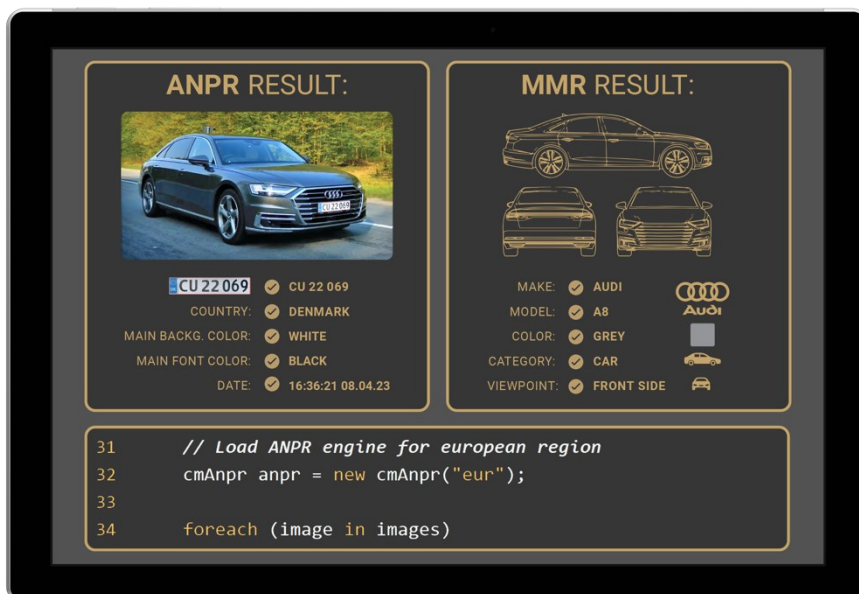


Carmen[®] User Manual



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

Carmen®

User Manual

Document version: 2023-10-02

Table of Contents

LICENSE MANAGER.....	6
1. INTRODUCTION.....	6
2. WHAT IS THE PURPOSE OF THE LICENSE MANAGER.....	6
3. INSTALLATION OF THE LICENSE MANAGER.....	7
4. OPENING THE LICENSE MANAGER.....	7
4.1. ON WINDOWS.....	7
4.2. ON LINUX.....	7
5. STRUCTURE OF THE LICENSE MANAGER.....	8
5.1. STATIC LICENSES.....	8
5.2. DYNAMICALLY UPLOADABLE LICENSES.....	9
5.3. UPLOAD LICENSES.....	10
6. LICENSE INSTALLATION.....	11
7. LICMAN.....	12
8. TROUBLESHOOTING.....	16
ENGINE MANAGER.....	17
1. INTRODUCTION.....	17
2. ENGINE INSTALLATION.....	18
2.1. INSTALLATION ON WINDOWS.....	19
2.2. INSTALLATION ON LINUX.....	20
3. ENGINE MANAGER PRO APPLICATION.....	21



4.	UNINSTALLING ENGINE(S)	23
4.1.	FROM WINDOWS	23
4.2.	FROM LINUX	23
5.	ENGMAN	24
	LICENSE SERVER	27
1.	INTRODUCTION	27
2.	SERVER APPLICATION	28
2.1.	SERVER APPLICATION AS CONSOLE APPLICATION	29
2.2.	SERVER APPLICATION AS A SERVICE	31
2.3.	SERVER APPLICATION FOR MORE THAN ONE CLIENT	32
3.	ConfigLSClient	33
	VIDEO_SDK	35
1.	INTRODUCTION	35
2.	ENGINE AND LICENSE REQUIREMENTS	35
3.	BUILD REQUIREMENTS	35
3.1.	LINUX	35
3.2.	WINDOWS	37
4.	VIDEO INPUT	39
4.1.	NETWORK STREAM	39
4.2.	VIDEO FILE	40
5.	A MINIMAL APPLICATION EXAMPLE	41
5.1.	CREATE ANPR	41
5.2.	ONEVENTCALLBACK	42
5.3.	CREATE STREAMPROCESSOR OBJECT	43
5.4.	START STREAMPROCESSING	44
6.	CONSTRUCTING COMPONENTS	45
6.1.	ANPR BUILDER	45
6.2.	MMR BUILDER	46
6.3.	STREAMPROCESSOR BUILDER	46
6.4.	LOGGER	55
7.	RESULT CLASSES	57
7.1.	IMAGE CLASSES	57
7.2.	ANPR RESULT CLASSES	62
7.3.	MMR RESULT CLASSES	65
7.4.	EVENT CLASSES	68
8.	REGION LIST	74
9.	KNOWN ISSUES	75
	ADI DEMO	76
1.	INTRODUCTION	76



