OCR engines released prior to 22Q1 on Linux x86_64 systems. If you intend to use CARMEN® ANPR 7.3.1.28+ or CARMEN® OCR 7.3.1.19+ with engine versions prior to 22Q1, you may need to download and install the iclib-x.x.x-xx-zzz.tar.gz package separately. For the installation procedure, please refer to the [Carmen® Installation Guide](#).

To check the installed engine, open the **Engine Manager Pro** application (see [chapter 2](#)).

3. ENGINE MANAGER PRO APPLICATION

Once the CARMEN® ANPR / OCR Software and the recognition engine(s) have been installed open the Engine Manager Pro application.

On Windows, this application is available for download at:

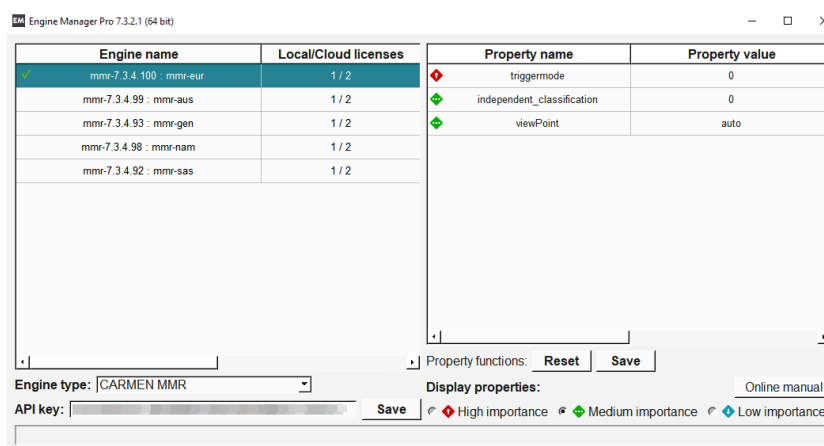
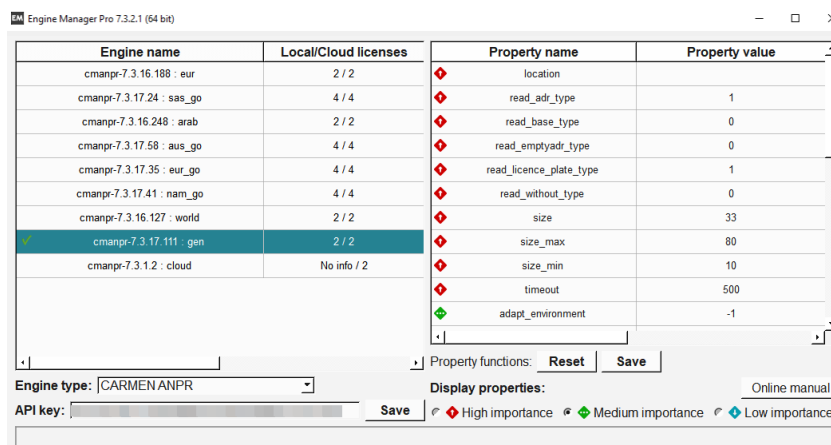
https://adaptiverecognition.com/app/uploads/DOC/Software/Support/EngMan_x64.zip

On Linux: navigate to `/opt/gx64/EngineManagerPro/`

When you start the utility, the following window will appear:

Important!

If you want to take effect your modification in this application, you must run it as an **administrator** or **root**, otherwise the application will not be able to save the changes into **GXSD.DAT** file!



EngMan Api key feature:

When you subscribe to Carmen Cloud License, the API key becomes essential for authentication purposes. It ensures that your application is authorized to use the services provided by Carmen Cloud. Once the API key is stored, it functions as a bridge between your application and the cloud. Without an API key, unauthorized access is denied, preventing misuse of the services.

The API key feature within Engine Manager is only for requesting licenses within Engine Manager, it does not make any changes to the system.

When you enter the API key, the screen updates within 2-3 seconds to display the licenses. The **Save** button only saves the API Key in the Engine Manager configuration so that it is entered in the application when it is reopened.

Engine name:

List of all successfully installed engines.

Local Licenses:

Available number of licenses for the given engine (Single: 1, Dual: 2, Quad: 4, No available licence for the engine: 0, Missing files: No info)

Cloud licenses:

In the "licenses" column next to the engines, it will also show whether the given engine has a Cloud license.

Property name / Property value:

Properties and their values for finetuning the engine. For quick information about the property, hover the mouse over it.

The screenshot shows the Engine Manager Pro 7.3.2.1 (64 bit) interface. It features a table of engines and their properties, and a detailed view of a specific property.

Engine name	Local/Cloud licenses	Property name	Property value
cmanpr-7.3.16.188 : eur	2 / 2	location	
cmanpr-7.3.17.24 : sas_go	4 / 4	size	15
cmanpr-7.3.16.248 : arab	2 / 2	size	
cmanpr-7.3.17.58 : aus_go	4 / 4	size	
cmanpr-7.3.17.35 : eur_go	4 / 4	tim	
cmanpr-7.3.17.41 : nam_go	4 / 4	adapt_er	
cmanpr-7.3.16.127 : world	2 / 2	confider	
✓ cmanpr-7.3.17.111 : gen	2 / 2	contra	
cmanpr-7.3.1.2 : cloud	No info / 2	gapto	
		general	5
		heapfreefreq	0

The 'size' property is highlighted with a tooltip:

size
The average height of the number plate characters in the image in pixels.
Possible values: positive integers (greater than 10)
Default value: varies with each engine release; default property value can be queried by the `GetProperty()` function
Suggested value: leave the default value

At the bottom of the interface, there are controls for 'Engine type' (set to CARMEN ANPR), 'API key', and 'Display properties' (with radio buttons for High, Medium, and Low importance). Buttons for 'Reset', 'Save', and 'Online manual' are also present.

For detailed information about the properties, click "Online manual" button which will open [this link](#) in case of ANPR engines, [this link](#) in case of OCR engines and [this link](#) in case of MMR engines.

Engine type:

With this selector, you can filter the installed engines to show the ANPR regional engines (like EUR, CAS, NAF, etc...), the ORC related engines (like ACCR, UIC, ISO, etc ...) and the MMR regional engines (like EUR, CAM, etc ...).

Display properties:

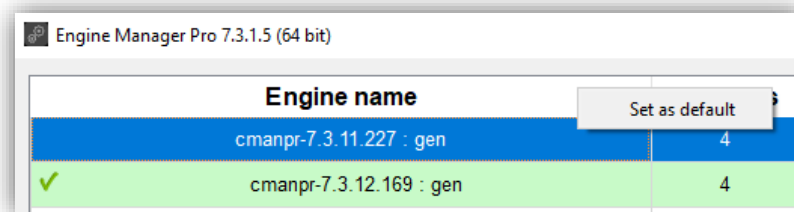
With this selector, you can filter the engine properties by their importance or their impact to the recognition results.

Default engine:

After installing a new engine, that will be set as the default engine. It is highlighted with green and a checkmark is also visible to the left of the engine name.

Changing the default engine (which is currently in use):

If you want to change the default engine, right click on the engine which you would like to make as default and click "set as default".



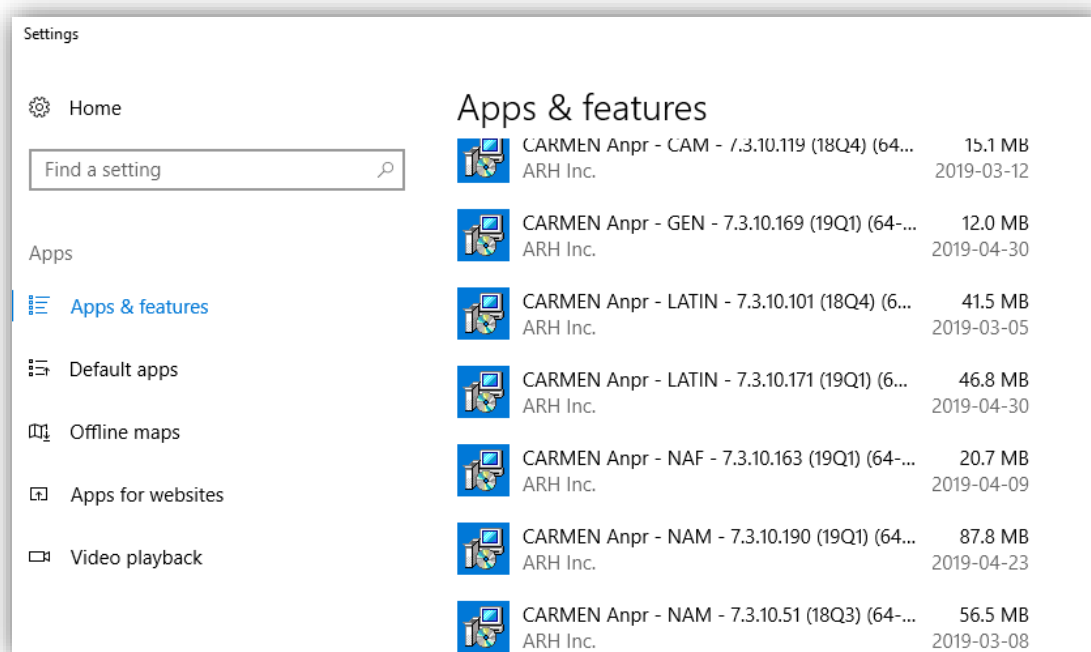
4. UNINSTALLING ENGINE(S)

Note

After deletion of an engine which is the current default engine, the default engine will be the first found engine in the GXSD.DAT file.

4.1. FROM WINDOWS

Go to Windows → Settings → Apps & features: select the desired engine from the **Installed engines**. Once the engine that you would like to uninstall have been selected, click the **[Uninstall]** button to uninstall the selected engine.



4.2. FROM LINUX

The „`_uninstall_cmanpr-gen-7.3.10-169_19Q1-x86_64.sh`“ is the script for uninstalling the engine.

The uninstall script does all the necessary file deletion from the relevant folders and deletes the engine properties from the gxsd.dat.

5. ENGMAN

CONSOLE APPLICATION

On Linux you can find this application in this folder: **/opt/gx64/EngMan/**

On windows this application is not part of the installed package, please turn to [AR support](#) if you need this application.

If you run this application in Terminal, or Command Prompt: you will get the following help:

```
"Use: EngMan_x64.exe/EngMan_x86_64.out version | -h | -? | [[list {-lic} | getdef | setdef -e:engine] {-anpr} {-ocr} {-mmr} {-alltype}] {-json} {-o:outputfile}]
```

Copyright © 2019-2022, Adaptive Recognition

Parameters:

=====

version Displays the version information of the program.

-h, -? Shows this help.

List Lists the installed engines. If no -anpr, -ocr and -mmr switches added lists anpr, ocr and mmr engines too.

 -lic checks the licenses for the engine

getdef Gets the default engine.

setdef Sets the default engine.

 -e:engine the name of the engine to set the default engine.

Global parameter (all commands except version and help):

-anpr Runs the command on the anpr engines

-ocr Runs the command on the ocr engines

-mmr Runs the command on the mmr engines

-alltype Runs the command on the anpr,ocr and mmr engines

-json The program create json format output.

-o:outputfile The program redirects the output from display to the outputfile.

Some examples:

Gets the installed ANPR engines:

Command: EngMan_x64.exe/EngMan_x86_64.out list -anpr

Output: Engines (ANPR):

```
=====
cmanpr-7.3.14.51 : arab
cmanpr-7.3.14.101 : eur
cmanpr-7.3.14.95 : nam
```

Gets the installed ANPR engines with license checking:

Command: EngMan_x64.exe/EngMan_x86_64.out list -lic -anpr

Output: Engines (ANPR):

```
=====
cmanpr-7.3.14.51 : arab License: FOUND
cmanpr-7.3.14.101 : eur License: FOUND
cmanpr-7.3.14.95 : nam License: NOT FOUND
```

Gets the installed engines (ANPR, OCR, MMR) with license checking in json format:

Command: EngMan_x64.exe/EngMan_x86_64.out list -lic -alltypes -json

Output:

```
{
  "Engines (ANPR)": [
    {
      "Engine": "cmanpr-7.3.14.51 : arab",
      "License": "FOUND"
    },
    {
      "Engine": "cmanpr-7.3.14.101 : eur",
      "License": "FOUND"
    },
    {
      "Engine": "cmanpr-7.3.14.95 : nam",
      "License": "NOT FOUND"
    }
  ]
}
```

```

],
"Engines (MMR)": [
  {
    "Engine": "mmr-7.3.2.31 : mmr-eur",
    "License": "FOUND"
  },
  {
    "Engine": "mmr-7.3.2.27 : mmr-arab",
    "License": "FOUND"
  },
  {
    "Engine": "mmr-7.3.2.32 : mmr-nam",
    "License": "NOT FOUND"
  }
],
"Engines (OCR)": [
  {
    "Engine": "cmocr-7.3.2.100 : isoilu",
    "License": "FOUND"
  },
  {
    "Engine": "cmocr-7.3.2.84 : uic",
    "License": "FOUND"
  }
],
]
}

```

Gets the default engines and write the result to the result.txt:

Command: EngMan_x64.exe/EngMan_x86_64.out getdef -alltype -o:result.txt

Output: The "result.txt" file in the same folder with the following content:

Default anpr: cmanpr-7.3.14.51 : arab

Default mmr: mmr-7.3.2.31 : mmr-eur

Default ocr: cmocr-7.3.2.100:isoilu

Sets the default ocr engine:

Command: `EngMan_x64.exe/EngMan_x86_64.out setdef -e:"cmocr-7.3.2.84 : uic" -ocr"`

Output: nothing, but after checking the default 'ocr' engine with this command:

`EngMan_x64.exe/EngMan_x86_64.out getdef -ocr` the output will be:

Default ocr: cmocr-7.3.2.84:uic

Check the version of this application:

Command: `EngMan_x64.exe/EngMan_x86_64.out version`

Output: Engine Manager (console version)

Version: 7.3.1.5

Copyright © 2019-2022, Adaptive Recognition



LICENSE SERVER

1. INTRODUCTION

From CARMEN® ANPR version 7.3.1.24 and CARMEN® OCR version 7.3.1.15, License Server is part of the installer package. This application allows users to share any Adaptive Recognition license over their network.

The server computer, which has the [hardware key\(s\)](#) (HW Key or NNC) with the licenses plugged in, runs the Server Application.

Client computer(s) may join this server by using the Client Application that, upon a successful configuration, will make it possible to do ANPR/OCR processes without any HW Key attached to the client computer(s).

Note

This solution is provided free for up to one client.

If you would like to connect more than one client to the server, please contact your Sales Manager to check the possibilities.

Note

The latest available version from License Server: **7.3.1.38**.

2. SERVER APPLICATION

This application can be found here:

- On Windows: "C:\Program Files\Adaptive Recognition\Common Utils\LicenseServer\Server\
- On Linux: "/opt/gx64/LicenseServer/"

The Server application allows the sharing of your licenses over the network. On Windows, there is a **LicenseServer.cfg** file in the same folder where the application is installed. This file is in the following folder on Linux-based computers: **/etc/gxd/licserver**. This file stores the configuration settings, which the user can modify manually.

This file contains the following properties:

Property and its default value	Description
SRVPORT=8998	This is the server port. The Client will connect through this to the Server. Modification of the property is permitted. Please make sure that the set port is allowed on your firewall.
CHECKPORT=48315	This is for an internal service port. Please DO NOT MODIFY .
LOGPATH=.\\logs\	The folder where the Server application stores the logs. If it is empty, the log files will be saved into the same folder where the application is installed.
IDLETIMEOUT=600	Indicates the idle timeout in milliseconds. The Client Application sends "keepalive" messages every 40 seconds to make the connection stable. If there is no "keepalive" message, then the Server application breaks the connection to prevent a client from being stuck.
NNCTIMEOUT_MS=2000	This value defines, in milliseconds, when one NNC lock request has to be finished. If the CPU load is high and the number of available licenses is low, it may result in a timeout error, indicated by the "HW Key lock error" message and the loss of the ANPR/AICR/MMR process.
LOCKTIMEOUT=3000	Sets, in milliseconds, the maximum time one license can be locked.
NPROC=6	The number of threads handling non-license lock calls.
NPROCLOCK=6	The number of threads handling license lock calls.
SAVEDATA=1	If it is set to "1," then every invalid request's binary data will be saved. Any other value than "1" will not save invalid requests' binary data.

If you properly set all the above properties in this file, please open a Console/Terminal window in the Server application's folder and just start the application.

2.1. SERVER APPLICATION AS CONSOLE APPLICATION

2.1.1. ON WINDOWS

Just go to this folder:

"C:\ProgramFiles\AdaptiveRecognition\CommonUtils\LicenseServer\Server\" and run this file: **LicenseServer_x64.exe** in command prompt.

2.1.2. ON LINUX

There are 3 possibilities to start the Server Application on Linux from the Console:

1. Go to this folder: **"/opt/gx64/LicenseServer/"** and this this command:
../LicenseServer_x86_64"

Note

Please make sure that kernel drivers are running if you are using this option.

2. Go to this folder: **"/opt/gx64/LicenseServer/"** and this this command:
../LicenseServer_start.sh"

Note

If you start the Server application like this, the script will check if the kernel drivers are compiled for the current kernel and running, and if not, it will try to compile and start them to make sure that Server application will be able to run.

3. Hit the following command from any folder: **"LicenseServer"**

Note

This will do the same as point #2, but from any folder.

Once the Server application is running, it will write to the console something like this:

```
Checking time functions
t0: 248725978581 -> t1: 248726984480, dt: 1005899
t2: 248725978582 -> t3: 248726984480, dt: 1005898
t4: 1649309382940272 -> t5: 1649309383955655, dt: 1015383
```

Log files:

Log files will be saved to `.\logs\` directory
[LSLOG] Current logfile: `.\logs\licsrv.20220407_072943_956.log` (20220407_072943_956)

```
[LSLOG] Clean logs in c:\Program Files\Adaptive Recognition\Common
Utils\LicenseServer\Server\logs (licsrv*.log)
[LSLOG] Current logfile: .\logs\lscons.20220407_072943_956.log (20220407_072943_956)
[LSLOG] Clean logs in c:\Program Files\Adaptive Recognition\Common
Utils\LicenseServer\Server\logs (lscons*.log)
[licsrv] Logging started
2022.04.07 7:29:43,956395 [11176] Main started
[lscons] Logging started
```

Searching for licenses:

```
2022.04.07 7:29:43,956507 Finding licenses type: 99900099, min. expiry date: 20220407 ...
2022.04.07 7:29:43,956761 Found license type: 99900099, licID: 111111, expiry date:
20230331
2022.04.07 7:29:43,956840 Query access to share licenses -> Locking(CARMEN Server license)
2022.04.07 7:29:43,958076 Finding licenses to share...
2022.04.07 7:29:43,958163 Found 1 devices
2022.04.07 7:29:43,958238 [01111111] XX licenses
2022.04.07 7:29:43,987310 nk: 21, nkf: 0
2022.04.07 7:29:43,987440 Found 21 license(s) to share.
2022.04.07 7:29:43,987647 Num. of license types shared: 4
    1affffff -> 4
    99902000 -> 1
    30ffffff -> 1
    16ffffff -> 4
```

Starting the License Server:

```
2022.04.07 7:29:43,988242 Starting License Server (v7.3.1.27 - License Server (rev:
d5944ac)) ...
2022.04.07 7:29:43,988353 Server running mode: Standalone
2022.04.07 7:29:43,988485 Server port: 8998
2022.04.07 7:29:43,988796 Health check port: 48315
2022.04.07 7:29:43,989069 Server status port: 8980
2022.04.07 7:29:43,989284 Max. clients: 20
2022.04.07 7:29:43,989442 idle time out: 600 sec
2022.04.07 7:29:43,989526 Lock time out: 3000 msec
2022.04.07 7:29:43,989681 Set NNC time out -> 2000 msec
2022.04.07 7:29:43,989875 NNC time out: 2 sec
2022.04.07 7:29:43,991339 Num of process / lock threads: 6 / 6
[LSLOG] Current logfile: .\logs\portcheck.20220407_072943_987.log (20220407_072943_987)
2022.04.07 7:29:43,992648 Wait for end of initialization
[LSLOG] Clean logs in c:\Program Files\Adaptive Recognition\Common
Utils\LicenseServer\Server\logs (portcheck*.log)
2022.04.07 7:29:43,993429 Threads initialization is SUCCESS. Init time: 0 msec
[portcheck] Logging started
```

Details about possible connections:

```
2022.04.07 7:29:43,994417 Server has created. IP: 0.0.0.0:8998 Max. client num.: 20
    192.168.6.175
    192.168.74.1
    192.168.109.1
Waiting for a connection (ESC to exit)...
2022.04.07 7:29:43,994799 [LICSRV]: Wait for event: -1
```

Note

If you would like to exit from the Server application, please press 'Esc' key on your keyboard.

2.2. SERVER APPLICATION AS A SERVICE

There is a possibility to use License Server as a service on Windows and Linux.

2.2.1. ON WINDOWS

Create and start the service with the following command: **LicenseServer_x64.exe -inst**

Once the service is created and started, it will be visible in Task Manager on the Services tab as 'licserver.' If you open Services from this tab, you will be able to stop and start the License Server service from here as well.

If you would like to stop and remove the License Server service, hit the following command:

LicenseServer_x64.exe -uninst

If you hit the above command, the License Server service will no longer be visible in Windows Services.