- 2. MAIN SCREEN.....77
- 3. FILE MENU.....78
 - 3.1. OPEN IMAGE(S).....78
 - 3.2. OPEN DIRECTORY78
 - 3.3. SAVE LOG78
- 4. EDIT MENU79
 - 4.1. ANPR / MMR SETTINGS.....79
 - 4.2. DEMO PREFERENCES.....80
- 5. PROCESS MENU.....81
- 6. HELP MENU.....82
- 7. SET ROI/ROU ON THE IMAGE.....83
- 8. MAGNIFIER.....86
- 9. ANPR RESULT TREE.....87
- 10. ANPR DATA.....89
- 11. DATA LOGGING90
- ADV DEMO.....91
 - 1. INTRODUCTION91
 - 2. MAIN SCREEN.....92
 - 3. FILE MENU.....93
 - 4. EDIT MENU94
 - 4.1. TRIGGER CONFIGURATION94
 - 4.2. ANPR THREADS97
 - 4.3. ANPR SETTINGS98
 - 4.4. ANPR PROCESSING PREFERENCES.....99
 - 4.5. LOG SETTINGS101
 - 5. VIEW.....101
 - 5.1. FONTS101
 - 5.2. RESULT IMAGE.....102
 - 5.3. TABLE COLUMNS:.....103
 - 5.4. TABLE ROW HEIGHT103
 - 6. CAMERA MENU104
 - 7. VIDEO PLAYER MENU105
 - 8. PROCESS MENU.....106
 - 9. HELP MENU.....106
 - 10. ANPR RESULT TREE.....107
 - 11. ANPR DATA.....108
 - 12. DATA LOGGING109
- ODI DEMO.....110
 - 1. INTRODUCTION110



2.	MAIN SCREEN.....	111
3.	FILE MENU.....	112
3.1.	OPEN DIRECTORY.....	112
3.2.	OPEN IMAGES.....	112
3.3.	CLOSE.....	112
4.	EDIT MENU.....	113
4.1.	ENGINE SETTINGS.....	113
4.2.	DEMO PREFERENCES.....	114
5.	PROCESSING MENU.....	115
6.	HELP MENU.....	116
7.	OCR RESULT TREE.....	117
8.	OCR DATA.....	119
9.	DATA LOGGING.....	120
	CONTACT INFORMATION.....	121



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: **7.3.1.20**

! Important

Please note, that this application is not available for ARM and CenOS6 packages 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:\Program Files\Adaptive Recognition\Common Utils\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:\Program Files (x86)\ARH\Common Utils\LicenseManager"*
- in case of 64bit OS: *"C:\Program Files\ARH\Common Utils\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 7.3.1.20 (64 bit) interface. It is divided into three main sections, each highlighted with a numbered callout:

- 1. Static licenses:** This section contains two tables. The first table, 'Old license storage devices', has columns for 'Serial num.' and 'Device type'. The second table, 'Licenses', has columns for 'Lic. No.', 'Lic. type', and 'Description'.
- 2. Dynamically uploadable licenses:** This section contains two tables. The first table, 'New license storage devices', has columns for 'Serial num.', 'Device type', and 'HW group'. The second table, 'Licenses', has columns for 'Lic. ID', 'Lic. date', 'HWID/HWGRP', 'Lic. type', 'Expiry date', and 'Description'. Below these tables are several checkboxes and buttons:
 - Auto refresh devices and licenses
 - Auto upload licenses for new devices
 - Upload licenses
 - Last licenses
 - Best licenses
 - Buttons: Refresh devices and licenses, Online manual, Auto select and upload licenses.
- 3. Upload licenses:** This section contains a 'License directory' field with a 'Browse' button. Below it is a 'Saved user licenses' table with columns for 'Lic. ID', 'Lic. date', 'HWID/HWGRP', 'Lic. type', 'Expiry date', and 'Description'. To the right of the table are several buttons and checkboxes:
 - Buttons: Upload licenses, Clear licenses, Save changes, Summary.
 - Checkboxes: Auto save after upload, Auto summary after upload, Create log file.
 - Field: Log directory: with a 'Browse' button.

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 'Expire 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 where 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 is not part of the installed package, please turn to [AR support](#) if you need it.

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: [1] Device type: USB key, Serial: 11111111 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:

[1] Device type: USB key, Serial: 11111111:

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!	Select the directory which contains the license and not the license file itself. When the folder is selected the Choose button will be active!
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 presence 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 than 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?	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!