2.2.2. ON LINUX

It is possible to use License Server as a service on Linux if 'systemd' is available on the system.

- Enable: **systemctl enable licserverd.service**
- Start: **systemctl start licserverd.service**
- Get the current status about the service: **systemctl status licserverd.service**
- Stop: **systemctl stop licserverd.service**
- Disable: **systemctl disable licserverd.service**

Important!

CARMEN® has to be installed on the computer where the Server application is running.

2.3. SERVER APPLICATION FOR MORE THAN ONE CLIENT

This solution is available for free for one client only. To use it for more clients, a License Server license is necessary. This license allows the application to share any Adaptive Recognition licenses for more clients simultaneously.

If your **License Server license expires**, the application will **STOP working**. After you restart, only one client will be able to connect to the Server application.

If you run the License Server application as a service, license expiration will do the following:

- **On Windows:** the service is stopped. If you restart it, it will share the licenses for one client only.
- **On Linux:** the service restarts automatically as a service for one client only.

Note

If you would like to connect more than one client to the server, please contact your Sales Manager to check the possibilities.

3. CONFIGLSCLIENT

Once the Server application is running — either as a Console application or as a service — the ConfigLSClient application's configuration must be done as follows.

This application can be found in the following folder:

- On Windows: "C:\ProgramFiles\AdaptiveRecognition\CommonUtils\LicenseServer\Client\
"
- On Linux: "/opt/gx64/LicenseServer/Client/"

! Important!

If you want to take effect ConfigLSClient application, you must run it as an administrator or root, otherwise the application will not be able to save the changes into GXSD.DAT file!

- Set the Server address where the licenses should search for:
 - Only IP address:
Windows: **ConfigLSClient_x64.exe SET 192.168.0.1**
Linux: **./ConfigLSClient_x86_64.out SET 192.168.0.1**
 - IP address with port (in cases when, for example, the Server application port was changed from the default 8998 to 7683)
Windows: **ConfigLSClient_x64.exe SET 192.168.0.1:7683**
Linux: **./ConfigLSClient_x86_64.out SET 192.168.0.1:7683**

Note

The SET command will automatically ENABLE the usage of the License Server.

- Enable License Server on the client computer (if it is configured, but DISABLED):
Windows: **ConfigLSClient_x64.exe ENABLE**
Linux: **./ConfigLSClient_x86_64.out ENABLE**

Note

If License Server is enabled, licenses will always be looked for over the network, even if a HW Key with licenses is plugged into the client computer.

- Get the status of the License Server on the client computer:

Windows: **ConfigLSClient_x64.exe STATUS**

Linux: **./ConfigLSClient_x86_64.out STATUS**

The STATUS messages could be the follows:

- If the usage of License Server is ENABLED on the client computer where the STATUS request is sent:
 - The connection is OK. More clients can connect.
 - The connection is OK. No more clients can connect.
 - The connection cannot be established. It's switching off the using of the License Server. (It means the STATUS request will DISABLE the License Server usage in this case)
 - If the usage of License Server is DISABLED on the client computer where the STATUS request is sent:
 - The server is running. More clients can connect.
 - The server is running. NO more clients can connect!
 - The connection cannot be established.
- Set the client timeout from the default 10000 to, for example, 20000:
Windows: **ConfigLSClient_x64.exe TIMEOUT 20000**
Linux: **./ConfigLSClient_x86_64.out TIMEOUT 20000**
 - Disable License Server on the client computer:
Windows: **ConfigLSClient_x64.exe DISABLE**
Linux: **./ConfigLSClient_x86_64.out DISABLE**

 **Important!**

CARMEN® must be installed on the computer where the Client application is configured.

VIDEO SDK

1. INTRODUCTION

This is a CARMEN® Video SDK library for the CARMEN® ANPR (Automatic Number Plate Recognition) and MMR (Make & Model Recognition) engines. This library offers interfaces in C, C++ and C# for getting ANPR and MMR results from passing vehicles in camera stream or video file.

Note

The latest available version from Video SDK: **1.2.1**

2. ENGINE AND LICENSE REQUIREMENTS

This SDK utilizes CARMEN® engines either locally or through the CARMEN® Cloud, and these engines require licenses to operate.

There are four use cases available to suit different needs:

- **Local usage:** Engines are installed and run on the machine. Hardware key is connected to the same machine with the proper licences.
- **License Server usage:** Engines are installed and run on the local machine but obtain licensing from a License Server on another computer, which has the hardware key with the appropriate licenses connected.
- **Cloud licensing:** Engines are installed and run on the local machine but obtain licensing from an LSC (like in the case of the License Server). LSC can run locally or remotely, and connects to the CARMEN® Cloud using an API Key to access the required licenses.
- **Cloud processing:** Only the video decoding and image preselection components run on the machine with the Video SDK. Licensing for preselection is obtained directly from the CARMEN® Cloud. The ANPR and MMR engines run in the CARMEN® Cloud.

For more information on the available cloud-based options, please visit the CARMEN® Cloud website: <https://carmencloud.com/docs/content/tutorials/which-subscription>

For system design details, see the chapter „Carmen Licensing,“ For instructions on setting the licensing mode, refer to the chapter „[Licensing Modes](#)“.

CARMEN® Video SDK requires the CARMEN® ANPR package installed. This package already includes the CARMEN® ANPR world engine by default (see the [CARMEN® ANPR/OCR Installation Guide](#)). If you intend to run a regional ANPR engine (e.g., EUR, NAM, etc.) or a CARMEN® MMR engine locally, these components must be installed separately.

3. VIDEO SDK INSTALLATION

The Video SDK package is available directly from [Adaptive Recognition's documentation page](#):

It arrives in a zip file. The downloaded zip file contains two different folders:

- linux
- windows

3.1. INSTALLATION ON LINUX

In the "linux" folder in the zip package which you have downloaded you can find separate tar.gz files for the supported architectures. For example, the 1.2.0 version contains the following files:

Files in the package:

carmen_video_sdk-1.2.0.34322-linux-arm64-package.tar.gz

carmen_video_sdk-1.2.0.34322-linux-x86_64-package.tar.gz

After extracting the right package for your architecture, you will find all the necessary files for the installation of CARMEN® ANPR and CARMEN® Video SDK.

The Video SDK requires a minimum of 7.3.1.29 CARMEN® ANPR software installed. This version of CARMEN® can be found in the extracted package. If not already installed, the user must install this before installing the CARMEN® Video SDK, otherwise the installer will warn the user and exit with an error. For example, from the extracted package on Linux x86_64, the CARMEN® ANPR software can be installed by extracting "CARMEN_ANPR-7.3.1.29-x86_64.tar.gz" and running the "_install_all.sh" script inside.

To install CARMEN® Video SDK, you may run the install script "install-carmen_video_sdk.sh". Before running the script, ensure that only one tar.gz file with a filename like "carmen_video_sdk-*-linux-*.tar.gz" is present next to the install script (for example, in the case of x86_64, only carmen_video_sdk-1.2.0.34322-linux-x86_64.tar.gz).

To uninstall CARMEN® Video SDK, you may run the uninstall script "uninstall-carmen_video_sdk.sh".

After you install the CARMEN® Video SDK, the files are placed in the following directories:

- Header files:
 - o C headers: /usr/include/carmen_video_sdk/c
 - o C++ headers: /usr/include/carmen_video_sdk/cpp
- Shared object files:
 - o Located in: /usr/lib/carmen_video_sdk
- NuGet Package (for C#):
 - o Located in: /usr/include/carmen_video_sdk/csharp

Additional Information for Linux_x86_64 Users:

Some engines require the Intel "iclib" package. Therefore, the installer places its shared library files separately in /usr/lib. These files are not removed when you uninstall the Video SDK because they might be used by other applications on your system. If you wish to remove them manually, use the following commands:

```
sudo rm /usr/lib/libintcl.so.5
sudo rm /usr/lib/libsvml.so
sudo rm /usr/lib/libimf.so
```

3.2. INSTALLATION ON WINDOWS

In the "windows" folder in the zip file which you have downloaded you can find the installer package for Windows x86_64. After extracting this package, inside you can find the MSI installer and the sample programs for CARMEN® Video SDK. For example, the 1.2.0 version contains the following files:

Files in the package:

CARMEN_Video_SDK-1.2.0.34322.msi

carmen_video_sdk_samples-1.2.0.34322.zip

The MSI installer contains the software components required for running the Video SDK, but it does not include the CARMEN® ANPR software or any ANPR/MMR engines. The Video SDK requires a minimum of 7.3.1.29 CARMEN® ANPR package installed, which you must download and install separately on Windows. For the installation, see the [CARMEN® ANPR/OCR Installation Guide](#).

The install path is: "*C:\Program Files\Adaptive Recognition\CARMEN Video SDK*".

The Windows MSI installer sets this directory path to a system environment variable called "**CMV_INSTALL_DIR**". The built-in programs can use the necessary dynamic libraries from the SDK binary directory, which the installer also adds to the system PATH.

The sample programs are provided to the user in a separate package with all supported languages.

4. BUILD REQUIREMENTS

CARMEN® Video SDK 1.2.1 version supports both Linux and Windows. Binary compatibility between different versions is not guaranteed.

4.1. LINUX

The Video SDK provides C, C++, and C# interfaces. Its functionality is implemented in a core native library, compiled with GCC/Visual Studio for x86_64 and ARM 64-bit architectures.

4.1.1. C

Compiler	Target architecture	Target platform
Any Linux C compiler	x86_64, arm64	64 bit

To build it, you need to set the following:

1. Include directory is: `"/usr/include/carmen_video_sdk/c"`
2. Link directory is: `"/usr/lib/carmen_video_sdk"`
3. Link library name is: `carmenVideoSDK_c`

The C sample project's directory contains CMake files to be able to open the project with a wide range of IDEs.

4.1.2. C++

Compiler	Target architecture	Target platform	C++ standard
Any Linux C++ compiler, tested with GCC 9.4.0	x86_64, arm64	64 bit	C++17

This is a header-only interface over the C interface.

To build it, you need to set the following:

1. Include directory is: `"/usr/include/carmen_video_sdk/c"`,
`"/usr/include/carmen_video_sdk/cpp"`
2. Link directory is: `"/usr/lib/carmen_video_sdk"`
3. Link library name is: `carmenVideoSDK_c`

The C++ sample project's directory contains CMake files to be able to open the project with a wide range of IDEs.

4.1.3. C#

Target architecture	Target platform	Target framework
x86_64, arm64	64 bit	.NET 6.0

In order to use this library, you need to add a reference to the local NuGet package located in the `/usr/include/carmen_video_sdk/csharp` folder. There is a `nuget.linux.config` file in the solution folder; by renaming it to `NuGet.Config`, you can add the package source to your NuGet configuration.

The C# sample project's directory contains a VisualStudio 2019 solution file.

! Important!

Ensuring the Built Application Finds the Video SDK .so Files

In order for the built application to locate the Video SDK's shared object (.so) files, we have defined a method in the sample .csproj file. If you set the RuntimeIdentifier to *linux-x64* or *linux-arm64*, the build process will perform the following actions:

- **Copy the Necessary .so File:** After the build, it will automatically copy the required shared object file to the output directory.
- **Set the RPATH Using patchelf:** It will modify the runtime search path (rpath) of the copied shared object file using patchelf to ensure it can find the .so files at runtime.

 Note

Prerequisite: Please make sure that patchelf is installed on your system. You can install it using your distribution's package manager.

4.2. WINDOWS

The Video SDK provides C, C++, and C# interfaces. Its functionality is implemented in a core native library, compiled with Visual Studio 2019 for x86_64 architectures.

4.2.1. C

Compiler	Target architecture	Target platform
Any Windows C compiler	x86_64	64 bit

To build it, you need to set the following:

To access the installation folder, you can use this environment variable in CMake:

"\$ENV{CMV_INSTALL_DIR}", in VisualStudio: "\$({CMV_INSTALL_DIR})".

1. Include directory is: "\$ENV{CMV_INSTALL_DIR}\\sdk\\c\\include"
2. Link directory is: "\$ENV{CMV_INSTALL_DIR}\\sdk\\c\\lib"
3. Link library name is: carmenVideoSDK_c

The C sample project's directory contains a VisualStudio 2019 solution file, and CMake files to be able to open the project with a wide range of IDEs.

4.2.2. C++

Compiler	Target architecture	Target platform	C++ standard
Any Windows C++ compiler	x86_64	64 bit	C++17

This is a header-only interface over the C interface.

To build it, you need to set the following:

To access the installation folder, you can use this environment variable in CMake: `"$ENV{CMV_INSTALL_DIR}"`, in VisualStudio: `"$(CMV_INSTALL_DIR)"`.

1. Include directory is: `"$ENV{CMV_INSTALL_DIR}\\sdk\\C\\include ;$ENV{CMV_INSTALL_DIR}\\sdk\\C++\\include"`
2. Link directory is: `"$ENV{CMV_INSTALL_DIR}\\sdk\\C\\lib "`
3. Link library name is: `carmenVideoSDK_c`

The C++ sample project's directory contains a VisualStudio 2019 solution file, and CMake files to be able to open the project with a wide range of IDEs.

To use experimental features in C and C++, `CMV_ENABLE_EXPERIMENTAL_FEATURES` needs to be added as a compiler definition.

4.2.3. C#

Target architecture	Target platform	Target framework
x86_64	64 bit	.NET 6.0 .NET Framework 4.6

In order to use this library, you need to add a reference to the local NuGet package located in the `sdk/CSharp` folder. There is a `nuget.windows.config` file in the solution folder; by renaming it to `NuGet.Config`, you can add the package source to your NuGet configuration.

The C# sample project's directory contains a VisualStudio 2019 solution file.

5. VIDEO INPUT

In this chapter, we explain the types of acceptable video inputs. The most important input parameter of the CARMEN® Video SDK is the video file or network stream that it must process and return the passing vehicles as an event.

In order for CARMEN® Video SDK to work properly on the video/stream, it is important that the license plates of the vehicles passing by on the video meet the specifications in the [Imaging for Carmen®](#).

CARMEN® Video SDK includes an advanced vehicle-detection algorithm. As a result, your camera does not require an external hardware trigger to select frames from the video stream for **license-plate recognition**. The algorithm requires a moving vehicle (e.g., cars, buses, trucks) to be present in the video feed.

Processing works with recordings from both fixed-installation cameras and moving vehicles. It generally consumes less power in a fixed-installation setup.

It is important that the vehicle detector is partly based on text recognition, so the ROI must be in a way that no other non-license plate characters can be seen such as posters, signs or camera watermarks.

There are two main types of videos inputs: video file and network stream. Their processing is partially different.

5.1. NETWORK STREAM

Typically, RTSP or H264 streams coming from cameras are considered network streams. In case of such an input, the processor continuously tries to request the images from the stream, and selects the received images for ANPR. If too many images are selected and the system cannot perform ANPR fast enough, the buffer will fill up, increasing RAM usage and potentially causing some frames to be dropped. In such cases, unfortunately, it may happen that the license plate of a vehicle only appears in images that are discarded by the system, thus losing an event.

If the network stream is interrupted and autoreconnect is false (default), processing stops. To have the processor continuously attempt to reconnect until the stream is restored, set autoreconnect=true during initialization.

For events and frames, the timestamp is taken from the input stream when available; otherwise it is set to the frame's arrival time in the StreamProcessor.

Accepted protocols: HTTP, HTTPS and RTSP

Examples:

"rtsp://[USER]:[PASSWORD]@[IP]:[PORT]" -> rtsp://192.168.0.50/stream/jpeg

[http://\[URL\]](http://[URL]) -> <http://192.168.0.50:9901/video.mjpeg>

5.2. VIDEO FILE

In case of video file input, the operation is similar to the network stream, with the difference, that since the file is continuously available, if the buffer is full, do not discard the images, but wait until the ANPR makes room for the image in the buffer. In this case, the processing will slow down, but it will not lose the event because of this.

Note

When rendering video via the frame callback, **buffering and timestamp-synchronized playback are the responsibility of the application.** The SDK does not provide a display queue. Due to internal pipeline behavior, frames may arrive in bursts or with variable latency; without an application-level, timestamp-ordered buffer, playback can stutter or be interrupted when buffers fill.

In this case, the timestamp is read from the video file by the codec, which will start from 0 in most cases. The advantage of this is that in case of two runs, the same frame will receive the same timestamp. The disadvantage is that it is more difficult to put it in real time, if this is important, then it is worth adding the video's production time or its playback time to the timestamps of all frames and events.

Accepted formats: KV (H.264), MP4 (H.264), ASF (MPEG4), MJPEG, AVI (H.264)

Examples:

[file:\[path\]](#)

In case of relative path:

[file:videos/cars.mp4](#)

Absolute path on Linux:

file:/videos/cars.mp4

Absolute path on Windows:

file:C:/videos/cars.mp4

6. A MINIMAL APPLICATION EXAMPLE

This minimal application performs recognition locally and serves as a basic demonstration of how to use the Carmen Video SDK. It is similar to sample code "01_minimal" included with the SDK. For more detailed results, advanced settings, and additional use cases—including different engines and licensing modes—please refer to the other sample applications provided in the SDK package and discussed in later chapters.

6.1. CREATE ANPR

In order to run ANPR on a stream, you have to build an Anpr object, which should be added to the StreamBuilder:

6.1.1. C

```
CM_ANPR_BUILDER anprBuilder;  
cm_anprbuilder_create(&anprBuilder);  
  
CM_ANPR anpr;  
cm_anprbuilder_build(anprBuilder, &anpr);  
cm_anprbuilder_free(anprBuilder);
```

You will have to free this Anpr object later with:

```
cm_anpr_free(anpr);
```

6.1.2. C++

```
cm::anpr::Anpr anpr = cm::anpr::AnprBuilder()  
    .build();
```

6.1.3. C#

The AnprBuilder and Anpr classes implement IDisposable. To ensure that resources are released safely, use a *using* statement or a *using* declaration when working with AnprBuilder and Anpr objects.

```
using Anpr.AnprBuilder anprBuilder = Anpr.Builder();  
using Anpr anpr = anprBuilder.Build();
```

6.2. ONEVENTCALLBACK

This is a callback function, which is triggered when an Event is generated during stream processing. These examples show a few properties that are available in an Event object, find more details there: [RESULT CLASSES](#).

6.2.1. C

The callback parameter has to be of type `Event*` and at the end of the function has to free the event.

```
void onEventCallback(CM_EVENT* e, void* userdata) {
    printf("-----\n");
    printf("Plate: %s\n", e->vehicle->plate->text);
    printf("Country: %s\n", e->vehicle->plate->country);

    printf("\n");
    fflush(stdout);
    cm_event_free(e);
}
```

6.2.2. C++

In C++ the callback function parameter is a `const cm::Event&`. You should use *try-catch* block around your code.

```
void onEventCallback(const cm::Event& event) {
    try {
        std::cout << "-----" <<
std::endl;
        std::cout << "Plate text: " << event.vehicle().plate().text() <<
std::endl;
        std::cout << "Country: " << event.vehicle().plate().country() <<
std::endl;

        std::cout << std::endl;
    } catch(const std::exception& e) {
        std::cout << "Exception in event callback " << e.what() << std::endl;
    } catch(...) {
        std::cout << "Exception in event callback" << std::endl;
    }
}
```

6.2.3. C#

In C# callback the function parameter is a `Carmen.Event`, you should use try-catch block around your code. The `Event` class implements `IDisposable`. To ensure that resources are released safely, use a `using` statement or a `using` declaration when working with `Event` objects.

Be aware that the lifetime of `ImageProxy` objects is tied to the containing `Event` object. If you intend to use the image after the lifetime of the `Event` object, we recommend converting it into an `Image` object first. For the safe handling of `ImageProxy` and the nested classes of `Event`, refer to sample "03_result_details".

```
static void EventHandlerCallback(Event e)
{
    try
    {
        using (e)
        {
            Console.WriteLine("-----");
            Console.WriteLine("Plate text: " + e.Vehicle.Plate.Text);
            Console.WriteLine("Country: " + e.Vehicle.Plate.Country);
            Console.WriteLine();
        }
    }
    catch (Exception ex)
    {
        Console.Error.WriteLine("Exception in event callback. " + ex);
    }
}
```

6.3. CREATE STREAMPROCESSOR OBJECT

You also need to use builder pattern to create a StreamProcessor object. In these examples, there will be only a minimal implementation, for more options, see [STREAMPROCESSOR BUILDER](#).

6.3.1. C

```
cm_streamprocessorbuilder_set_source(builder, streamUrl);
cm_streamprocessorbuilder_set_region(builder, region);
cm_streamprocessorbuilder_set_name(builder, "Stream 1");
cm_streamprocessorbuilder_set_event_callback(builder, onEventCallback, NULL,
NULL);
cm_streamprocessorbuilder_set_anpr(builder, anpr);

CM_STREAM_PROCESSOR stream;
cm_streamprocessorbuilder_build(builder, &stream);
cm_streamprocessorbuilder_free(builder);
```

6.3.2. C++

```
cm::video::StreamProcessor stream = cm::video::StreamProcessorBuilder()
    .source(streamUrl)
    .region(region)
    .name("Stream 1")
    .eventCallback(onEventCallback)
    .anpr(anpr)
    .autoReconnect(true)
    .build();
```

6.3.3. C#

The StreamProcessorBuilder and StreamProcessor classes implement IDisposable. To ensure that resources are released safely, use a *using* statement or a *using* declaration when working with StreamProcessorBuilder and StreamProcessor objects.

```
using StreamProcessor.StreamProcessorBuilder streamProcessorBuilder =
StreamProcessor.Builder();
using StreamProcessor stream = streamProcessorBuilder
    .Source(streamUrl)
    .Region(region)
    .Name("Stream 1")
    .Anpr(anpr)
    .EventCallback(EventHandlerCallback)
    .Build();
```

6.4. START STREAMPROCESSING

After you create the StreamProcessor object, you can Start and Stop it. Start function starts the stream processing asynchronously. You have to keep your main thread alive. In these examples we wait for a 'q' character input from the user. When you want to end processing, you have to call the Stop function.

6.4.1. C

In C you have to free the stream processor object.

```
cm_streamprocessor_start(stream);  
while(getc(stdin) != 'q');  
cm_streamprocessor_stop(stream);  
cm_streamprocessor_free(stream);
```

6.4.2. C++

```
stream.start();  
while(std::cin.get() != 'q');  
stream.stop();
```

6.4.3. C#

```
stream.Start();  
while (Console.ReadKey().KeyChar != 'q');  
stream.Stop();
```

7. CONSTRUCTING COMPONENTS

The following builders are present in the sample codes:

- AnprBuilder -> Anpr
- MmrBuilder -> Mmr
- AdaptiveRecognitionCloudBuilder -> AdaptiveRecognitionCloud
- StreamProcessorBuilder -> StreamProcessor

Please refer sample codes "01_minimal" and "02_basic" for simple implementation examples to get started, while sample "03_result_details" has more a detailed structure.

7.1. ANPR BUILDER

7.1.1. TYPE

It sets the type of ANPR. Currently there are two options:

- LOCAL: local ANPR processing with CARMEN® FreeFlow engines
- LOCAL_GO: local ANPR processing with CARMEN® Go engines

Default value is LOCAL.

7.1.2. LOCAL CONCURRENCY LIMIT

This allows you to set how many ANPR threads can be run via the given ANPR class. It is recommended to set it to the same value as the number of ANPR Core Licenses, but this can also be used to distribute resources in the case of multiple streams.

Default value: 1

7.1.3. REGISTERING ANPR PROFILE TO AN ALREADY BUILT ANPR OBJECT

Once you have built an Anpr object with an Anpr Builder *Build* function, it provides a *registerProfile* method, which allows you to define custom ANPR profiles. An ANPR profile determines how license plate recognition is performed, including regions, engine versions, locations, and custom properties.

Options for Registering Profiles:

1. AnprStage

- **Purpose:** Allows you to add a single ANPR stage with specific settings.
- **Configuration Options:**
 - **ANPR Region:** Set the region from Region List for the recognition process.
 - **Engine Version:** Specify the version of the ANPR engine to use.
 - **Location:** Define the engine *location* property.
 - **Custom Properties:** Set additional engine properties as needed.
- **Usage:** Ideal for simple recognition tasks requiring a single stage.

2. AnprProfile

- **Purpose:** Allows you to create a complex ANPR profile containing multiple stages and custom logic.
- **Components:**
 - **Stages:** One or more AnprStage objects defining each recognition step.
 - **Callback Function:** A user-defined function that dictates the logic for transitioning between stages. This function can be used to refine results or apply conditional processing.
- **Usage:** Suitable for advanced recognition workflows that require multiple stages or custom logic.

Return Value:

- The *registerProfile* method returns an **AnprProfileId**, a unique identifier for the registered profile. This ID can be used later to assign the profile to a Stream Processor.

Example Usage:

- **Sample "07_multistage_anpr"** demonstrates how to register an ANPR profile with multiple stages and custom logic. Refer to this sample for practical implementation details.

7.2. MMR BUILDER

7.2.1. TYPE

It sets the type of MMR, currently it can only be LOCAL.

Default value is LOCAL.

7.2.2. REGISTERING MMR PROFILE TO AN ALREADY BUILT MMR OBJECT

It is similar to the Anpr object. Once you have built an Mmr object withan Mmr Builder *Build* function, it provides a registerProfile method, which allows you to define custom MMR profiles.

Profile Configuration:

- **mmrGroupName:** Specifies the MMR group name, such as mmr-eur, to choose the MMR engine.
- **Engine Version:** Specifies the version of the MMR engine to use.

Return Value:

- The registerProfile method returns an **MmrProfileId**, a unique identifier for the registered profile. This ID can be used later to assign the profile to a Stream Processor.

Example Usage:

- **Sample "07_multistage_anpr"** also includes examples of registering MMR profiles. Refer to this sample to see how ANPR and MMR profiles can be used.

7.3. ADAPTIVE RECOGNITION CLOUD BUILDER

This solution allows you to run ANPR (Automatic Number Plate Recognition) and MMR (Make & Model Recognition) engines in the cloud instead of on your local device. This solution reduces local resource consumption and eliminates the need for a hardware key. To use this service, you must register with the CARMEN® Cloud to obtain an API Key.

7.3.1. API KEY

An API Key is required to authenticate your application with the CARMEN® Cloud services. You must set the API Key in the Cloud Builder.

7.3.2. THE BUILT CLOUD OBJECT

Once you have configured the Cloud Builder, you can build the Cloud object. This object must be set in the StreamProcessorBuilder using the *cloud setter* to enable cloud-based ANPR and MMR processing instead of local processing.

The Cloud object provides `getCountries` method that returns the possible options for the region and location parameters of the StreamProcessorBuilder.

7.4. STREAMPROCESSOR BUILDER

7.4.1. SOURCE

The source setter determines the input video stream or file that needs to be processed by the CARMEN® Video SDK. There are two options for specifying the source:

Using a Source URL

The string variable determines the input video stream or file which needs to be processed. For files, use: "[file:\[path\]](#)". For stream, use: "[rtsp://\[USER\]:\[PASSWD\]@\[IP\]:\[PORT\]](#)" or "[http://\[URL\]](#)".

No default value, has to be set.

About CARMEN® Video SDK input source types there is a detailed description in [VIDEO INPUT](#).

Using a StreamFactory for Custom Image Callback

This mode allows you to provide a custom StreamFactory object to the SDK. By implementing specific functions, you can control how the SDK retrieves images, which is useful for custom or non-standard input sources.

Purpose:

- Allows integration with custom video capture hardware.
- Enables processing of pre-captured frames or images from memory.
- Facilitates integration with other media frameworks or custom pipelines.

For an example implementation, please refer to the "06_custom_stream" sample in C++ and C# languages.

It's important to set *processingMode* when using StreamFactory as source.

This is an *experimental feature*.

7.4.2. NAME

This variable determines the stream name.

Default value is: "Untitled".

7.4.3. ANPR

It determines which ANPR resource is used for license plate recognition. An ANPR object is built with [ANPR BUILDER](#).

Stream has to have an ANPR object or a Cloud object (see below) in order to be able to work.

7.4.4. MMR

It determines which MMR resource is used for Make&Model recognition. An MMR object is built with [MMR](#).

MMR parameter is optional. If not set, the stream processor will not recognize make, model and color.

7.4.5. CLOUD

It determines which Cloud resource is used for license plate recognition. A Cloud object is built with [ADAPTIVE RECOGNITION CLOUD BUILDER](#).

Stream has to have a Cloud object or an ANPR object (see above) in order to be able to work.

7.4.6. REGION

The region setter in the Stream Processor Builder allows you to specify a region code that determines which ANPR and MMR engines are used for processing. This provides a quick and convenient way to configure the recognition engines without the need to set up profiles explicitly.

The region code directly determines the ANPR and MMR engines used. It simplifies the setup by automatically selecting the appropriate engines based on the specified region.

In the 1.1.0 versions of the SDK, setting the region was mandatory.

For MMR, if the engine for the specified region was not installed, it defaulted to using the generic (GEN) engine.

In the 1.2.0 version with the introduction of [AnprProfile](#), you now have other option for configuring recognition engines.

When using Adaptive Recognition Cloud, setting the region is mandatory.

7.4.7. LOCATION

Sets the ANPR engine location parameter from the [ANPR reference manual](#).

Do not use the location parameter if you are using an AnprProfileId. When using profiles, the location should be specified within the profile itself.

7.4.8. ANPR PROFILE ID AND MMR PROFILE ID

The AnprProfileId and MmrProfileId setters allow for more complex configurations, such as multiple ANPR stages, custom engine parameters, and specific engine versions.

You should assign to the Stream Processor Builder the **AnprProfileId** that you obtained from the same ANPR object's registerProfile method as set for Anpr resource.

These settings are ideal when you need fine-grained control over the recognition process.

When you assign ANPR Profile Id to the Stream Processor, do not set region and location parameter.

7.4.9. AUTO RECONNECTION TO STREAM

If this parameter is set to true, then the stream automatically restarts in case of connection interruptions.

Default value is *false*.

7.4.10. ANPR COLOR RECOGNITION

This parameter controls the number plate color (text color, strip color, dedicated area color) recognition feature of the ANPR engine. If set to false, there will be no color information in the result. Default value is *true*.

7.4.11. MMR COLOR RECOGNITION

This parameter enables or disables the vehicle color recognition feature of the MMR engine. If set to false, there will be no color information in the result.

Default parameter is *true*.

7.4.12. EVENT DUPLICATION TIMEOUT

This parameter determines how much time is needed to pass between two identical results to be considered separate events. Identical results generated within this timeframe will be disregarded. (Think of a slowly moving vehicle, with the LP partially getting covered in the flow of traffic sometimes. This LP can show up as separate events, however it is just one event in reality. On the other hand, if the vehicle takes a turn and passes again, that is really a separate event, which should be recorded.)

Default parameter is *600 000* ($600\ 000ms = 600s = 10min$).

7.4.13. ROI (REGION OF INTEREST)

The ROI defines a **polygonal** area in image coordinates. **Concave polygons are supported.**

Plate triggering occurs **only** for detections **inside** the ROI; plates outside the ROI do not generate events. **MMR** analyzes the **entire frame**, but results are **gated by plate detection** within the ROI. A **smaller ROI** generally reduces processing time.

API / format

- The ROI is specified as an array of **N points** ($N \geq 4$).
- Points may be listed in **clockwise or counter-clockwise** order; the polygon is **implicitly closed**.
- Coordinates are **normalized** to **[0.0, 1.0]**.
 - **(0.0, 0.0)** corresponds to pixel **(0, 0)** (top-left corner).
 - **(1.0, 1.0)** corresponds to pixel **(image.width - 1, image.height - 1)** (bottom-right corner).
- The polygon must be **simple (non-self-intersecting)** — no bow-tie / crossing edges.

Default

- By default, the ROI covers the **full frame**: [(0, 0), (1, 0), (1, 1), (0, 1)].

Example of a setting:



ROI is: [(0.39271, 0.174946), (0.653949, 0.157667), (0.701337, 0.99568), (0.242041, 0.965443)]

7.4.14. EVENT CALLBACK

This function is called when an event was generated in video processing. This function is called in asynchronous mode.

Examples can be found in section ONEVENTCALLBACK.

If you don't set an event callback, the program won't call any function when an event occurs.

7.4.15. ON FRAME CALLBACK

This function is called after a frame from the source is decoded. This function is called in asynchronous mode.

If you don't set an "onFrame" callback, the program won't call any function when a frame is decoded.

7.4.16. STATUS CHANGE CALLBACK

This function is called when the video processor status is changing.

If you don't set a status change callback, the program won't call any function.

Possible stream status parameters are:

```
enum class StreamProcessorStatus {
    IDLE,
    RUNNING,
    STOPPING,
    FAILURE,
    FINISHED
};
```

Callback function parameter is the changing Stream object and the new status of it.

7.4.17. PROCESSING MODE

The CARMEN® Video SDK provides a `processingMode` setting that allows you to control how the stream processor handles incoming video streams or files. The `StreamProcessorMode` enum class defines the available modes:

```
enum class StreamProcessorMode {
    Auto,
    LiveStream,
    NonLiveStream
};
```

- **Auto:** (Default) The SDK automatically determines the processing mode based on the source URL. If the URL starts with "file:", it will use `NonLiveStream` mode; otherwise, it will use `LiveStream` mode.
- **LiveStream:** Intended for real-time streaming sources (e.g., live camera feeds). In this mode:
 - Buffers can drop images if they overflow, which helps maintain real-time processing by discarding frames when the system is under heavy load.
 - Suitable for applications where staying current is more important than processing every single frame.
- **NonLiveStream:** Designed for processing video files. In this mode:
 - If buffers are full, the processor will not drop frames, ensuring that all frames are processed.
 - The video processing speed is limited by the hardware capabilities, not by the video's FPS.
 - With **Go licenses**, this mode is limited to **30 FPS**.
 - Ideal for offline processing where completeness is required.

7.5. LOGGER

The inner library of the CARMEN® Video SDK makes logs. There are 5 log levels: *Debug*, *Info*, *Warning*, *Error*, *Critical*. The default log level is *Warning*. The default logging method is to log to the standard output: "[\${LOG_LEVEL}]: \${MESSAGE}".

7.5.1. CHANGING LOGGING LEVEL

There is an example of setting log levels in the „2_basic“ and „3_result_details“ sample codes in each program language.

7.5.2. SET LOGGING CALLBACK

There is an example of changing log callback in the 3rd sample codes in each program language. Set logger callback only before initializing any Video SDK class!

In this callback, there is an option to write logs to file or disable the logger.

7.6. LICENSING

The CARMEN® Video SDK utilizes a flexible licensing system that can be configured according to your deployment needs. By default, GX reads the licensing mode from the gxsd.dat file. However, you can modify the licensing mode directly within your application code to suit different scenarios.

For example implementations, refer to sample "05_cloud" to see how cloud licensing is set up. You can also adjust any of the other sample applications to use License Server mode if you wish to connect to a License Server or LSC while experimenting with these samples.

7.6.1. LICENSING MODES

The SDK supports multiple licensing types, and you can retrieve the current licensing mode using the `getCurrentLicensingType()` function. This function returns an enum value representing the licensing type:

```
enum class LicensingType {
    Unknown,
    Local,
    LicenseServer,
    CloudNNC
};
```

- **Unknown:** The licensing type could not be determined.
- **Local:** Licensing is managed locally with a hardware key connected to the machine.
- **LicenseServer:** Licensing is managed through a License Server or LSC within your LAN or WAN.
- **CloudNNC:** Licensing for the PlateFinder preselector is managed via the CARMEN® Cloud using an API key.

7.6.2. SETTING THE LICENSING MODE

You can set the licensing mode programmatically using the provided setter functions. This allows you to override the default settings from `gxsd.dat` and configure the licensing mode that best fits your application.

- **Set Local Licensing**

Use this function to configure the SDK to use local licensing with a hardware key connected to the machine:

```
void setLocalLicensing()
```

- **Set License Server**

Use this function to connect to a License Server or LSC within your network:

```
void setLicenseServer(const std::string& host, uint16_t port)
```

- host: The IP address or hostname of the License Server.
- port: The port number on which the License Server is running.

- **Set Local Licensing**

Use this function to configure cloud-based licensing using an API key:

```
void setCloudNNC(const std::string& apiKey)
```

- apiKey: Your CARMEN® Cloud API key for authentication.

8. RESULT CLASSES

The CARMEN® Video SDK provides several result classes to represent data extracted from video processing, including events for each vehicle captured. Events includes related information, such as license plate details, vehicle attributes, and relevant images.

8.1. IMAGE CLASSES

8.1.1. IMAGE

The **Image** class represents a bitmap image in memory. In the C and C++ SDKs, it is provided as a class. In C#, users should utilize the corresponding image handling classes provided by the .NET framework or the SDK's specific implementation.

8.1.2. IMAGEPROXY

The **ImageProxy** class is an image wrapper used in the SDK's callback parameters. It provides access to image metadata and allows the retrieval of image data through cloning.

8.2. ANPR (AUTOMATIC NUMBER PLATE RECOGNITION) CLASSES

8.2.1. PLATE

The **Plate** class represents the result of a license plate recognition.

8.2.2. PLATEDETECTION

The **PlateDetection** class contains information related to the license plate recognition process on an image.

8.3. MMR (MAKE & MODEL RECOGNITION) CLASSES

8.3.1. MMRDATA

The **MmrData** class contains attributes of a vehicle as recognized by the Make & Model Recognition process.

8.3.2. MMRDETECTION

The **MmrDetection** class contains information related to the Make & Model Recognition process on an image.

8.4. EVENT CLASSES

8.4.1. PLATEONIMAGE

The **PlateOnImage** class contains information related to the license plate recognition process on an image, along with a reference to that image.

Fields:

- **detection:** (*PlateDetection*)
The properties of the recognized license plate.
- **image:** (*ImageProxy*)
A reference to the image on which the license plate was detected.

8.4.2. MMRONIMAGE

The **MmrOnImage** class contains information related to the Make & Model Recognition process on an image, along with a reference to that image.

Fields:

- **detection:** (*MmrDetection*)
The Make & Model recognition result.
- **image:** (*ImageProxy*)
A reference to the image on which the vehicle attributes were detected.

8.4.3. VEHICLE

The **Vehicle** class represents a vehicle descriptor, which may include license plate information and vehicle attributes.

Fields:

- **plate:** (*optional, Plate*)
The license plate of the vehicle. This may be null if no license plate was detected.
- **mmrData:** (*optional, MmrData*)
The attributes of the vehicle as determined by the Make & Model Recognition process. This may be null if MMR was not performed.

8.4.4. EVENT

The **Event** class represents the result of processing a single vehicle passage. It aggregates all relevant information, including detected license plates, vehicle attributes, and associated images. An event may include multiple detections if the vehicle was captured in multiple frames or from multiple viewpoints.

9. REGION LIST

Region name	Region code
ARABIC	ARAB
Australia	AUS
Bangladesh	BGD
Caribbean	CAR
Central America	CAM
Central Asia	CAS
East Asia	EAS
Europe	EUR
General Latin	GEN
India	IND
Indonesia-Timor-Papua	ITP
Inside Asia	IAS
Iran	IRN
Iraq	IRQ
Israel	ISR
Japan	JPN
Nepal	NPL
New Zealand	NZL
North Africa	NAF
North America	NAM
Pacific	PAC
Pakistan	PAK
Philippines	PHL
South America	SAM
South Asia	SAS
Southern Africa	SAF
Taiwan	TWN
Turkey	TUR
Vietnam	VNM



10. KNOWN ISSUES

- All C functions return with error code -1 if error occurs inside. Later there will be more dedicated error codes
- **Memory Usage Increase During Overload Conditions:** When the system is overloaded, the process's memory usage may grow beyond expectations and not decrease afterward due to FFmpeg's buffer handling, where internal buffers are not freed until the StreamProcessor object is destroyed.



ADI DEMO

1. INTRODUCTION

The ANPR Demo for Images (ADI) is a program that was developed by Adaptive Recognition to serve as a simple and versatile tool for evaluating our core LPR technology before having to first develop an application.

The goal of the SDK is to allow you to develop a similar or even a more complicated application based on the Carmen API in order to meet the exact requirements of the particular project where it will be deployed.

The demo can handle both a single image as well as an image directory as an input source.

Supported image formats: **.jpg, .jpeg, .png, .bmp, .jp2**

This application can be found in the following folder:

- On Windows: "c:\Program Files\Adaptive Recognition\CARMEN softwares\Demos\ADI"
- On Linux: "/opt/gx64/ADI_Demo/"

Note

The latest available version from ANPR Demo for Images: **7.4.1.1**

Not all images are adequate for ANPR, the input images must meet a specific set of criteria for the engine to be able to recognize them. Please study the following document to learn more about these requirements: [Imaging for Carmen®](#).

2. MAIN SCREEN

After starting the demo, you will be presented with the main screen of the application. This window is split into three separate sections; the **Result Image**, the **Result Tabs** and the **Log** sections. All three sections provide information in connection with the scanned image.

Note

The application puts a red frame around the image area, where it has located a license plate and turquoise frame around the vehicle (only in case of running MMR).