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:\Program Files\Adaptive Recognition\CARMEN softwares\EngineManagerPro\"
- On Linux: "\opt\gx64\EngineManagerPro\"

Note

The latest available version from Engine Manager Pro: **7.3.1.10**

! Important

Please note, that this application is not available for ARM and CenOS6 packages 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 to be downloaded from ATSS and arrive in a zip file. For example, the 2020 Q3 general engine: *cmanpr-gen-7.3.12.169_20Q3.zip*. This zip file contains 2 different folders:

- linux
- windows

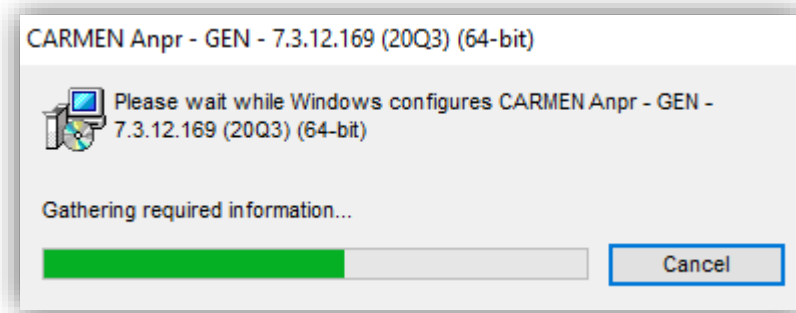
Note

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 2020 Q3 general engine: *cmanpr_gen_7.3.12.169_20Q3_x86.msi* or *cmanpr_gen_7.3.12.169_20Q3_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.



Click **yes** when asked for allowing app to make changes to your device.

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 (for Visual Studio 2015, 2017 and 2019) is a must 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.

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

3. ENGINE MANAGER PRO APPLICATION

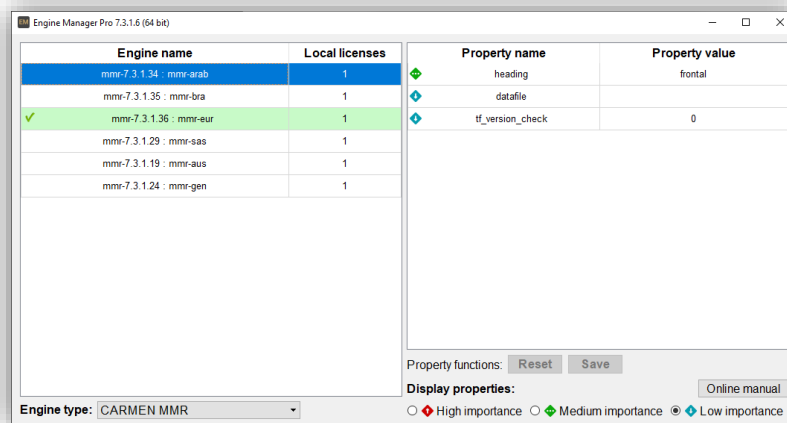
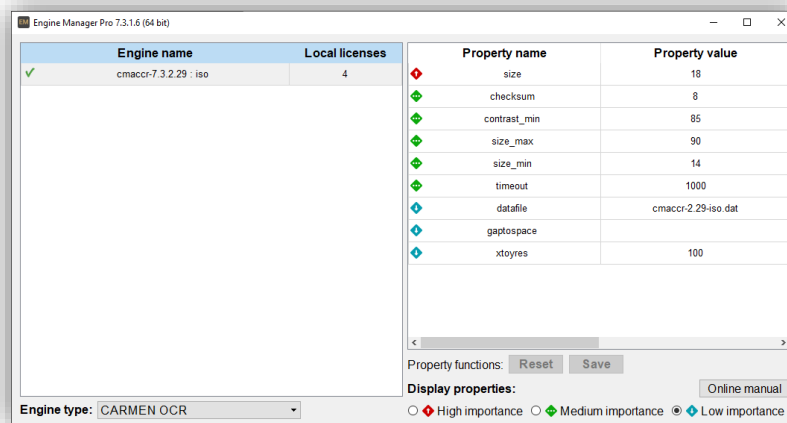
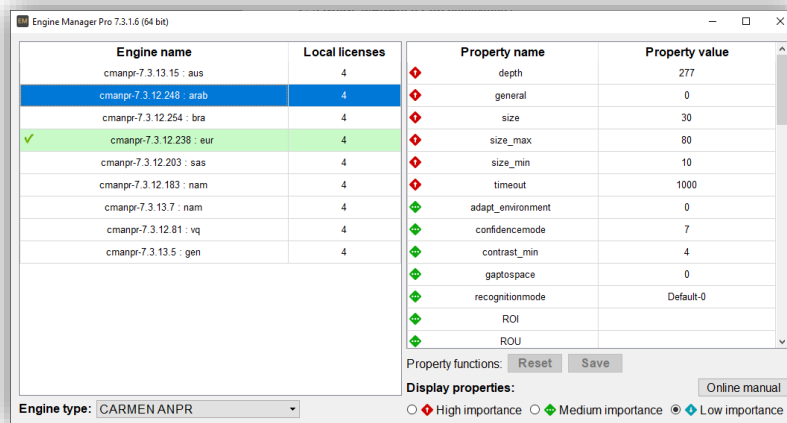
Once the CARMEN® ANPR / OCR Software and the recognition engine(s) have been installed open the Engine Manager Pro application.