#	Filename	Plate Text	Plate Category	Country / State	ANPR Time (ms)	Make & Model	Category	Color	Viewpoint	MMR Warning	MMR Time (ms)						
1	ARAB_01.jpg	663B5M	99%	COMMON	OMN	100%	157	Hyundai Elantra	99%	Car	98%	BLUE	99%	Rear-side	98%	-	160
2	CAM_01.jpg	P564JWV	98%		GTM	98%	160	Mazda CX-5 - CGI	100%	Car	100%	RED	99%	Front	99%	-	111
3	CAM_02.jpg	P726GKJ	98%		GTM	100%	171	Jeep Compass	95%	Car	99%	GRAY	88%	Rear-side	99%	-	175
4	CAM_03.jpg	M356139	98%		NIC	100%	139	Toyota Hilux	100%	Pick-up	100%	WHITE	99%	Front	99%	-	130
5	CAS_01.jpg	01M002402	98%		UZB	95%	215	Chevrolet Suburban ~ Tahoe	94%	Car	99%	WHITE	76%	Front	99%	-	155
6	EUR_02.jpg	3483FPK	99%	MIXED	ESP	99%	93		99%	Car	99%	GRAY	40%	Front	99%	-	136

In the title you can see the following information:

 ADI Demo 7.4.1.1 (64 bit) [ANPR: cmanpr-7.3.18.12 : world] - [MMR: mmr-7.3.4.100 : mmr-eur] - [License: IP NNC (10.0.6.65:8998)]

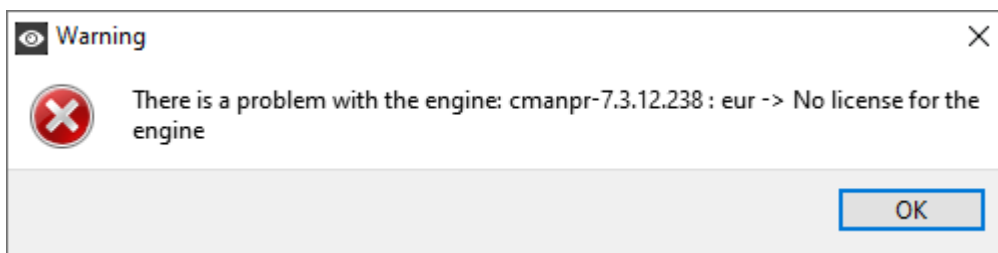
ANPR: the currently used ANPR engine region and version

MMR: the currently used MMR engine region and version

License: showing where CARMEN® is looking for the licenses (local/network)

 Note

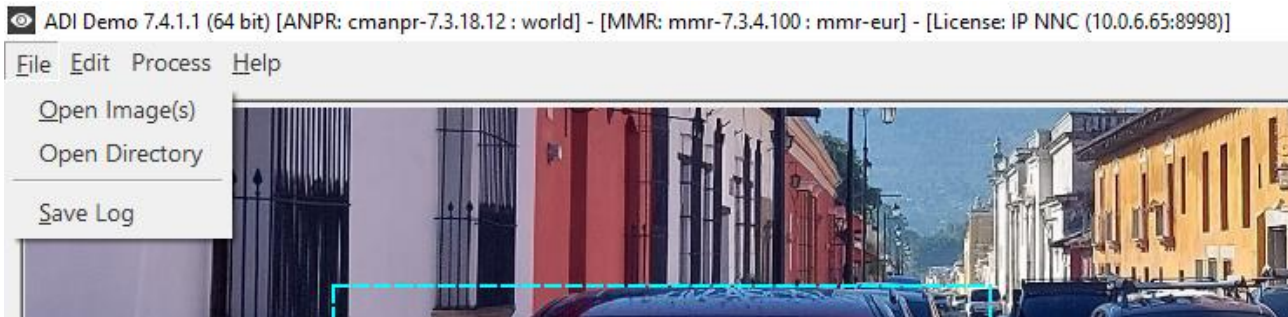
In case of missing hardware key, the following error message will appear:



3. FILE MENU

The ADI menu bar consists of four menu buttons, located in the upper left corner of the program window as follows:

File, Edit, Process, and Help.



The **File** menu contains the following menu items:

3.1. OPEN IMAGE(S)

Click to select specific images for LPR processing.

3.2. OPEN DIRECTORY

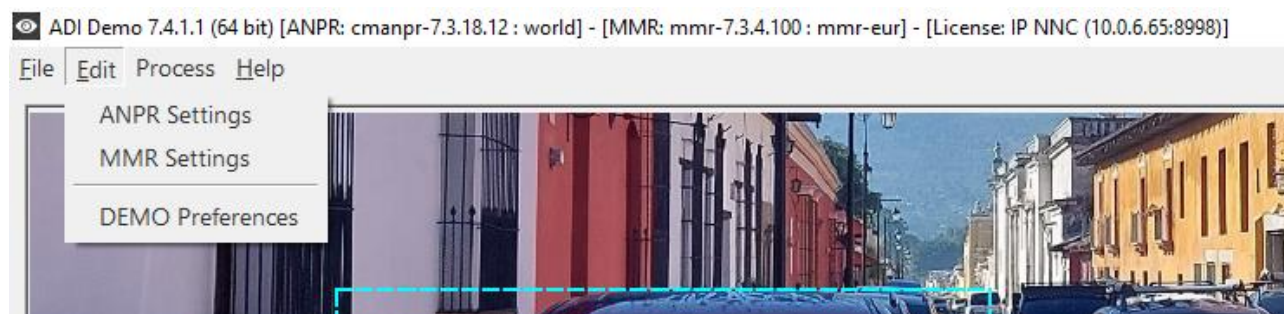
Click to specify a directory containing images for LPR processing.

3.3. SAVE LOG

Click to save the results log manually. The log can also be saved automatically by adjusting the Demo Preferences under the Edit menu.

4. EDIT MENU

The **Edit** menu contains the following menu items:

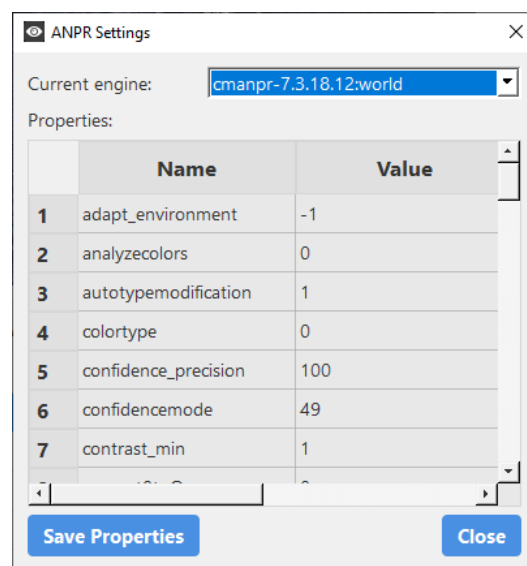


4.1. ANPR / MMR SETTINGS

Contains various configuration options related to the ANPR / MMR process. The engine used for processing is also selected here. By clicking this menu item, the following pop-up window will appear:

Engine: Click the drop-down menu to select an engine from the list of installed engines.

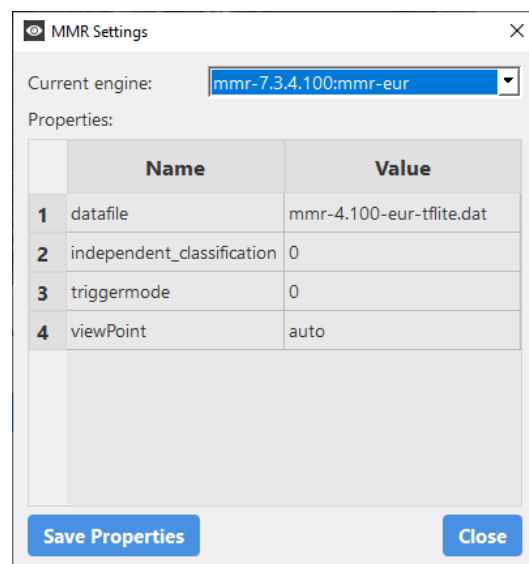
Properties: Lists the property values for the selected engine. To adjust the pre-set values, click on one of the value fields and then input a desired value. A full description of the engine properties can be found in the [CARMEN® ANPR Reference Manual](#).



ANPR Settings

Click on the **[Close]** button to apply your changes. These settings will remain active until the application is closed or until the engine in use is replaced with another one from the drop-down list. When you exit the application, the changes will be lost and the values will revert to the ones saved in the `gxsd.dat` configuration file.

Clicking on the **[Save Properties]** button will write the current values of each property into the `gxsd.dat` configuration file. This feature requires write permissions to the `gxsd.dat` file. The default values of an engine can be reset by reinstalling (uninstalling and installing) it, check it [here](#) how to do that. This function works only with ANPR properties as the MMR properties are read only.



MMR Settings

4.2. USING ADI DEMO WITH THE CLOUD ENGINE

ADI Demo can also operate in **cloud processing mode** using the `cmanpr-cloud` engine. In this mode, there is **no need for a hardware key, License Server, or LSC**. The regional ANPR and MMR engines do not run locally — the recognition takes place entirely in the **CARMEN® Cloud**. Communication with the cloud service is performed through an **API key**.

To generate an API key, you must register on the **CARMEN® Cloud website** at (<https://carmencloud.com/>) and subscribe to the **Carmen® Recognition Service**.

For more information on the available cloud-based options, please visit the CARMEN® Cloud website: <https://carmencloud.com/docs/content/tutorials/which-subscription>

The cloud engine is included in the **CARMEN® ANPR software package starting from version 7.3.1.27**, or the latest `cmanpr-cloud` engine can also be downloaded from [here](#).

4.3. CONFIGURING THE CLOUD ENGINE IN ADI DEMO

To configure cloud-based processing in ADI Demo:

1. In **ANPR Settings**, select *cmanpr-cloud* from the list of installed engines.
2. In the **Properties** panel:
 - o Set the **api_key** to the valid API key generated in CARMEN® Cloud.
 - o Select a **region** from the drop-down menu (this property is mandatory).
 - o Optionally select a **location**.
 - o The default value of the **server** property is the Vehicle API endpoint of CARMEN® Cloud: <https://api.carmencloud.com/vehicle> (no modification is required).

ANPR Settings

Current engine: cmanpr-7.3.1.3:cloud

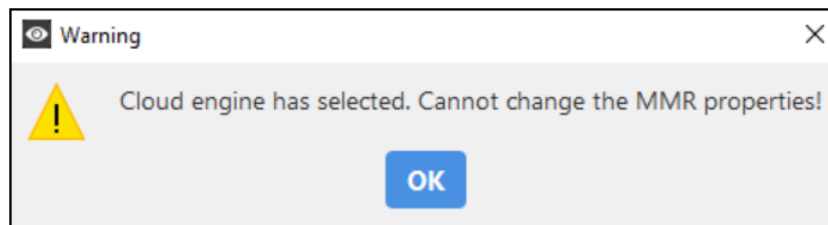
Properties:

	Name	Value
1	api_key	abcdefghijklmnopqrstuvwxy
2	autommr	1
3	datafile	cmanpr-1.3-cloud.dat
4	location	Hungary
5	maxreads	1
6	region	Europe
7	server	https://api.carmencloud.com/vehicle
8	timeout	10000

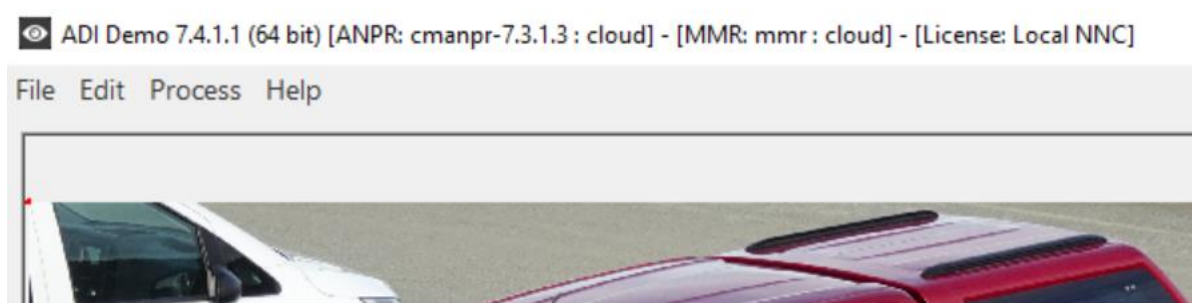
Save Properties Close

The **autommr** property does not need to be set manually. Its value is determined automatically based on whether **Read MMR automatically** is enabled in **DEMO PREFERENCES** (see section 4.4). If automatic MMR reading is enabled, the value of *autommr* will be **1**, otherwise **0**.

When using the *cmanpr-cloud* engine, and automatic MMR reading is enabled, **MMR recognition also happens in the cloud**, so a locally installed MMR engine is not required. In this case, you do not need to configure anything in **MMR Settings**. If you click on MMR Settings, the application will display a warning message indicating this.



If the cloud engine properties were set correctly, the status bar will display the following information.



4.4. DEMO PREFERENCES

Contains various configuration options related to the demo software.

By clicking this menu item, the following pop-up window will appear:



Auto-Log:

If checked, the results will be logged automatically.



Find All Plates:

If checked, the engine will search for all plates in the image that conform to the set parameters.



Find Empty ADR

If checked, the engine will search for the Empty ADR plates as well.



Loop (toggle:L):

If checked, the program continuously repeats processing of the specified image sequence/folder.



Process Sample Image At Start-Up:

If checked, the engine will process the specified sample image right after application start-up.



Real plate frame drawing

If checked, the real plate will be framed.



Show ROI

If checked, the set ROI will be visible on the image.



Show ROU

If checked, the set ROU will be visible on the image.

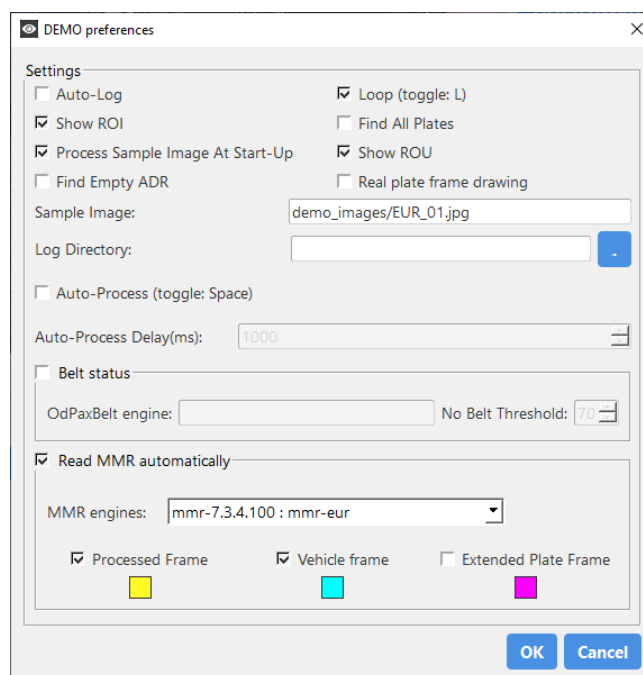
Sample Image:

This field shows the file that is loaded into ADI upon starting the application.

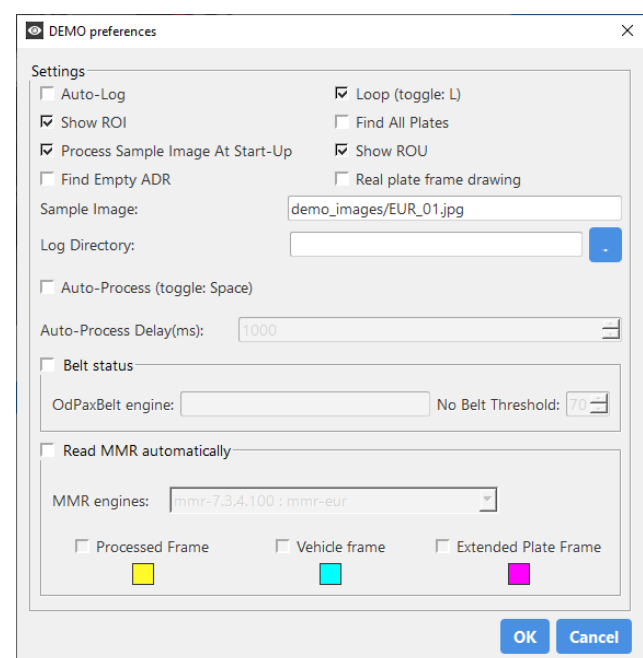
Log Directory:

Displays the directory, where the log file will be saved. Click [...] button to specify another directory.

The default folder is: "installation folder / logs"



MMR engine is installed



MMR engine is not active



Auto-Process (toggle: Space):

If checked, the application runs through the images and do ANPR on them automatically.

 **Note**

If 'Auto-Process' is on, you will not be able to examine the individual results, even if there is no more image to processing. You must set it back to 'Manual'. You can do it by clicking on the image and then pressing the 'space' on the keyboard.

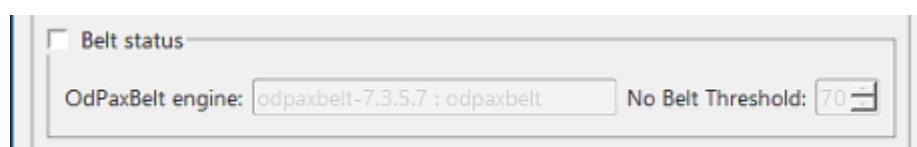
Auto-Process Delay(ms):

Type or use the arrows to adjust the amount of delay (in milliseconds) before the next image is loaded during Auto-Process.

 Belt Status:

- A new **"Belt status"** section appears when an OdPaxBelt engine is installed.
- The **engine version** are shown in the text field (e.g., odpaxbelt-7.3.5.7 : odpaxbelt).
- The **"No Belt Threshold"** slider allows users to define a detection threshold for vehicles without seat belts.

Make sure to select an appropriate OdPaxBelt engine version for proper functionality.



Read MMR automatically: (can be checked only in case if you have any MMR engine installed).

If checked, ADI demo will automatically return the MMR results.

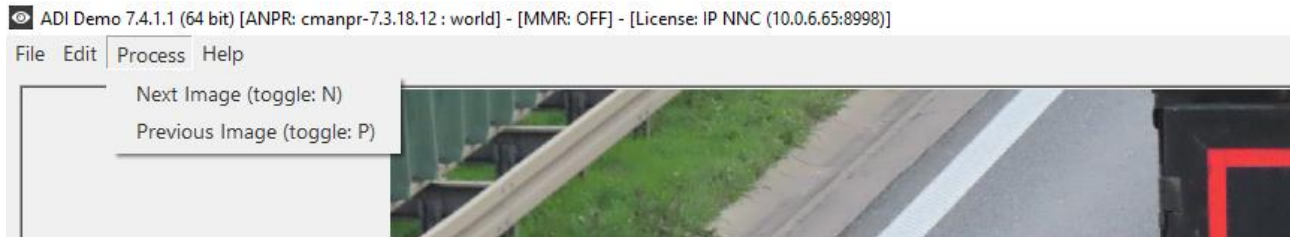
MMR frame:

- **vehicle frame:** the frame enclosing the entire vehicle.
- **extended plate frame:** the frame enclosing a broader area around the license plate.
- **processed frame:** This frame marks the vehicle, processed by the engine. (extended plate frame or in its absence a vehicle frame).



5. PROCESS MENU

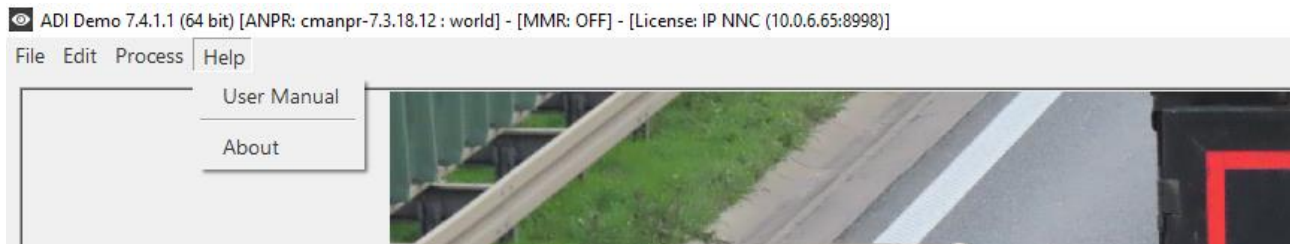
The **Process** menu contains the following menu items:



Click **Next Image (N)** or **Previous Image (P)** to navigate between the images. Alternatively, you may also use the N/P keys on the keyboard or simply click once on the image.

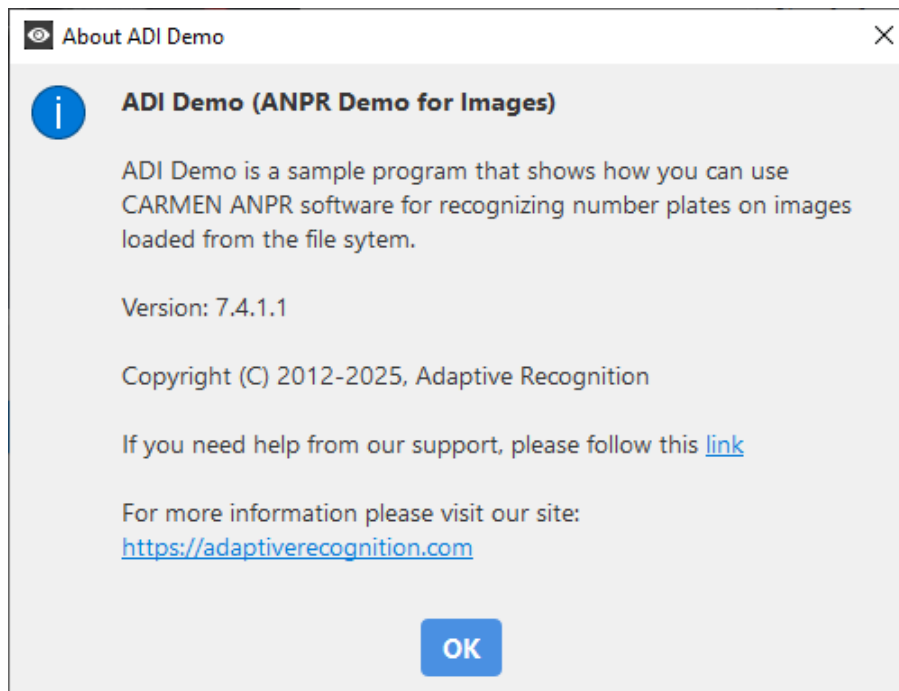
6. HELP MENU

The **Help** menu contains the following menu items:



User Manual: Opens this User Manual.

About: Provides the license, version, and copyright information for the installed application.



7. SET ROI/ROU ON THE IMAGE

From CARMEN® 7.3.1.27 there is a possibility to set more ROI/ROU polygons on the image which is loaded to ADI Demo application.

Steps:

- **ROI:** Press CTRL+I
- Start clicking the points of the polygon, if you are ready, press CTRL + R to save ROI



In written form: 22,731;1468,893;922,2047;29,2047

- If you would like to add more polygons, press CTRL+I again



In written form: 22,731;1468,893;922,2047;29,2047 + 2036,915;2546,2079;3783,2105;3671,900



- **ROU:** Press CTRL+U
- Start clicking the points of the polygon, if you are ready, press CTRL + R to save ROI



In written form: 7,7;11,1345;3826,1447;3823,40

- If you would like to add more polygons, press CTRL+U again



In written form: 7,7;11,1345;3826,1447;3823,40 + 22,2014;3815,2177;3823,2622;18,2622

This is how it looks like if you are using ROI and ROU at the same time:



 Note

ROU is the stronger property so if there is ROI and ROU on the same part of the image. The engine will not search on that area!

 Note

The property will be saved into the set property, but if you would like to save them to GXSD.DAT as well, do not forget to press Save Properties button in the Edit -> Demo Preferences menu

 Note

If you would like to delete the set ROI/ROU in ADI Demo you can do that by pressing ALT + "I" for ROI or ALT + "U" for ROU.

8. MAGNIFIER

If the characters are very small on the visible processed image in ADI Demo, but there is a result for that, for example:

The screenshot shows the ADI Demo interface with a license plate recognition result for '66385M'. The interface includes a main image of three cars, a detailed view of the license plate, and a table of results.

#	Filename	Plate Text	Plate Category	Country / State	ANPR Time (ms)
1	EUR_01.jpg	AA164YN		ALB	102
2	EUR_02.jpg	3483FPK	MIXED	ESP	98
3	ADAB_01.jpg	66385M	COMMON	OMN	186
4	CAM_02.jpg	P726GKJ		GTM	179
5	EUR_03.jpg	ZN1163B		SJM	228
6	SAF_01.jpg	No result			368

But are not able to decide if it is correct or not, you can hover your mouse over the license plate which you would like to check and hold down the **right button** and a magnifier will highlight the surrounded area.

The screenshot shows the ADI Demo interface with a magnified view of the license plate '66385M'. The magnified view shows the license plate text and the surrounding area of the car. The table of results is also visible, with the third row highlighted.

#	Filename	Plate Text	Plate Category	Country / State	ANPR Time (ms)
2	EUR_02.jpg	3483FPK	MIXED	ESP	98
3	ADAB_01.jpg	66385M	COMMON	OMN	186
4	CAM_02.jpg	P726GKJ		GTM	179
5	EUR_03.jpg	ZN1163B		SJM	228
6	SAF_01.jpg	No result			368
7	C&E_01.jpg	No result			368

9. ANPR RESULT TABS

The ANPR Result Tabs is located to the right of the input image display area. It is divided into three main sections.

9.1. LICENSE PLATE RESULT:

Provides a detailed summary of the recognized image:

- **Plate Text:** license plate text
- **Plate Category:** category of the license plate
- **Country/State:** Country full name / state full name
- **Confidence:** overall confidence level (%) of the plate.
- **Character Size:** height of the characters, measured in pixels.
- **License Plate Frame:** pixel coordinates of the plate corners.
- **TypeID:** code containing country/state ID of the plate.
- **Background Color:** background color of license plate text.
- **Text Color:** color of the dedicated area of the license plate.
- **Dedicated Area Color:** color of dedicated area in RGB format.
- **ANPR Time (ms):** time to detect and evaluate the license plate.

ANPR Result	MMR Result	Additional Information
Plate Text		3483FPK
Plate Category		MIXED
Country / State		Spain
Confidence		97%
Character Size		18
License Plate Frame		[304; 351] [426; 355] [303; 374] [425; 378]
Type ID		128060
Background Color		RGB(255,255,255)
Text Color		RGB(0,0,0)
Dedicated Area Color		No color
ANPR Time (ms)		104

Only available if MMR is enabled!

- **Make&Model:** the recognized vehicle Make and Model data
- **Category:** the recognized vehicle category
- **Color:** the color of the recognized vehicle
- **Viewpoint:** the viewpoint of the recognized vehicle
- **Body Type:** Vehicle's structural category (e.g., sedan, van, truck).
- **Generation:** Identified model generation of the vehicle.
- **Variation:** Specific trim or version within the same model.
- **MMR Warning:** Alert indicating uncertain or low-confidence MMR result.
- **MMR Time (ms):** MMR processing time in milliseconds

ANPR Result	MMR Result	Additional Information
Make & Model		Audi A4 99%
Category		Car 99%
Color		BROWN 54%
Viewpoint		Rear-side 99%
Body Type		Combi/Wagon 99%
Generation		Not recognized -
Variation		Not recognized -
MMR Warning		-
MMR Time (ms)		156

3483FPK

Background Color: [Spanish Flag]

Text Color: [Black]

ESP

Additional Information

Plate Text	3483FPK
Plate Category	MIXED
Country / State	Spain
Confidence	97%
Character Size	18
License Plate Frame	[304; 351] [426; 355] [303; 374] [425; 378]
Type ID	128060
Background Color	RGB(255,255,255)
Text Color	RGB(0,0,0)
Dedicated Area Color	No color
ANPR Time (ms)	104

#	Filename	Plate Text	Plate Category	Country / State	ANPR Time (ms)	Make & Model	Category	Color	Viewpoint	MMR Warning	MMR Time (ms)				
1	0108_00.jpg	3483FPK	99%	MIXED	ESP	99%	BMW 3	99%	Car	99%	148				
2	CAM_03.jpg	M356139	98%	NIC	100%	Toyota Hilux	100%	Pick-up	100%	WHITE	99%	Front	99%	-	198

The detailed result view on the right-hand side of the image has been changed from the traditional expandable tree to a **tabbed and table-based layout**. Key results such as license plate number, recognized colors, country/state with flag, vehicle category, and make/model are now displayed visually above the tabbed results section.

The **main result table** at the bottom, displays core ANPR and, if enabled, MMR results. The default columns cover the most relevant data points. Additional result fields can be enabled by right-clicking the table header and selecting from the list.

For result elements where a confidence value is available, the application displays this value as well.

The default sorting is based on the row number (first column). It is possible to sort the results by other columns as well by clicking once on the desired column header. If the column contains confidence values, the results will be sorted by those values in ascending order, and clicking again will switch to descending order. If no confidence values are present, the sorting will follow alphabetical order. To sort alphabetically in a column that does contain confidence values, double-click the column header. The interface now includes **contextual tooltips**. When hovering the mouse over specific areas.

The screenshot shows a license plate 'M 356 139' on a white car. Below it is a table with columns: Country / State, ANPR Time (ms), Make & Model, Category, and Color. A tooltip is displayed over the 'Make & Model' header, providing instructions on how to interact with the table headers.

Country / State	ANPR Time (ms)	Make & Model	Category	Color
-	0	Daihats		47%
-	0			VN 55%
ALB 100%	58	Audi A4		44%
GBR / GBR 84%	91	Mazda		98%
DEU 59%	161	Toyota Hilux	Pick-up 100%	WHITE 99%
-	296	Jeep Compass	100% Car 100%	GRAY 44%

Contextual Tooltip:
 Press right mouse button to select columns.
 Sort by clicking on header of a column.
 One click: if there is a confidence value, then by confidence, otherwise by alphabetical order.
 Double click: in alphabetical order.

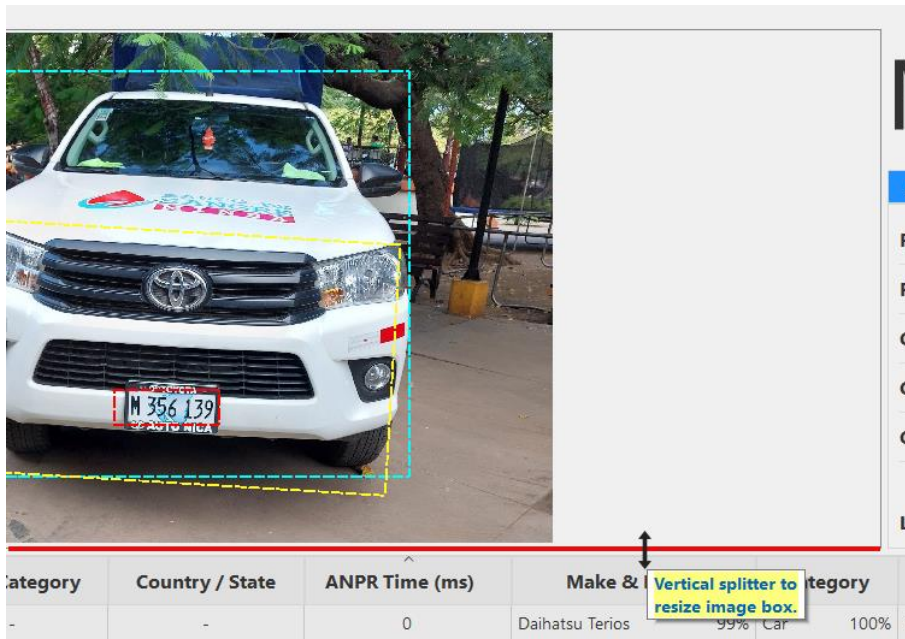
- Right-click + drag zoom is explained with tooltip: "Zoom mode: hold down the right mouse button and move the mouse."



- Horizontal splitter resizing is guided with: "Horizontal splitter to resize image box."



- Vertical splitter resizing is labeled: "Vertical splitter to resize image box."



10. ANPR DATA

Results of the ANPR process are displayed in the **log** section in the bottom part of the screen. The log contains the following data:

- **Filename:** name of the image file.
- **Plate Text:** alphanumeric license plate text.
- **Plate Category:** category of the license plate.
- **Country/State ISO 3166-1 alpha-3** country code / State. If you hover your mouse over the Country/State column in the Log area, it will pop up the full name of the country / state.
- **ANPR Time(ms):** ANPR processing time in milliseconds.

#	Filename	Plate Text	Plate Category	Country / State	ANPR Time (ms)
1	EUR_01.jpg	AA164YN 99%		ALB 100%	234
2	CAM_03.jpg	M356139 98%		NIC 100%	320
3	EUR_02.jpg	3483FPK 99%	MIXED	ESP 99%	206

Only in case if MMR reading is enabled in Demo Preferences and there is an ANPR result.

- **Make&Model:** the recognized vehicle Make and Model data
- **Category:** the recognized vehicle category
- **Color:** the color of the recognized vehicle
- **ViewPoint:** the viewpoint of the recognized vehicle
- **MMR Warning:** The MMR engine detected some uncertainty, conflict, or error.
- **MMR Time(ms):** MMR processing time in milliseconds

If you need more information about MMR please check this document: [MMR Brief Description](#).

#	Filename	Plate Text	Plate Category	Country / State	ANPR Time (ms)	Make & Model	Category	Color	Viewpoint	MMR Warning	MMR Time (ms)
1	CAM_03.jpg	M356139 98%		NIC 100%	317	Toyota Hilux 100%	Pick-up 100%	WHITE 99%	Front 99%	-	291
2	EUR_02.jpg	3483FPK 99%	MIXED	ESP 99%	185	BMW 3 99%	Car 99%	GRAY 40%	Front 99%	-	220

Note

The table above can be saved in a semi-colon delimited log file by clicking **File/Save Log**.

By clicking a data row, the corresponding image and its result tree will appear above the log.

11. DATA LOGGING

The application saves each ANPR process in a log file.

Naming format of the log file:

ADI_YYYY-mm-dd_hhmm_sss.log (ADI_Year-Month-Day_HourMinute_Second.log)

E.g.: ADI_2012-07-09_1712_001

If the date changes, the process of the automatic logging continues in a different log file.

When saving a log file, a separate *anpr* file with the same name will also be created in the same directory that contains the property settings of the currently used engine.

Format of the log file:

UTF16 (LE) encoding

Semicolon separated values

First Line: header with information in the following order:

"#;Filename;Plate Text;Plate Category;Country / State;Confidence;Character Size;LP Frame;Type ID;Background Color;Text Color;Dedicated Area Color;ANPR Time (ms);

Make & Model;Category;Color;Viewpoint;Body Type;Generation;Variation;MMR Warning;MMR Time (ms); Max. Belt Confidence; EADR type;EADR frame"

ADV DEMO

Note

From Carmen 7.3.1.28 this application is not part of the package any more.

1. INTRODUCTION

The ANPR Demo for Videos (ADV) is an application that was developed by Adaptive Recognition to serve as a simple, yet versatile tool for testing, evaluating, and familiarizing oneself with the core features the CARMEN® engine has to offer. This document will describe in detail the features and functionalities available within the demo software.

ADV can operate live (processing a live camera stream) or as a backend process (processing a recorded video stream). The demo supports the following video formats:

- .mjjpg
- .mjpeg
- .avi
- .mpg
- .mpeg
- .mp4
- .mkv



Not all images are adequate for ANPR processes, the input images (in this case the video frames) have to meet a specific set of criteria in order for the engine to be able to recognize the licenses. Please study the following document to learn more about these requirements: [Imaging for Carmen®](#).

This application can be found in the following folder:

- On Windows: "C:\Program Files\Adaptive Recognition\CARMEN softwares\Demos\ADV\"
- On Linux: "/opt/gx64/ADV_Demo/"

 Note

The latest available version from ANPR Demo for Videos: **7.4.0.22**

 Important!

Please note, that this application is not available for ARM package on Linux.

 Note

The application was tested only with MJPEG streams from ParkIT and FreewayCAM cameras. Using third-party cameras as a source may lead to performance issues.

2. MAIN SCREEN

After starting ADV, you will be presented with the main screen of the application. This window is split into four separate sections; the Active Stream, the Result Image, the Result Tabs and the Log sections. All four sections provide information in connection with the scanned image. The ADV menu bar consists of seven menu buttons, located in the upper left corner of the program window. The following menus are available:

- File
- Edit
- View
- Camera
- Video Player
- Process
- Help

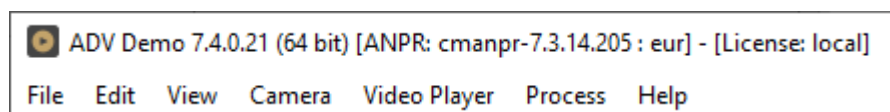
The screenshot displays the main interface of the ADV application. It is divided into four main sections:

- Active Stream:** Shows a live video feed of a street scene with a car and motorcycles.
- Result Image:** Shows the same scene with a white car highlighted by a red bounding box.
- Result Tree:** A hierarchical tree view showing detection results for the current frame, including confidence, type, color, and frame information.
- LOG:** A table listing all detected objects with their respective frame times, plate texts, country/state, confidence, and other attributes.

#	Frame Time	Plate Text	Country/State	Confidence	Frame	Character Size	Text Background	Text Color	Dedicated Color	ANPR Timeout
98.0.0	1979-01-01 01:03:37.200	821209F42	DN 1279020	37	05078011,048,8011,048,8206...	13	0.0,0.0	(251,251,251)	0.0,0.0	48
98.1.0	1979-01-01 01:03:37.220	821209F4	DN 1279020	37	05068710,048,8710,048,8206...	13	0.0,0.0	(251,251,251)	0.0,0.0	54
98.2.0	1979-01-01 01:03:37.260	821209F4	DN 1279020	38	05043210,048,3210,048,8206...	13	0.0,0.0	(251,251,251)	0.0,0.0	36
99.0.0	1979-01-01 01:03:40.000	8191444	DN 1279000	23	04088902,048,8902,048,8206...	13	0.0,0.0	(251,251,251)	0.0,0.0	48
99.1.0	1979-01-01 01:03:40.020	8191444#E	DN 1279000	21	04088710,048,8710,048,8206...	13	0.0,0.0	(251,251,251)	0.0,0.0	55
99.2.0	1979-01-01 01:03:40.040	8191444#E	DN 1279000	21	04073210,048,3210,048,8206...	13	0.0,0.0	(251,251,251)	0.0,0.0	48
100.0.0	1979-01-01 01:03:43.700	8188295#E	DN 1279020	38	04063010,048,3010,048,8206...	13	0.0,0.0	(251,251,251)	0.0,0.0	56
100.1.0	1979-01-01 01:03:43.800	8188295#E	DN 1279020	38	04053502,048,3502,048,8206...	14	0.0,0.0	(251,251,251)	0.0,0.0	67
100.2.0	1979-01-01 01:03:43.840	8188295#E	DN 1279020	39	04033010,048,3010,048,8206...	13	0.0,0.0	(251,251,251)	0.0,0.0	67
101.0.0	1979-01-01 01:03:48.300	8189783#E	DN 1279020	40	03063110,048,3110,048,8206...	14	0.0,0.0	(251,251,251)	0.0,0.0	71
101.1.0	1979-01-01 01:03:48.400	8189783#E	DN 1279020	40	03063002,048,3002,048,8206...	14	0.0,0.0	(251,251,251)	0.0,0.0	54
101.2.0	1979-01-01 01:03:48.440	8189783#E	DN 1279020	40	03053002,048,3002,048,8206...	14	0.0,0.0	(251,251,251)	0.0,0.0	48
102.0.0	1979-01-01 01:03:54.400	8191500#E	DN 1279020	41	03101110,048,1110,048,8206...	14	0.0,0.0	(251,251,251)	0.0,0.0	43
102.1.0	1979-01-01 01:03:54.520	8191500#E	DN 1279020	41	03081002,048,1002,048,8206...	13	0.0,0.0	(251,251,251)	0.0,0.0	43
102.2.0	1979-01-01 01:03:54.560	8191500#E	DN 1279020	41	03061002,048,1002,048,8206...	14	0.0,0.0	(251,251,251)	0.0,0.0	43

Quick overview of the main screen

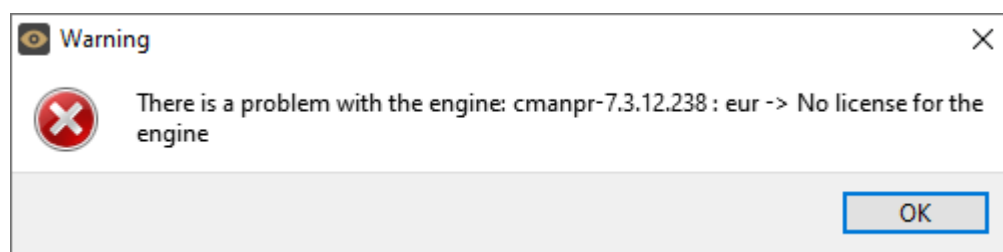
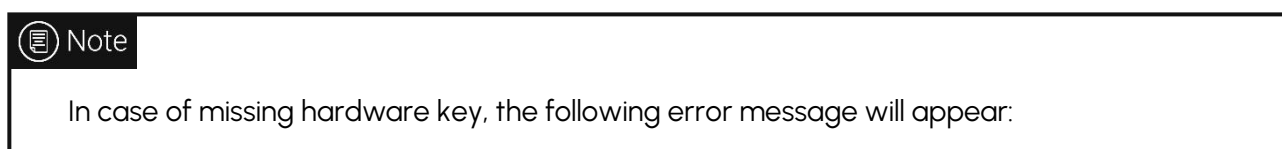
In the title you can see the following information:



ANPR: the currently used ANPR engine region and version

MMR: the currently used MMR engine region and version

License: showing where CARMEN® is looking for the licenses (local/network)



The upper part of the screen is divided into three sections:

- **Active Stream:** Section on the left that displays the active video stream.
- **Result Image:** Section in the centre that displays the image corresponding to the row selected in the log.
- **Result Tabs:** Section on the right that lists the details for each frame processed. ADV saves all engine parameters for each frame processed. In addition, it also displays an ANPR result tabs where it recognized a license plate.

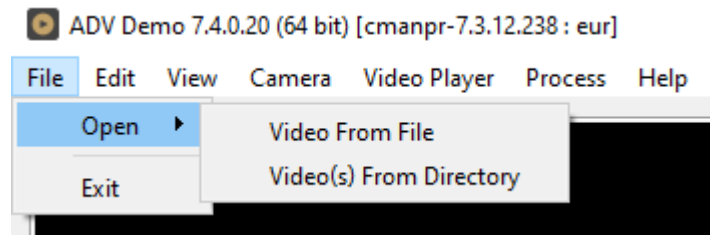
The lower part of the screen contains only one section.

- **Log:** Shows the license plate data for each frame. See a detailed description in the [ANPR Data](#) chapter.

3. FILE MENU

Open: displays two options:

- **Video from file:** Select this option to add a single video file (see supported formats above) for ANPR processing.
- **Video(s) From Directory:** Select to add a directory with multiple video files (see supported formats above) for ANPR processing.



Exit: Closes the application.