On Windows: navigate to **Start / All Programs / ARH Inc. / Engine Manager** or type “Engine Manager” in the search box of your **Start Menu**.

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!



Engine name:

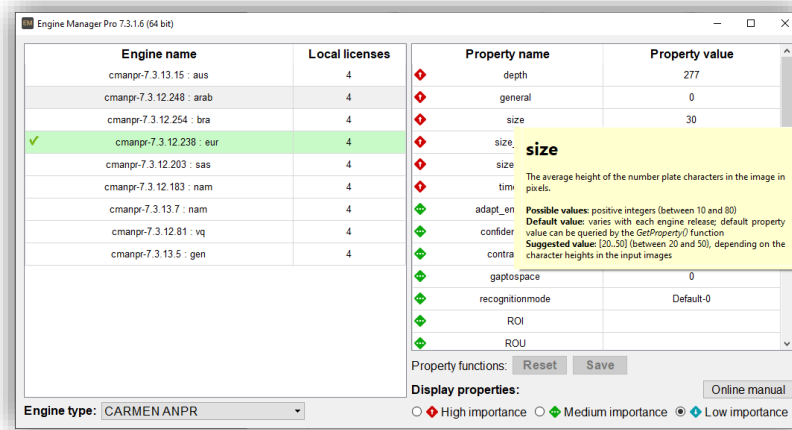
List of all successfully installed engines.

Local Licenses:

Available number of licenses for the given engine (Single: 1, Dual: 2, Quad: 4)

Property name / Property value:

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



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:

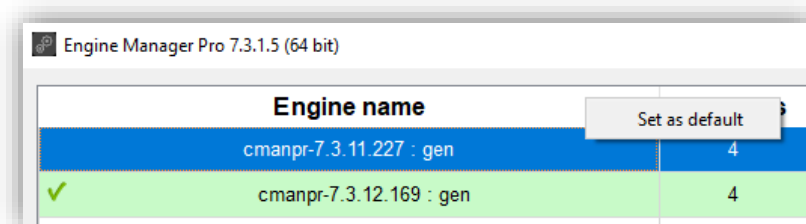
With this selector, you can filter the engine properties by their importance or their impact to the recognition results.

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”.



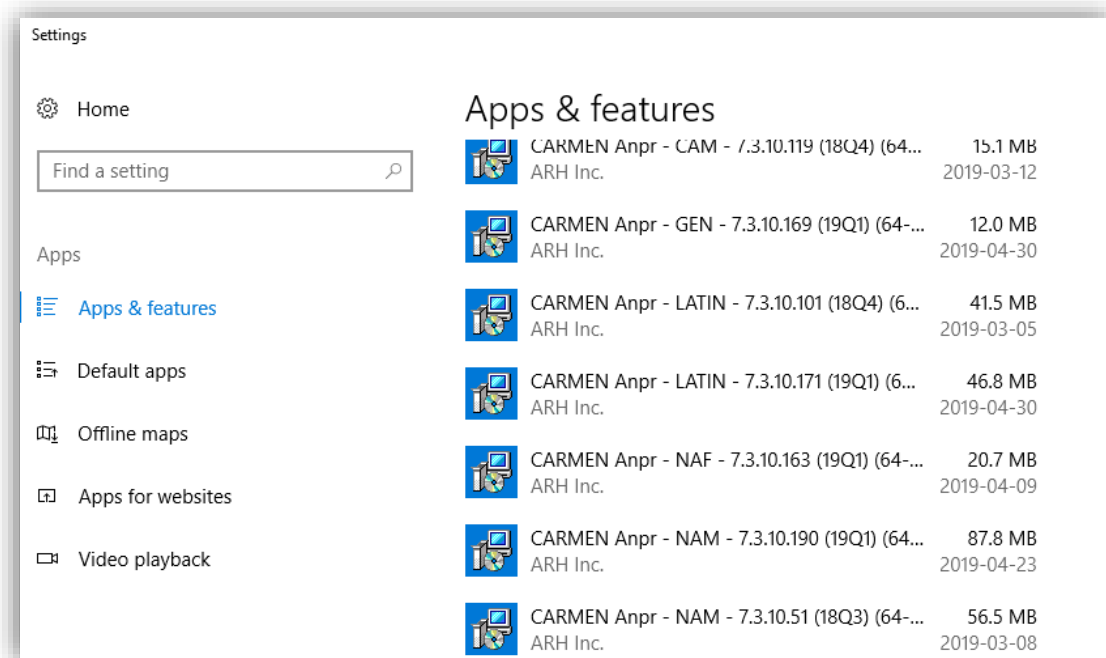
4. UNINSTALLING ENGINE(S)

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: **7.3.1.32**



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 not handling 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:\Program Files\Adaptive Recognition\Common

Utils\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:\Program Files\Adaptive Recognition\Common Utils\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:
 - o The server is running. More clients can connect.
 - o The server is running. NO more clients can connect!
 - o 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 and MMR 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: **1.1.0**

2. ENGINE AND LICENSE REQUIREMENTS

This SDK uses CARMEN® ANPR and MMR engines. Proper engines and licenses need to be installed in the running system. In stream building the program has to set the required region, which determine the used ANPR and MMR engine regions, and the library chooses the latest installed engines. Active regional engines need to be available, not being used by another application, for the corresponding license being used. The library also uses a Plate Finder engine for triggering. This is installed with the SDK library and needs a license to be available, not being used by another application.

3. BUILD REQUIREMENTS

CARMEN® Video SDK 1.1 version supports both Linux and Windows. Binary compatibility between different versions is not guaranteed.

3.1. LINUX

At the moment, we have released only one translation, in which we compile the core library with GCC 9.4.0 in Release for x86_64 ARM architecture with 64-bit version. We provide C and C++ interfaces for this (there is no C# interface in Linux at this time).

CARMEN® Video SDK requires a minimum of 7.3.1.26 CARMEN® ANPR package installed. This version of CARMEN® can be found in the package. The user must install this before installing the CARMEN® Video SDK, otherwise the installer will warn the user and exit with an error.

The installer puts headers in the “usr/include/carmen_video_sdk/c” and “usr/include/carmen_video_sdk/cpp” folders, and the shared object files in “/usr/lib/carmen_video_sdk”.

Besides that, it adds a file “/etc/ld.so.conf.d/carmen_video_sdk.conf” to link the libraries and updates to the ldconfig.

To run the samples, you need the CARMEN® ANPR engine and you may need the MMR engine.

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

3.1.1. 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. Import 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.

3.1.2. 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 things:

1. Import 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.

3.2. WINDOWS

The MSI contains the software components required for running, except for CARMEN® ANPR and MMR engines.

At the moment, we release only one translation, in which we compile the core library with VS 2019 in Release for x86_64 architecture with 64-bit version. We provide C, C++ and C# interfaces for this.

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

This 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.

3.2.1. 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. Import 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.

3.2.2. 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 things:

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

1. Import directory is: "\$ENV{CMV_INSTALL_DIR}\\sdk\C\\include
; \$ENV{CMV_INSTALL_DIR}\\sdk\C++\\include"
3. Link directory is: "\$ENV{CMV_INSTALL_DIR}\\sdk\C\\lib "
4. 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.

3.2.3. C#

Target architecture	Target platform	Target framework
x86_64	64 bit	.NETFramework v4.0

In order to use this library, you have to link the dynamic library from sdk/CSharp.

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

4. 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 is equipped with an advanced vehicle detection algorithm. This means, your camera does not need any hardware triggering for selecting the images from the stream to do number plate recognition. In turn, it needs a moving vehicle (car, bus, truck, etc..) on the stream to work.

The processing also works on recordings from fixed-installation cameras and moving vehicles. On fixed-installation, processing takes more power.

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.

4.1. 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 at a sufficient speed, the buffer will fill up (which with high RAM usage) which may result in some frames being 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.

It may happen that the network stream is interrupted, in this case if autoreconnect is false, (which is the default), the processing stops. If, in the event of a disconnection, we want the processor to continuously try to reconnect until the connection is restored, this parameter must be set to true when building it.

For events and frames, the timestamp that can be extracted from the stream, or the time when the frame arrives for processing in the StreamProcessor.

Accepted protocols: HTTP, HTTPS and RTSP

Examples:

"rtsp://[USER]:[PASSWD]@[IP]:[PORT]" -> rtsp://192.168.0.50/stream/jpeg
[http://\[URL\]](http://[URL]) -> <http://192.168.0.50:9901/video.mjpeg>

4.2. 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, we 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 we will not lose the event because of this.

It is important to buffer the display of frames in this mode and synchronize them based on the timestamp, so that when the buffer is full, the display is not interrupted.

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://\[path\]](#) vagy "file:[path]"

In case of relative path:

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

Absolute path on Linux:

file:///home/user/videos/cars.mp4

Absolute path on Windows:

"file://C:\\Program Files\\videos\\cars.mp4"

5. A MINIMAL APPLICATION EXAMPLE

This chapter presents the necessary processes, which are the minimum to be able to start processing a stream using the CARMEN® VideoSDK.