4. EDIT MENU

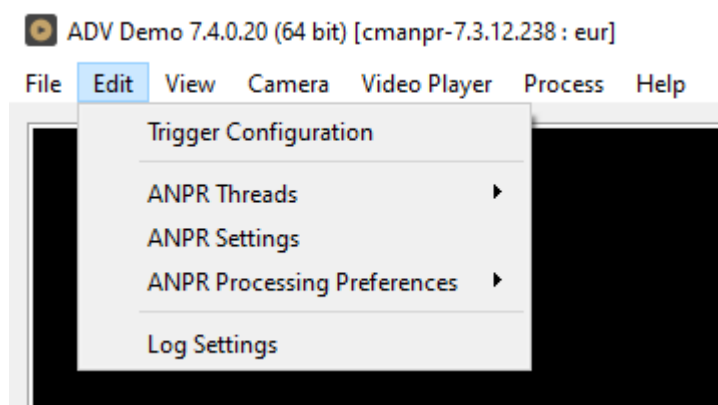
4.1. TRIGGER CONFIGURATION

! Important!

Trigger configuration largely depends on camera positioning, whether the camera deployed monitors front or rear license plates and many other factors. As a result, correct trigger configuration will always be custom for every installation.

The trigger signal can automatically identify an observed event (the time interval during which the system encounters a vehicle for ANPR processing). Typically, one event represents a sequence of frames/images of only one vehicle at a time.

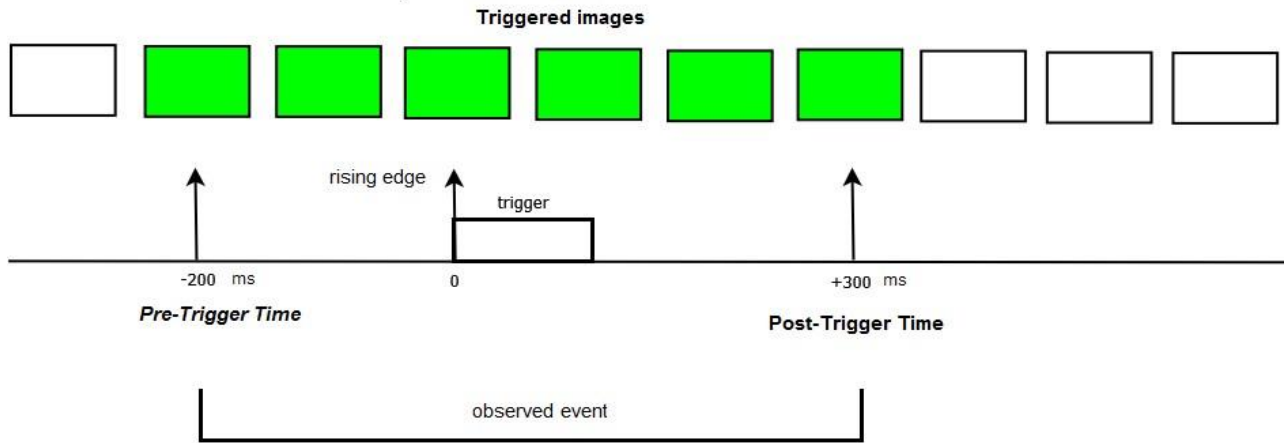
Trigger configuration allows the user to determine a time interval when the system anticipates only those frames from the continuous video frame sequence that are relevant for ANPR. From these relevant images, the engineer must decide what to select as triggered images. Triggered images should be those that were captured at an optimal time (when all of the optical characteristics of the target license plate are in their ideal ranges).



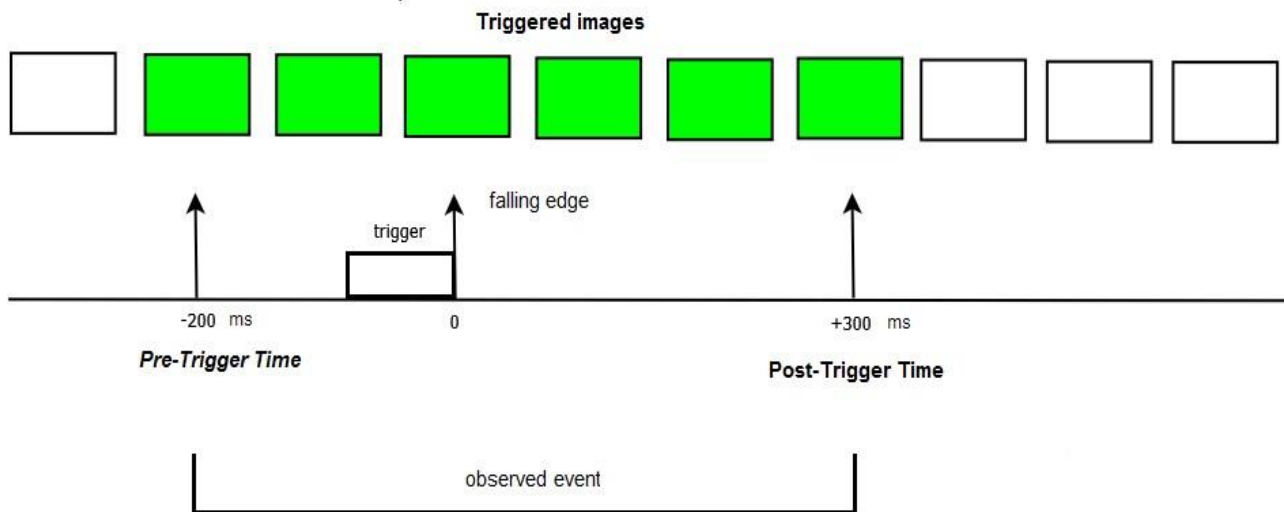
The diagrams below demonstrate how Pre-Trigger Time and Post-Trigger Time are used to define what the triggered images are within an observed event. The application will request the triggered images from the camera for ANPR processing. Since the camera buffers the images internally, even negative offset values of a few seconds (depending on the frame rate) can be set.

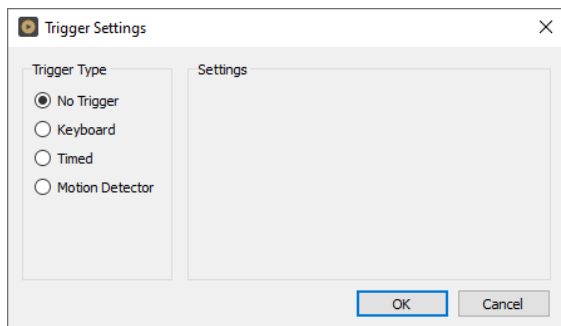
The following examples demonstrate some possible trigger scenarios:

1. Trigger on the rising edge

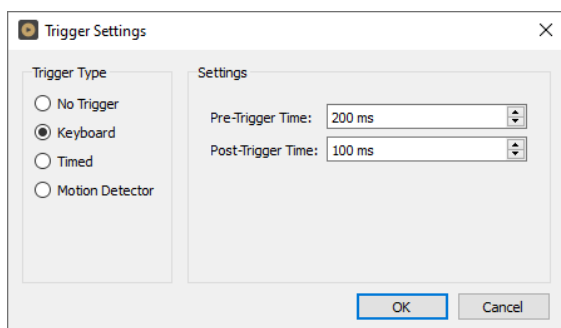


2. Trigger on the falling edge

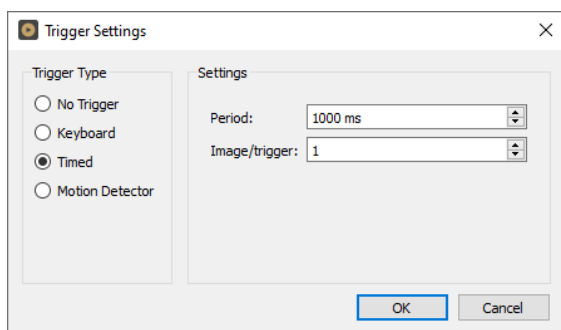




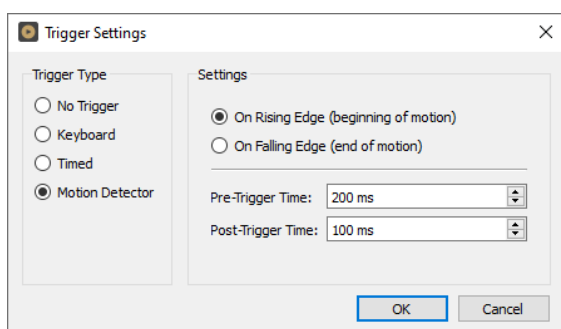
No Trigger: The engine processes all incoming frames that are not dropped from the buffer.



Keyboard: Generates a manual trigger event when the [F10] button is pressed. The engine processes all images within a specified time interval of the manual trigger (defined in milliseconds by the **Pre-Trigger Time** and **PostTrigger Time** values).



Timed: The engine processes images according to a preset frequency (defined in milliseconds, up to 60,000ms).



Motion Detector (Only applicable to Adaptive Recognition cameras): The application uses the built-in motion detection feature of the camera to create a trigger event. Select either the **On Rising Edge (beginning of motion)** or the **On Falling Edge (end of motion)** option. The first one will capture the images at the beginning of the motion, the second option at the end of motion. **Pre-Trigger Time** and **Post-Trigger Time** allows you to set a time interval before and after the trigger signal. The engine processes all images captured within this specified interval.

the beginning of the motion, the second option at the end of motion. **Pre-Trigger Time** and **Post-Trigger Time** allows you to set a time interval before and after the trigger signal. The engine processes all images captured within this specified interval.

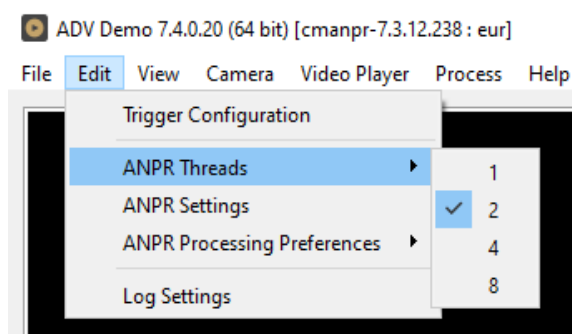
In **No Trigger** and **Timed** modes, the system sends individual images for processing, while in **Keyboard** and **Motion Detector** modes, it sends image packages based on the pre-set time intervals.

Note

When developing a final application with an efficient system architecture in mind, it is highly recommended to use a hardware trigger (e.g., inductive loop, infrared gate) in order to identify events. This way the events will only contain relevant frames with useful LPR information.

4.2. ANPR THREADS

Select the number of simultaneous ANPR processing threads. In order to run multiple processing threads parallel to each other, each thread must have a dedicated CPU core and a CARMEN® software license (NNC hardware key). Multi-core software licenses for parallel processing, are available in dual and quad versions.



Possible values: 1, 2, 4, 8

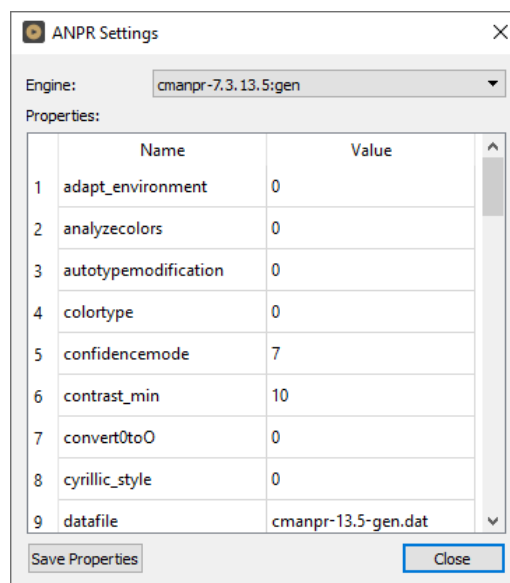
4.3. ANPR SETTINGS

Contains various configuration options related to the ANPR process. The engine used for processing is also selected here.

By clicking this menu item, the following pop-up window will appear:

Engine: Click the drop-down menu to select an engine from the list of installed engines.

Properties: Lists the property values for the selected engine. To adjust the pre-set values, click on one of the value fields and then input a desired value. A full description of the engine properties can be found in the [CARMEN® ANPR Reference Manual](#).



Click on the **[Close]** button to apply your changes. These settings will remain active until the application is closed or until the engine in use is replaced with another one from the drop-down list. When you exit the application, the changes will be lost and the values will revert to the ones saved in the gxsd.dat configuration file.

Clicking on the **[Save Properties]** button will write the current values of each property into the gxsd.dat configuration file. This feature requires write permissions to the gxsd.dat file. The default values of an engine can be reset by reinstalling (uninstalling and installing) it, check it [here](#) how to do that.

4.4. ANPR PROCESSING PREFERENCES

The **ANPR Processing Preferences** menu item contains three sub-menu items.

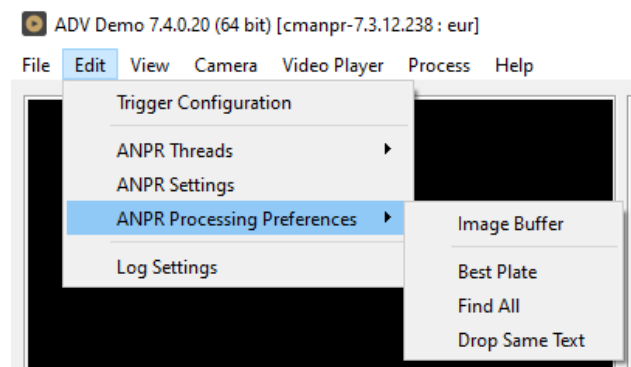
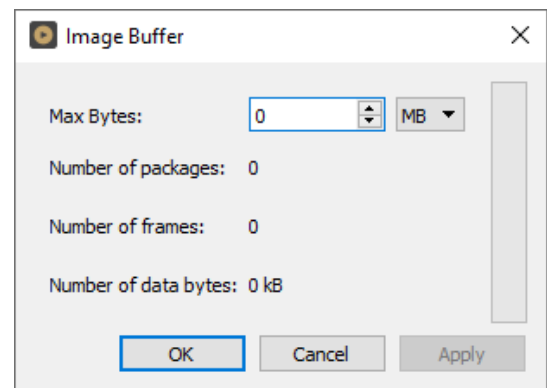


Image Buffer:

Max Bytes: Maximum size of the image buffer, specified in kilobytes or megabytes. The size of the image buffer is equal to the sum of all the image sizes in the buffer.

When the image buffer reaches the pre-set value, ADV will start to overwrite the oldest images in the buffer.

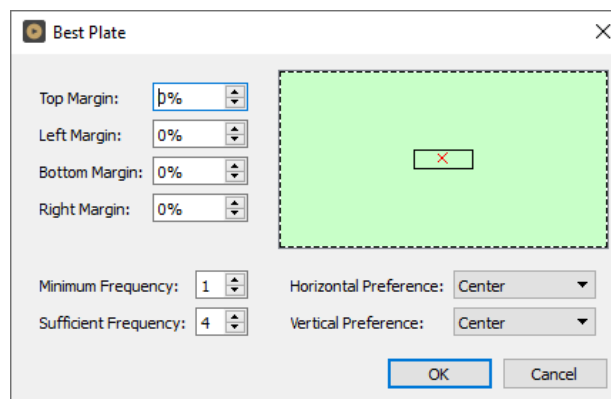


Best Plate:

This is an algorithm that aids the selection of the best possible results.

ADV anticipates plates in the Plate Area marked with green on the sample frame. You can adjust the Plate Area by configuring the **Top Margin, Left Margin, Bottom Margin, and Right Margin** fields.

The Target Area (rectangle marked with an X) is used to identify the part of the frame that is most likely to contain an optimal license plate image.



Minimum Frequency: The minimum number of identical license plate recognitions within the image package of a single event.

Sufficient Frequency: The number of identical license plate recognitions required to stop additional image processing within the image package of a single event.

Horizontal Preference: Moves the target area horizontally.

Vertical Preference: Moves the Target Area vertically.

ADV will not consider a result for final selection if any of the following applies:

4. The plate is not within the margins
5. Minimum Frequency criteria is not fulfilled

From the remaining results, the application will select a final result by finding the frame that is closest to the Target Area (frame centre point closest to "X").

Find All:

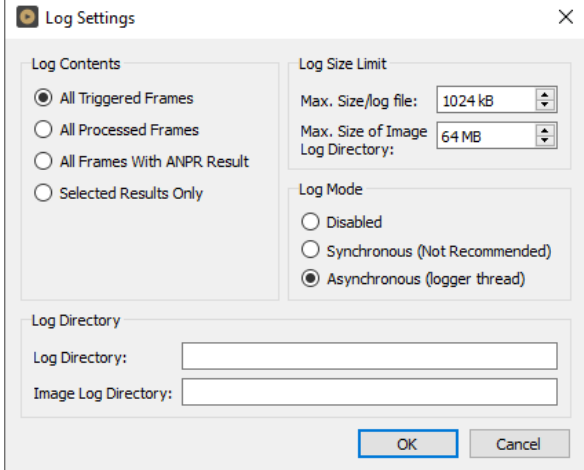
If checked, the system searches for all license plates on the image.

Drop Same Text:

If checked, the system will not provide a result when there was an identical license plate text in the previous 20 seconds.

4.5. LOG SETTINGS

Allows you to specify the contents, size, and path of the log as well as the corresponding images.



The screenshot shows a dialog box titled "Log Settings" with a close button (X) in the top right corner. The dialog is divided into several sections:

- Log Contents:** A group box containing four radio button options:
 - All Triggered Frames
 - All Processed Frames
 - All Frames With ANPR Result
 - Selected Results Only
- Log Size Limit:** A group box containing three spinners:
 - Max. Size/log file: 1024 kB
 - Max. Size of Image: 64 MB
 - Log Directory: (empty)
- Log Mode:** A group box containing three radio button options:
 - Disabled
 - Synchronous (Not Recommended)
 - Asynchronous (logger thread)
- Log Directory:** A group box containing two text input fields:
 - Log Directory: (empty)
 - Image Log Directory: (empty)

At the bottom right of the dialog, there are two buttons: "OK" and "Cancel".

5. VIEW

5.1. FONTS

Set Tree Font...

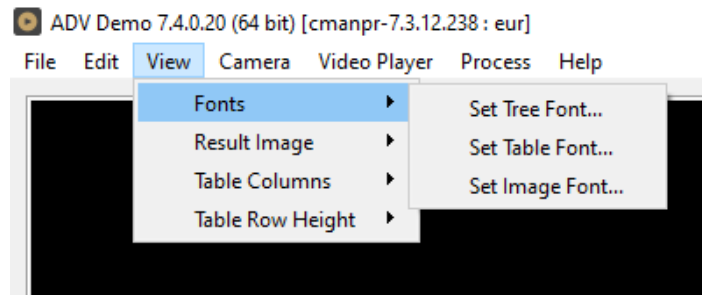
Change the font used to display the Result Tree (the detailed ANPR data found in the upper right corner of the main screen).

Set Table Fonts...

Change the font used to display the Log (the table of results in the lower part of the ADV screen).

Set Image Font...

Select the font used to display the license plate text displayed in the image.



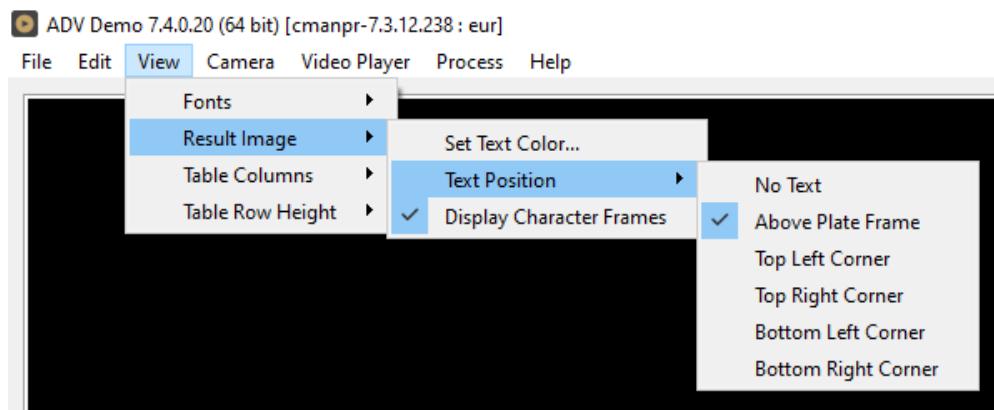
5.2. RESULT IMAGE

Set Text Color:

Select the color of text displayed in the image.

Text position:

Adjust the position of the text displayed in the image.



Possible values:

Result text will not be displayed in the image: No text

Result text will be displayed as indicated by sub-menu item:

- Above Plate Frame
- Top Left Corner
- Top Right Corner
- Bottom Left Corner
- Bottom Right Corner

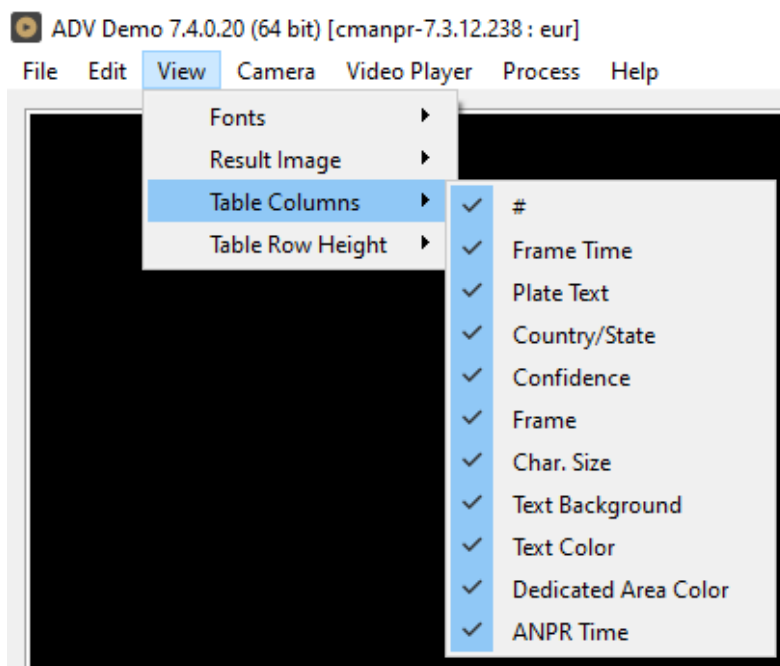
Display Character Frames:

Display a frame around each license plate character found in the image. Possible values: on, off

5.3. TABLE COLUMNS:

Select the columns that should be displayed in the header of the Log.

- # (sequence number of frame)
- Frame Time
- Plate Text
- Country/State
- Confidence
- Frame
- Char.Size
- Text Background
- Text Color
- Dedicated Area Color
- ANPR Time

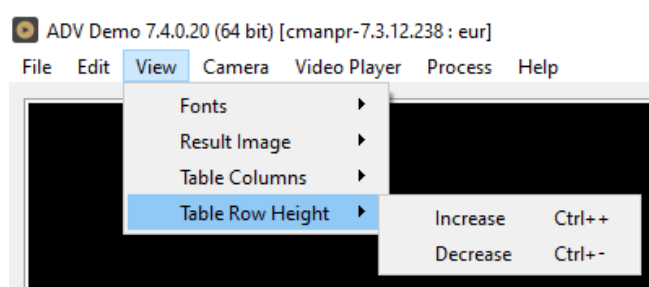


5.4. TABLE ROW HEIGHT

Adjusts the row height of the Log by clicking on **Increase** or **Decrease**.

Hotkeys for Increase: **Ctrl + +**

Hotkeys for Decrease: **Ctrl + -**

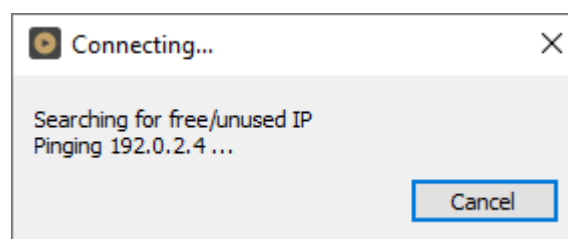
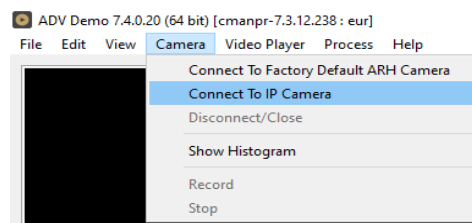


6. CAMERA MENU

By clicking this menu item, the following pop-up window will appear:

Connect To New Adaptive Recognition Camera:

The default factory IP address of new Adaptive Recognition cameras is set to 192.0.2.3 with the 255.255.255.0 netmask. Clicking on this menu item will start a scanning process that will look for this type of IP cameras in the default subnet.



Important!

You must set up the network adapter with the same subnet range as the factory default IP subnet of the ARH cameras. Please avoid IP address conflicts by selecting a number different from the camera's default "3" in the last section of the IP address.

Connect To IP Camera:

Allows user to connect to a camera on the network.

Note

The stream URL of the cameras are different by manufacturers! Please visit your camera's manual to get more information about its stream link (only "mjpeg" stream is allowed). Authentication on the cameras may also change this URL address.

Disconnect/Close:

Disconnects the camera and closes the session.

Show Histogram:

Displays the image histogram on the live view image of the camera (Feature only available with Adaptive Recognition cameras).

Record:

Allows user to save the MJPEG stream downloaded from the camera (or other URL).

Stop:

Stops the recording of the MJPEG stream.

7. VIDEO PLAYER MENU

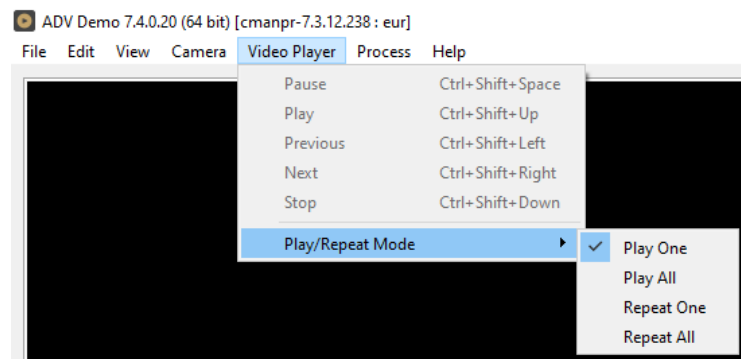
Allows the user to control playback and switch between multiple video streams (if more than one video stream has been loaded).

Play One: Playback will stop at the end of the video stream.

Play All: Playback continues with the next video stream until the end of the last one.

Repeat One: Plays the selected video in an infinite loop.

Repeat All: Plays all selected videos in an infinite loop.



8. PROCESS MENU

Start:

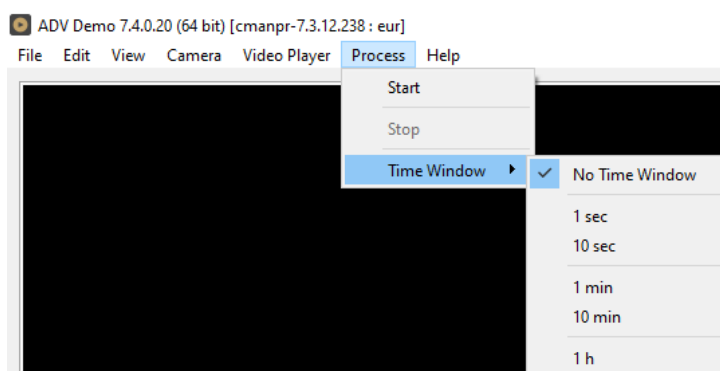
Allows the user to initiate the ANPR process.

Stop:

Allows the user to manually stop the ANPR process.

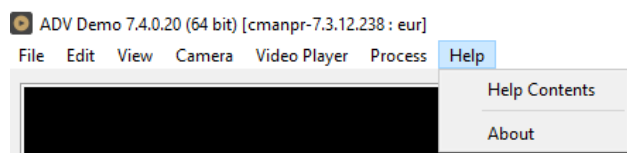
Time Window:

The ANPR process can also be stopped by setting a timer in the Time Window sub-menu prior to starting the ANPR process.



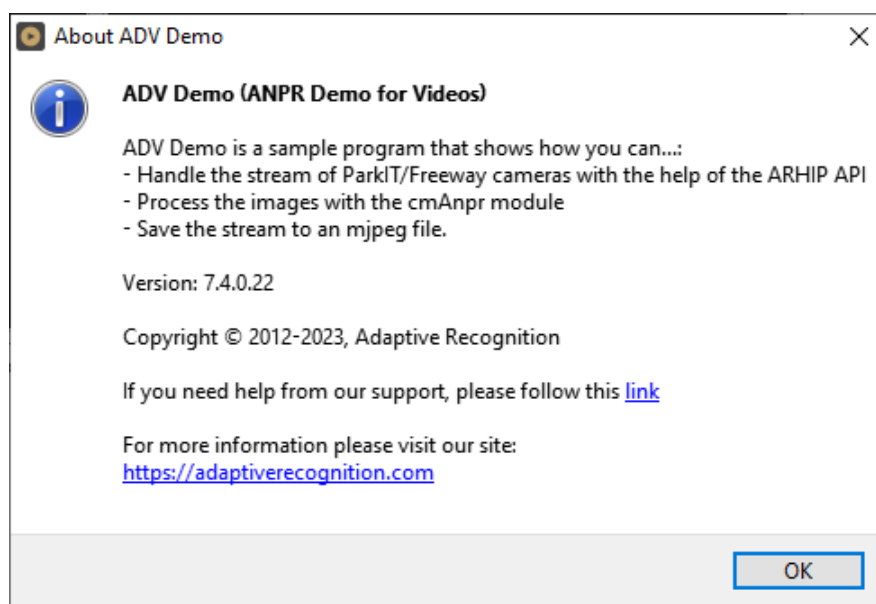
9. HELP MENU

By clicking this menu item, the following pop-up window will appear:



Help Contents: Opens this User's Manual

About: Provides the license, version, and copyright information for the installed application.



10. ANPR RESULT TREE

For each selected frame in the **log** section, the **Result Tree** displays all of the corresponding details available.

FRAME:

Shows date- and timestamp of the processed frame, and contains subheadings called **Params** and **ANPR Results**.

I. PARAMS:

Parameters contains a list of all the engine property settings.

II. ANPR RESULT(S):

1. LICENSE PLATE RESULT:

- **Confidence:** Overall confidence level (%) of the plate.
- **Type:** Code containing country/state ID of the plate.
- **Color:** Color of the dedicated area on the license plate.
- **BkColor:** Background color of license plate text.
- **Frame:** Pixel coordinates of the plate corners.

2. CHARACTERS:

Individual details of each character of the final result. Ordered from left to right.

- **Code:** Unicode character ID.
- **Confidence:** Confidence level (%) of the overall plate.
- **Color:** Color of the character.
- **BkColor:** Background color of character.
- **Frame:** Pixel coordinates of the character corners.

3. TIPS:

List of preliminary results from which the engine assembled the final result.

- **Code:** Unicode character ID.
- **Confidence:** Confidence level (%) of the character tip.
- **Color:** Color of the character.
- **BkColor:** Background color of character.
- **Frame:** Pixel coordinates of the character corners

Results	
▲ Frame	2012-11-27 14:26:44.535
▷ Params	
▲ ANPR Result(s)	
▲ ARH001	
Confidence:	89
Type:	0
Color(RGB):	(0,0,0)
BkColor(RGB):	(255,255,255)
Frame:	((322,330),(439,329),(439,358),(322,358))
▲ Characters (ARH001)	
▲ A	
Code:	0x0041
Confidence:	99
Color(RGB):	(0,0,0)
BkColor(RGB):	(255,255,255)
Frame:	((324,331),(338,331),(338,357),(324,358))
▷ R	
▷ H	
▷ 0	
▷ 0	
▷ 1	
▲ Tips	
▲ A	
Code:	0x0041
Confidence:	100
Color(RGB):	(0,0,0)
BkColor(RGB):	(255,255,255)
Frame:	((326,361),(340,361),(340,387),(326,387))
▲ R	
Code:	0x0052
Confidence:	99
Color(RGB):	(0,0,0)
BkColor(RGB):	(255,255,255)
Frame:	((344,361),(358,361),(358,387),(344,387))
▲ H	
Code:	0x0048
Confidence:	99
Color(RGB):	(0,0,0)
BkColor(RGB):	(255,255,255)
Frame:	((362,361),(377,361),(377,387),(362,387))
▷ 0	
▷ 0	
▷ 1	

11. ANPR DATA

Results of the ANPR process are displayed in a log found at the bottom part of the screen. The log contains the following data:

- **Frame Time:** Time and date when the image was capture
- **Plate Text:** Alphanumeric license plate text.
- **Country/State** ISO 3166-1 alpha-3 country code.
- **Confidence:** Overall confidence level of the plate.
- **Frame:** Pixel coordinates of the corners of the license plate.
- **Character Size:** Average height of the characters in pixels.
- **Text Background:** Background color of text in RGB t.
- **Text Color :** Character color in RGB.
- **Dedicated Color:** Color of dedicated area in RGB.
- **ANPR Time(ms):** ANPR processing time in milliseconds.

#	Frame Time	Plate Text	Country/State	Confidence	Frame	Character Size	Text Background	Text Color	ANPR Time(ms)
24.0.0	2012-12-04 16:26:23.691	[ARH001]		92	((306,332),(424,329),(425,360),(306,362))	26	(255,255,255)	(0,0,0)	21
23.0.0	2012-12-04 16:26:23.642	[ARH001]		91	((307,332),(422,330),(423,359),(307,361))	25	(255,255,255)	(0,0,0)	18
22.0.0	2012-12-04 16:26:23.592	[ARH001]		92	((307,333),(424,331),(424,360),(308,361))	26	(255,255,255)	(0,0,0)	21
21.0.0	2012-12-04 16:26:23.541	[ARH001]		91	((307,332),(424,331),(424,359),(308,360))	25	(255,255,255)	(0,0,0)	22
29.0.0	2012-12-04 16:26:23.941	[ARH001]		91	((307,332),(424,329),(425,358),(308,361))	26	(255,255,255)	(0,0,0)	20
20.0.0	2012-12-04 16:26:23.491	[ARH001]		92	((307,332),(424,329),(425,360),(308,362))	26	(255,255,255)	(0,0,0)	22
1.0.0	2012-12-04 16:26:22.541	[ARH001]		91	((306,333),(424,329),(424,359),(306,362))	26	(255,255,255)	(0,0,0)	22
18.0.0	2012-12-04 16:26:23.391	[ARH001]		91	((306,333),(424,332),(424,359),(306,361))	26	(255,255,255)	(0,0,0)	17
17.0.0	2012-12-04 16:26:23.342	[ARH001]		93	((306,332),(424,331),(424,360),(306,360))	27	(255,255,255)	(0,0,0)	18
16.0.0	2012-12-04 16:26:23.292	[ARH001]		91	((306,331),(423,331),(423,360),(306,360))	25	(255,255,255)	(0,0,0)	19
15.0.0	2012-12-04 16:26:23.243	[ARH001]		90	((306,331),(423,331),(423,360),(306,360))	26	(255,255,255)	(0,0,0)	19

Note

The data present in the table above is included in the log file.

12. DATA LOGGING

Log entries of the ANPR processes are logged by the application in the CARMEN® installation folder.

Naming format of the log file:

ADV_YYYY-mm-dd_hhmm_sss.log ADV_Year-Month-Day_HourMinute_counter E.g.: ADV_2021-02-08_1726_000.log

If the date changes, then automatic logging continues in a different log file

When saving a log file, a separate anpr file with the same name is also created in the same directory that contains the property settings of the engine being used.

FILE FORMAT:

UTF16 (LE) encoding

Semicolon separated values

First Line: header as follows (see detailed descriptions under [ANPR Data](#))

#;Path; Plate Text;Country/State;Confidence;Frame;Character Size;Text Background;Text Color;

Dedicated Color;ANPR Time(ms);

Note

In case of missing hardware key, the application indicates it in the status bar at the bottom of the screen.

ODI DEMO

1. INTRODUCTION

The OCR Demo for Images (ODI) is a program that was developed by Adaptive Recognition to serve as a simple and versatile tool for evaluating our core OCR technology before having to first develop an application.

The goal of the SDK is to allow you to develop a similar or even a more complicated application based on the Carmen® API in order to meet the exact requirements of the particular project where it will be deployed.

The demo can handle both a single image as well as an image directory as an input source.

Supported image formats: **.jpg, .jpeg, .png, .bmp, .jp2**

This application can be found in the following folder:

- On Windows: "C:\Program Files\Adaptive Recognition\CARMEN softwares\Demos\ODI\"
- On Linux: "/opt/gx64/ODI_Demo/"

Note

The latest available version from OCR Demo for Images: **7.3.2.1**

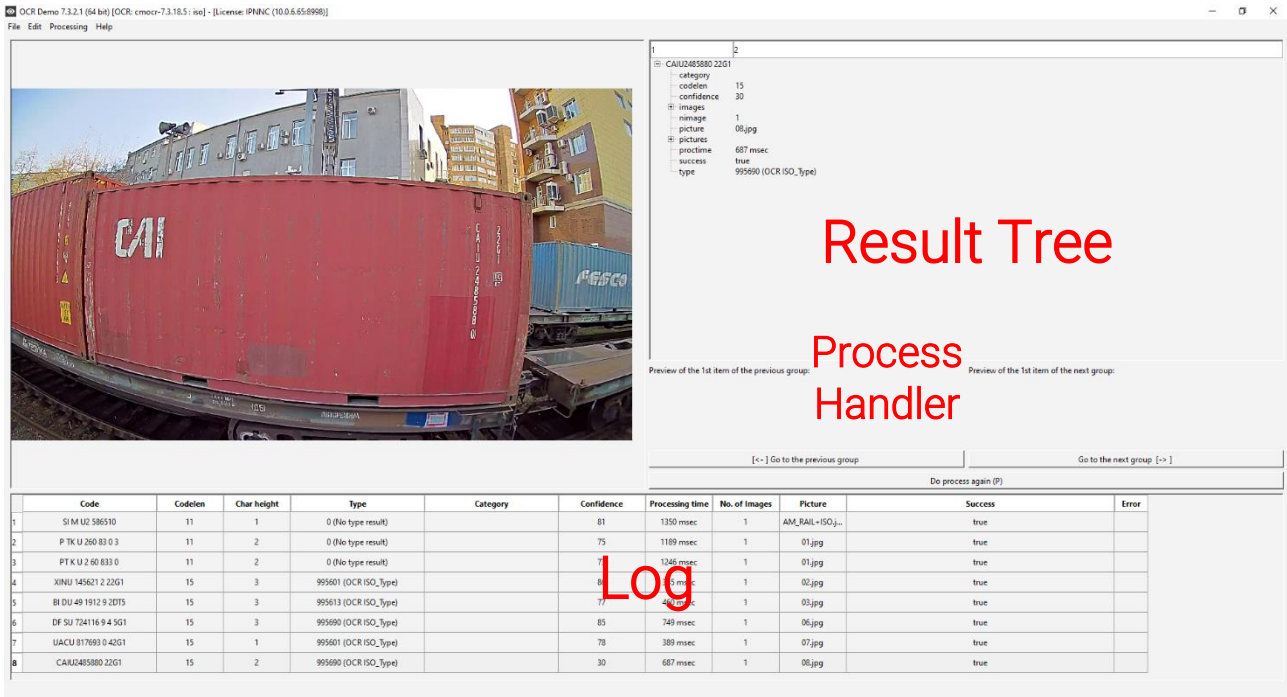
Not all images are adequate for OCR, the input images have to meet a specific set of criteria in order for the engine to be able to recognize them. Please study the following document to learn more about these requirements: [Imaging for Carmen®](#).

Important!

Please note, that this application is not available for ARM package on Linux.

2. MAIN SCREEN

After starting the demo, you will be presented with the main screen of the application. This window is split into four separate sections; the **Result Image**, the **Result Tree**, the **Process Handler** and the **Log** sections. All four sections provide information in connection with the scanned image.



Quick overview of the main screen

In the title you can see the following information:

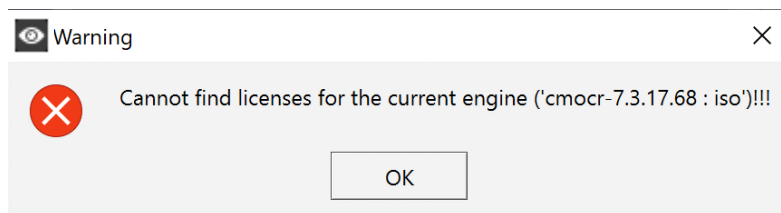
OCR Demo 7.3.2.1 (64 bit) [OCR: cmocr-7.3.19.41 : iso] - [License: IPNNC (10.0.6.65:8998)]

OCR: the currently used OCR engine type and version

License: showing where CARMEN® is looking for the licenses (local/network)

Note

In case of missing hardware key, the following error message will appear:



3. FILE MENU

The ODI menu bar consists of four menu buttons, located in the upper left corner of the program window as follows:

File, Edit, Processing, and Help.



The **File** menu contains the following menu items:

3.1. OPEN DIRECTORY

Click to specify a directory containing images for OCR processing.

3.2. OPEN IMAGES

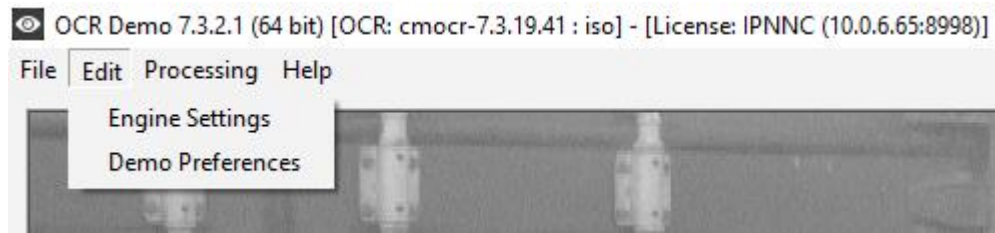
Click to select specific images for OCR processing.

3.3. CLOSE

Close the ODI Demo application.

4. EDIT MENU

The **Edit** menu contains the following menu items:



4.1. ENGINE SETTINGS

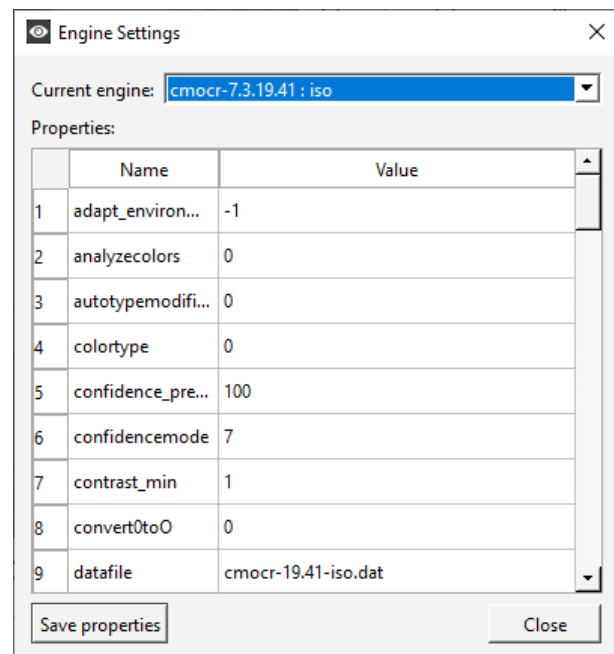
Contains various configuration options related to the OCR process. The engine used for processing is also selected here.

By clicking this menu item, the following pop-up window will appear:

Current engine: Click the drop-down menu to select an engine from the list of installed engines.

Properties: Lists the property values for the selected engine. To adjust the pre-set values, click on one of the value fields and then input a desired value. A full description of the engine properties can be found in the [CARMEN® OCR Reference Manual](#).

Click on the **[Close]** button to apply your changes. These settings will remain active until the application is closed or until the engine in use is replaced with another one from the drop-down list. When you exit the application, the changes will be lost and the values will revert to the ones saved in the gxsd.dat configuration file.



Engine Settings

Clicking on the **[Save Properties]** button will write the current values of each property into the gxsd.dat configuration file. This feature requires write permissions to the gxsd.dat file. The default values of an engine can be reset by reinstalling (uninstalling and installing) it, check it [here](#) how to do that. This function works only with ANPR properties as the MMR properties are read only.

4.2. DEMO PREFERENCES

Contains various configuration options related to the demo software.

By clicking this menu item, the following pop-up window will appear:

Logging section:

Time Format:

Allow the user to customize the time format.

Log File:

Browse where you would like to save the log files.

Enable Logging:

If you check this box, ODI will save the log files as you set above.

Processing section:

Max. no. of container codes in a picture:

Set maximum how many codes are allowed to return from 1 image.

Max. no. of cached images:

Set maximum how many images can be stored in the queue.

Max. no. of images in group:

Set maximum how many images are allowed in one group of images (Group means that engine will return only 1 time for the same code from the number of images in the group).

Auto Process Delay:

Set how many milliseconds should wait between 2 OCR processes.

Rotate Image for Display:

Set if you would like to rotate the images if the codes are turned on the image.

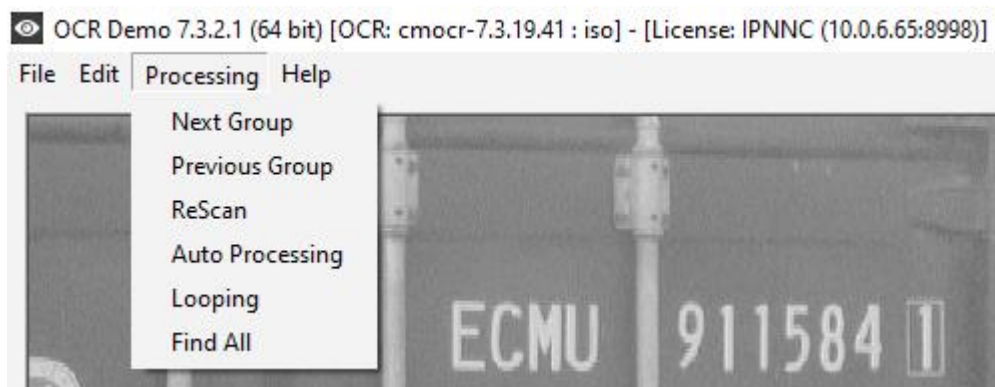
Initial Directory:

Browse from which directory ODI should search for an image for processing on start-up.

5. PROCESSING MENU

The **Processing** menu contains the following menu items:

Click **Next Group** (Right Arrow) or **Previous Group** (Left Arrow) to navigate between the images/groups. Alternatively, you may also use the left/right arrow keys on the keyboard.



ReScan:

If you click on this item, ODI will do the OCR process again on the same image/group.

Auto-Process:

If checked, the application plays the images automatically as a slideshow. (You can find related settings in the Demo Preferences menu)

Looping:

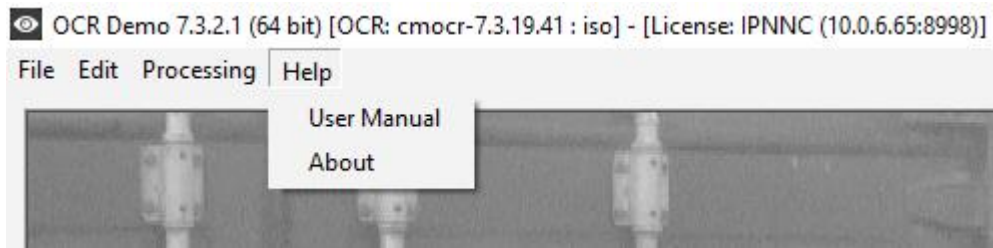
If checked, the program continuously repeats processing of the specified image sequence/folder.

Find All:

If checked, the engine will search for all plates in the image that conform to the set parameters.

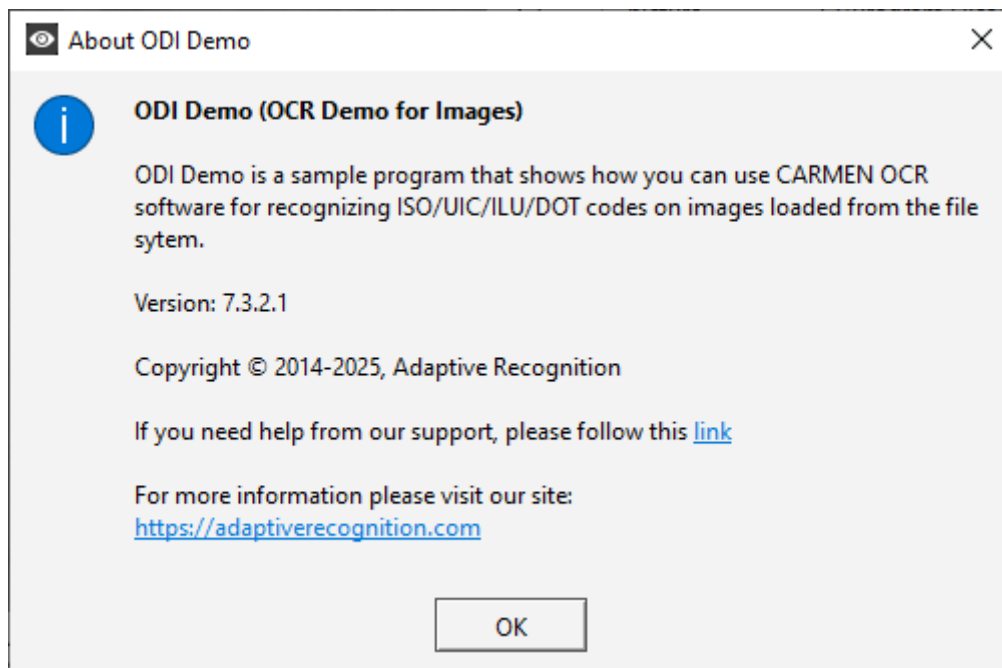
6. HELP MENU

The **Help** menu contains the following menu items:



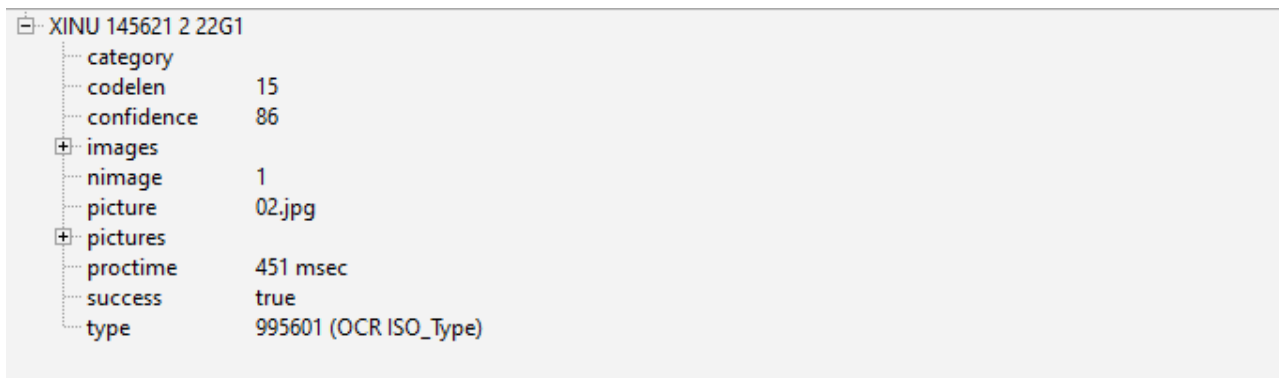
User Manual: Opens this User Manual.

About: Provides the license, version, and copyright information for the installed application.



7. OCR RESULT TREE

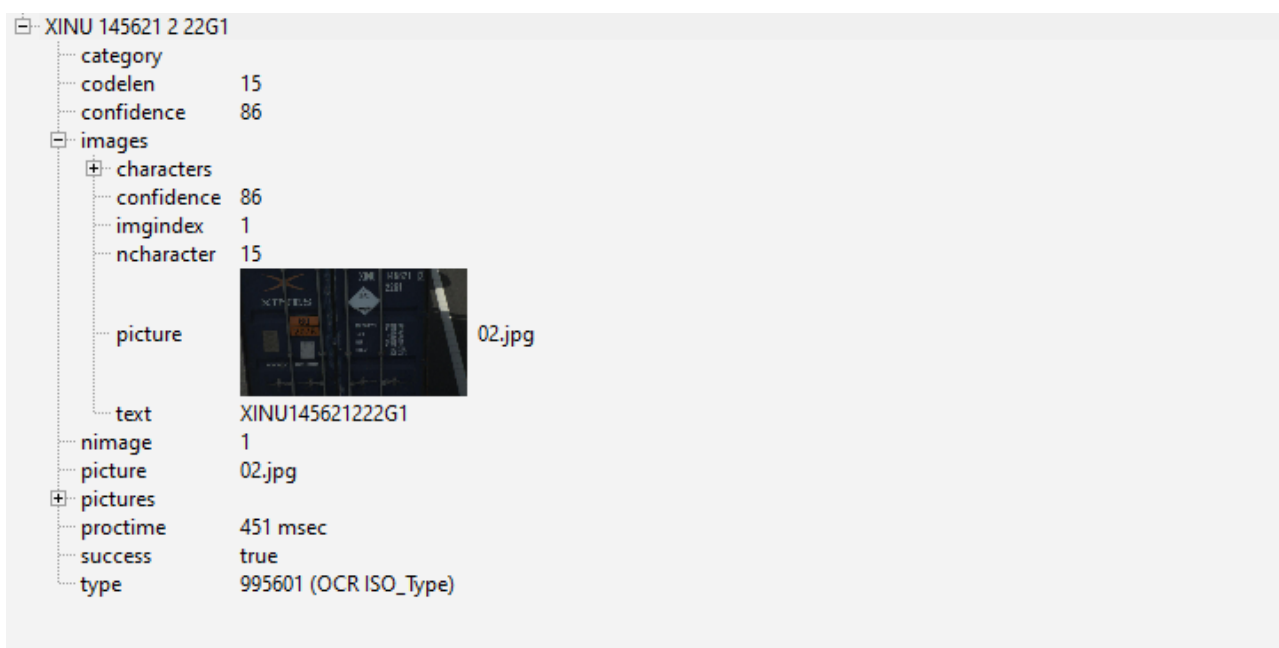
The OCR Result Tree is located to the right of the input image display area. It is divided into three main sections.



1. Code result:

Provides a detailed summary of the recognized image:

- **codelen:** length of the code
- **confidence:** the overall confidence of the result
- **nimage:** number of images in the group
- **picture:** path of the first image in the group
- **proctime:** the processing time in milliseconds
- **success:** true or false based on the results
- **type:** our internal ID of the currently recognized code



2. Results on the images:

Individual details for each recognized character on every image in the group. Ordered from left to right than up to down.

• **characters:**

- code: ASCII code
- code_char: ACSII character
- confidence: confidence of this character
- frame: pixel coordinates of the character corners
- **confidence:** confidence level (%) of the overall plate
- **imgindex:** index of the image in the group
- **ncharacter:** number of characters on this image
- **picture:** resized image, with its path
- **text:** recognized text from the image
- **type:** our internal ID of the currently recognized code

```

images
├── characters
│   ├── code          69
│   ├── code_char     E
│   ├── confidence    100
│   ├── frame         (15,7)-(16,7)-(16,9)-(15,9)
│   ├── code          67
│   ├── code_char     C
│   ├── confidence    100
│   ├── frame         (17,7)-(18,7)-(18,9)-(17,9)
│   ├── code          77
│   ├── code_char     M
│   ├── confidence    98
│   ├── frame         (18,7)-(19,7)-(19,9)-(18,9)
│   ├── code          85
│   ├── code_char     U
│   ├── confidence    100
│   ├── frame         (19,7)-(20,7)-(20,9)-(19,9)
│   └── code          57
    
```

3. Pictures:

List of resized images in the group with their path.



8. OCR DATA

Results of the OCR process are displayed in the **log** section in the bottom part of the screen. The log contains the following data:

- **Code:** the read code from the image/group.
- **Codelen:** length of the code.
- **Char height:** average height of the characters
- **Type:** our internal ID of the currently recognized code
- **Confidence:** overall confidence level of the plate.
- **Processing time:** OCR processing time in milliseconds
- **No. of Images:** number of images in the group.
- **Picture:** path of the first image in the group.
- **Success:** true or false based on the result.

	Code	Codelen	Char height	Type	Category	Confidence	Processing time	No. of Images	Picture	Success	Error
1	TCVU 252032 9	11	29	0 (No type result)		83	938 msec	1	C:/Program File...	true	
2	BIDU49 1912 9 2DT5	15	46	995613 (OCR ISO_Type)		76	966 msec	1	C:/Program File...	true	
3	DFSU 724116 9 45G1	15	47	995690 (OCR ISO_Type)		76	2363 msec	1	C:/Program File...	true	
4	UACU 817693 0 42G1	15	28	995601 (OCR ISO_Type)		91	4264 msec	1	C:/Program File...	true	

9. DATA LOGGING

The application saves each OCR process in a log file if you allow it in the Demo Preferences menu.
Naming format of the log file: it is set in Demo Preferences menu

Format of the log file:

UTF16 (LE) encoding

Semicolon separated values

First Line: header with information in the following order:

time;code;codelen;type;category;confidence;proctime;nimage;picture;success;error

SDK DESCRIPTION

Example codes to show how to use CARMEN® for license plate and industrial code recognition.

1. ANPR SAMPLE CODES

1.1. CMANPR01_LIST_ENGINES

List engines and licenses. Shows how to list the current and installed engines, along with the licenses they can use.

1.2. CMANPR02_PLATE

Detect one number plate on one image. At the beginning, the application checks the gxsd.dat XML syntax and stops execution if errors are found. Demonstrates performing ANPR on an image and printing the result.

1.3. CMANPR03_PLATES

Detect all number plates on an image. Demonstrates performing ANPR on an image and printing all results.

1.4. CMANPR04_IMGLIST

Run ANPR on multiple images and detect all number plates. Shows how to do an ANPR on more images and print out all the result(s) in text.

1.5. CMANPR05_THREADS

Show the advantage of multiple Carmen ANPR licenses. Demonstrates parallel ANPR processing using multiple CPU cores and licenses. Requires at least two CPU cores and two core licenses.

1.6. CMANPR06_EMPTY_ADR

Empty ADR reading example. Demonstrates reading empty ADR plates with ADR-capable engines.

1.7. CMANPR07_K_LICENSE

Show an ANPR process using "K" licenses. Demonstrates use of "K" licenses, showing credit info after each ANPR from a folder of images. Uses `cmAnpr::GetCreditInfo`.

1.8. CMANPR08_MULTI_STAGE

Show multi-step engine usage. Demonstrates fallback logic: if the first engine fails, another is used for the same image.

1.9. CMANPR09_MMR_AUTO

CMANPR sample program using MMR. Shows how to use the automatic MMR feature of `cmAnpr` module. Uses automatic MMR after successful ANPR.

1.10. CMANPR10_MMR_MANUAL

CMANPR sample program using MMR. Shows how to use the manual MMR feature of `cmAnpr` module. Demonstrates manual MMR usage.

1.11. CMANPR11_MMR_MANUAL_WO_PLATE

CMANPR sample program using MMR. Shows how to use the manual MMR feature without detecting number plates. Demonstrates MMR vehicle frame detection without ANPR.

1.12. CMANPR12_PAXBELT

CMANPR sample program using `OdPaxBelt`. Shows how to use 'odpaxbelt' engines. Demonstrates detection of passengers and analysis of their seatbelt status.

2. OCR SAMPLE CODES

2.1. CMOCR01_IMGSEQ

Read container code from an image sequence input. Analyzes multiple images of the same container and outputs combined and per-image results.

2.2. CMOCR02_MULTI_CODES

Find multiple codes on one image. Uses `ReadPublicCode` and `NextPublicCode` to extract multiple public codes from a single image.

2.3. CMOCR03_ANPR_ADR_OCR

Combined License Plate, ADR, and OCR reading from one image. Requires multiple engines (regional ANPR, ADR, and OCR) to extract all data from a single image.

2.4. CMOCR04_THREADS

Multithreaded OCR processing. Demonstrates how to perform concurrent OCR recognition on multiple images using threading to improve throughput and performance.

2.5. CMOCR05_MULTI_ENGINE

OCR using multiple engine types. Demonstrates how to switch or combine multiple OCR engines to improve recognition accuracy based on code type or region.

1. GX SAMPLE CODES

1.1. GXDEVICES01

List devices example. Shows how to list GX system devices.

1.2. GXDEVICES02

List the number of the installed licenses in the GX system. Demonstrates retrieving installed license count.

1.3. GXLICENSES01

List licenses example. Demonstrates listing the available licenses in the GX system.



2. EXAMPLE CODES AVAILABILITY

2.1. ON WINDOWS

	C	C++	C#	Java
CMANPR01_LIST_ENGINES	+	+	+	+
CMANPR02_PLATE	+	+	+	+
CMANPR03_PLATES	+	+	+	+
CMANPR04_IMGLIST	+	+	+	+
CMANPR05_THREADS	NA	+	+	+
CMANPR06EMPTY_ADR	+	+	+	+
CMANPR07_K_LICENSE	+	+	+	+
CMANPR08_MULTI_STAGE	+	+	+	+
CMANPR09_MMR_AUTO	+	+	+	+
CMANPR10_MMR_MANUAL	+	+	+	+
CMANPR11_MMR_MANUAL_WO_PLATE	+	+	+	+
CMANPR12_PAXBELT	+	+	+	+
CMOCR01_IMGSEQ	+	+	+	+
CMOCR02_MULTIPLE_CODES	+	+	+	+
CMOCR03_ANPR_ADR_OCR	NA	+	+	+
CMOCR04_THREADS	NA	+	+	+
CMOCR05_MULTI_ENGINE	NA	+	+	+
GXDEVICES01	+	+	+	NA
GXDEVICES02	+	+	+	+
GXLICENSES01	+	+	+	+

+	Available
NA	Not Available

2.2. ON LINUX

	C	C++	Java
CMANPR EXAMPLES			
CMANPR01_LIST_ENGINES	+	+	+
CMANPR02_PLATE	+	+	+
CMANPR03_PLATES	+	+	+
CMANPR04_IMGLIST	+	+	+
CMANPR05_THREADS	NA	+	+
CMANPR06EMPTY_ADR	+	+	+
CMANPR07_K_LICENSE	+	+	+
CMANPR08_MULTI_STAGE	+	+	+
CMANPR09_MMR_AUTO	+	+	+
CMANPR10_MMR_MANUAL	+	+	+
CMANPR11_MMR_MANUAL_WO_PLATE	+	+	+
CMANPR12_PAXBELT	+	+	+
CMOCR EXAMPLES			
CMOCR01_IMGSEQ	+	+	+
CMOCR02_MULTIPLE_CODES	+	+	+
CMOCR03_ANPR_ADR_OCR	NA	+	+
CMOCR04_THREADS	NA	+	+
CMOCR05_MULTI_ENGINE	NA	+	+
GX EXAMPLES			
GXDEVICES01	+	+	NA
GXDEVICES02	+	+	+
GXLICENSES01	+	+	+

+	Available
NA	Not Available

CONTACT INFORMATION

Headquarters:

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

Service Address:

Adaptive Recognition, Hungary Inc.
Ipari Park HRSZ1113/1 HU
2074 Perbál Hungary
Web: adaptiverecognition.com/support/

Adaptive Recognition Hungary Technical Support System (ATSS) is designed to provide you the fastest and most proficient assistance, so you can quickly get back to business.

Information regarding your hardware, latest software updates and manuals are easily accessible for customers via our [Documents Site \(www.adaptiverecognition.com/doc\)](http://www.adaptiverecognition.com/doc) after a quick registration.

New User

If this is your first online support request, please contact your sales representative to register you in our Support System. More help [here \(www.adaptiverecognition.com/support/\)](http://www.adaptiverecognition.com/support/)!

Returning User

All registered ATSS customers receive a personal access link via e-mail. If you previously received a confirmation message from ATSS, it contains the embedded link that allows you to securely enter the support site.