5.1. CREATE ANPR

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

5.1.1. 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);
```

5.1.2. C++

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

5.1.3. C#

```
Anpr anpr = Anpr.Builder()  
    .Build();
```

5.2. ONEEVENTCALLBACK

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: [Hiba! A hivatkozási forrás nem található..](#)

5.2.1. 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);
}
```

5.2.2. 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) {
    std::cout << "-----" <<
std::endl;
    std::cout << "Plate: " << event.vehicle().plate().text() << std::endl;
    std::cout << "Country: " << event.vehicle().plate().country() << std::endl;

    std::cout << std::endl;
}
```

5.2.3. C#

In C# callback the function parameter is a Carmen.Event, you should use *try-catch* block around your code.

```
static void EventHandlerCallback(Event e)
{
    Console.WriteLine("-----");
    Console.WriteLine("Plate: " + e.Vehicle.Plate.Text);
    Console.WriteLine("Country: " + e.Vehicle.Plate.Country);

    Console.WriteLine();
}
```

5.3. CREATE STREAMPROCESSOR OBJECT

You also need to use builder pattern to create a StreamProcessor object. In these examples will be only a minimal implementation, other options are [Hiba! A hivatkozási forrás nem található.](#)

5.3.1. 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);
```

5.3.2. C++

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

5.3.3. C#

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

5.4. 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.

5.4.1. 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);
```

5.4.2. C++

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

5.4.3. C#

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

6. CONSTRUCTING COMPONENTS

6.1. ANPR BUILDER

6.1.1. TYPE

It sets the type of ANPR, currently it can only be LOCAL.
Default value is LOCAL.

- C

```
cm_anprbuilder_set_type(CM_ANPR_BUILDER handle, enum CM_ANPRTYPE type)
```

- C++

```
type(AnprType type)
```

- C#

```
Type(AnprType type)
```

6.1.2. LOCALCONCURRENCYLIMIT

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

- C

```
cm_anprbuilder_set_local_concurrency_limit(CM_ANPR_BUILDER handle, int nConcurrent);
```

- C++

```
localConcurrencyLimit(int nConcurrent)
```

- C#

```
LocalConcurrencyLimit(int nConcurrent)
```

6.2. MMR BUILDER

6.2.1. TYPE

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

- C:

```
cm_mmrbuilder_set_type(CM_MMR_BUILDER handle, enum CM_MMR_TYPE type)
```

- C++:

```
type(Type type)
```

- C#:

```
Type(MmrType type)
```

6.3. STREAMPROCESSOR BUILDER

6.3.1. SOURCE

This 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 [Hiba! A hivatkozási forrás nem található.](#)

- C:

```
cm_streamprocessorbuilder_set_source(CM_STREAM_PROCESSOR_BUILDER handle, const char* source)
```

- C++:

```
source(const std::string& source)
```

- C#:

```
Source(string source)
```

6.3.2. NAME

This variable determines the stream name.
Default value is: "Untitled".

- C:

```
cm_streamprocessorbuilder_set_name(CM_STREAM_PROCESSOR_BUILDER handle, const char* name)
```

- C++:

```
name(const std::string& name)
```

- C#:

```
Name(string name)
```

6.3.3. REGION

The region code determines the ANPR and MMR engines. There is a table about it ([Hiba! A hivatkozási forrás nem található.](#)).

You must give the region code! It is necessary to have the given engine installed and to have a license for it.

From MMR, if there is no installation from the given region, it tries to MMR with GEN engine.

- C:

```
cm_streamprocessorbuilder_set_region(CM_STREAM_PROCESSOR_BUILDER handle, const char* region)
```

- C++:

```
region(const std::string& region)
```

- C#:

```
Region(string region)
```

6.3.4. LOCATION

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

- C:

```
cm_streamprocessorbuilder_set_location(CM_STREAM_PROCESSOR_BUILDER handle, const char* location)
```

- C++:

```
location(const std::string& location)
```

- C#:

```
Location(string location)
```

6.3.5. ANPR

It determines which ANPR resource used to license plate recognition. An ANPR object is built with AnprBuilder. (Hiba! A hivatkozási forrás nem található.)

Stream has to have an ANPR object in order to be able to work.

- C:

```
cm_streamprocessorbuilder_set_anpr(CM_STREAM_PROCESSOR_BUILDER handle, CM_ANPR anpr)
```

- C++:

```
anpr(anpr::Anpr& anpr)
```

- C#:

```
Anpr(Carmen.Anpr.Anpr anpr)
```

6.3.6. MMR

It determines which MMR resource used to 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.

- C:

```
cm_streamprocessorbuilder_set_mmr(CM_STREAM_PROCESSOR_BUILDER handle, CM_MMR mmr)
```

- C++:

```
mmr(mmr::Mmr& mmr)
```

- C#:

```
Mmr(Carmen.Mmr.Mmr mmr)
```


6.3.7. 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*.

- C:

```
cm_streamprocessorbuilder_set_auto_reconnect(CM_STREAM_PROCESSOR_BUILDER handle,  
CM_BOOL reconnect)
```

- C++:

```
autoReconnect(bool reconnect)
```

- C#:

```
AutoReconnect(bool reconnect)
```

6.3.8. 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*.

- C:

```
cm_streamprocessorbuilder_set_anpr_color_recognition(CM_STREAM_PROCESSOR_BUILDER  
handle, CM_BOOL color)
```

- C++:

```
anprColorRecognition(bool color)
```

- C#:

```
AnprColorRecognition(bool color)
```

6.3.9. 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*.

- C:

```
cm_streamprocessorbuilder_set_mmr_color_recognition(CM_STREAM_PROCESSOR_BUILDER handle, CM_BOOL color)
```

- C++:

```
mmrColorRecognition(bool color)
```

- C#:

```
MmrColorRecognition(bool color)
```

6.3.10. 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*).

- C:

```
cm_streamprocessorbuilder_set_event_timeout(CM_STREAM_PROCESSOR_BUILDER handle, unsigned long long eventTimeout)
```

- C++:

```
eventTimeout(std::chrono::milliseconds eventTimeoutMs)
```

- C#:

```
EventTimeout(ulong eventTimeout)
```

6.3.11. ROI (REGION OF INTEREST)

This parameter determines a convex quadrilateral shape. Triggering will only happen inside this area, so if the number plate is out of the ROI, that is not going to generate an event. MMR is always working on the full image, but it is based on the number plate caught. Less area in ROI means less processing time. This function needs an array of 4 points. The points' order can be clockwise or counterclockwise and coordinates are normalized to [0.0, 1.0]. (0.0, 0.0) is equal to (0, 0) px of the image (top left corner) and (1.0, 1.0) is equal to (image.width - 1, image.height - 1) px of the image (bottom right corner).

Default parameter is same as full frame $((0, 0), (1, 0), (1, 1), (0, 1))$.

- C:

```
cm_streamprocessorbuilder_set_roi(CM_STREAM_PROCESSOR_BUILDER handle, const
struct CM_POINT* points, unsigned int size)
```

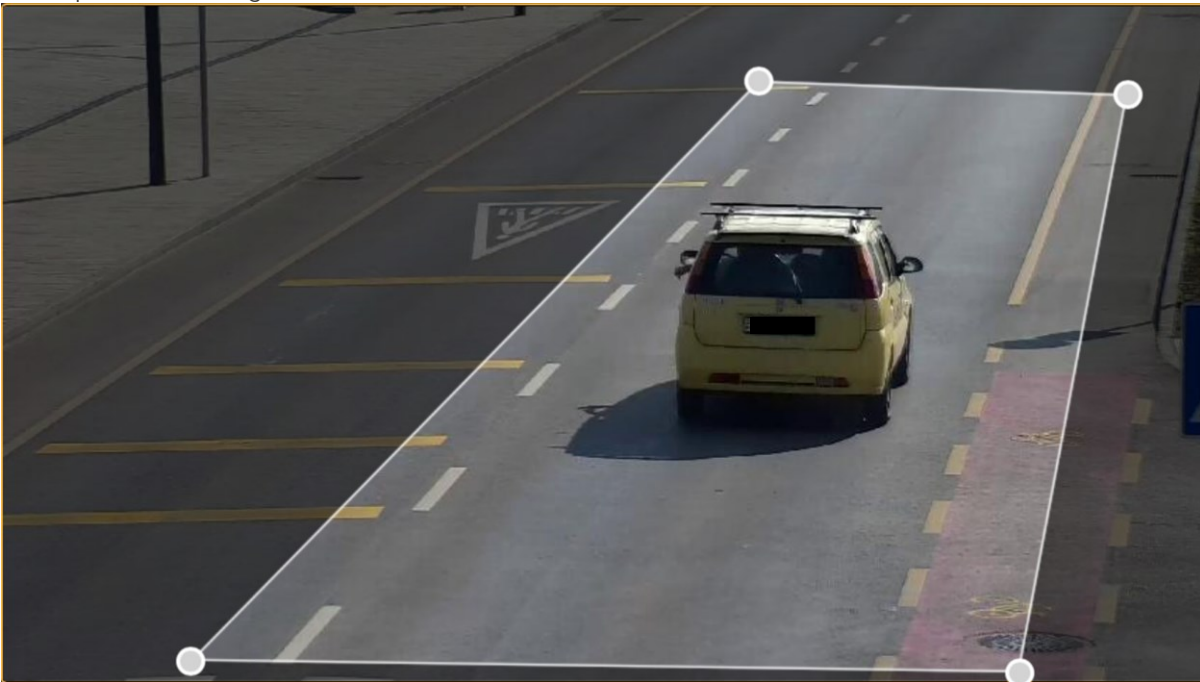
- C++:

```
roi(const std::vector<Point>& polygon)
```

- C#:

```
Roi(List<Carmen.Point> roi)
```

Example of a setting:



ROI is: $[(0.15635, 0.97063), (0.629019, 0.111032), (0.936093, 0.131089), (0.846061, 0.987822)]$.

- C:

```
CM_POINT roi[4] = {{0.15635, 0.97063}, {0.629019, 0.111032}, {0.936093,
0.131089}, {0.846061, 0.987822}};
cm_streamprocessorbuilder_set_roi(builder, roi, 4);
```

- C++:

```
.roi({{0.15635, 0.97063},{0.629019, 0.111032},{0.936093, 0.131089},{0.846061, 0.987822}})
```

- C#:

```
.Roi(new List<Carmen.Point>{
    new Carmen.Point(0.15635, 0.97063),
    new Carmen.Point(0.629019, 0.111032),
    new Carmen.Point(0.936093, 0.131089),
    new Carmen.Point(0.846061, 0.987822)
})
```

6.3.12. EVENT CALLBACK

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

Examples are in 0 section.

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

- C:

```
cm_streamprocessorbuilder_set_event_callback(CM_STREAM_PROCESSOR_BUILDER handle,
void fn(CM_EVENT*, void*), void* userdata, void(*userdata_free)(void*))
```

- C++:

```
eventCallback(std::function<void(const Event&)> fn)
```

- C#:

```
EventCallback(EventCB callback)
```

6.3.13. 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.

- C:

```
cm_streamprocessorbuilder_set_onframe_callback(CM_STREAM_PROCESSOR_BUILDER
handle, void fn(CM_IMAGE_PROXY*, void*), void* userdata,
void(*userdata_free)(void*))
```

- C++:

```
onFrameCallback(std::function<void(const ImageProxy&)> fn)
```

- C#:

```
OnFrameCallback(OnFrameCB callback)
```

Example for a callback which saves the frame to the filesystem:

(You should use unique image file names when running streams simultaneously.)

- C:

```
void onFrameCallback(CM_IMAGE_PROXY* ip) {
    Image* image;
    cm_imageProxyCloneImage(&image, ip);
    cm_imageSave(image, "frame.jpg", JPEG);
    cm_imageFree(image);

    cm_imageproxy_free(ip);
}
```

- C++:

```
void onFrameCallback(const cm::ImageProxy& ip) {
    try {
        ip.image()->save("frame.jpg", cm::FileFormat::JPEG);
    } catch(...) {
        std::cout << "Exception in frame callback" << std::endl;
    }
}
```

- C# with lambda in builder setter:

```
.OnFrameCallback
(
    (frame) =>
    {
        try
        {
            frame.Image.Save("frame.jpeg", ImageFormat.Jpeg);
        }
        catch (Exception ex)
        {
            Console.Error.WriteLine("Exception in frame callback. " + ex);
        }
    }
)
```

6.3.14. 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.

- C:

```
cm_streamprocessorbuilder_set_status_change_callback(CM_STREAM_PROCESSOR_BUILDER
handle, void fn(CM_STREAM_PROCESSOR, enum CM_StreamProcessorStatus, void*),
void* userdata, void(* userdata free)(void*))
```

- C++:

```
statusChangeCallback(std::function<void(StreamProcessor& stream,
StreamProcessorStatus)> fn)
```

- C#:

```
StatusChangeCallback(StatusChangeCB callback)
```

Examples:

- C:

```
void onStatusCallbacks(CM_STREAM_PROCESSOR handle, enum CM_StreamProcessorStatus
status) {
    printf("Stream \"%s\" (%s) status changed to: %i\n",
cm_streamprocessor_name(handle), cm_streamprocessor_sessionid(handle), status);
}
```

- C++:

```
void statusCallback(cm::video::StreamProcessor& stream, const
cm::video::StreamProcessorStatus& s) {
    std::cout << "StatusCallback: in stream: \"" << stream.name() << "\" (" <<
stream.sessionId() << ") status is: " << static_cast<int>(s) << std::endl;
}
```

- C# with lambda in builder setter:

```
.StatusChangeCallback
(
    (stream1, status) =>
    {
        Console.WriteLine("Stream: \"" + stream1.Name + "\" (" +
stream1.SessionId + ") status is: " +
status);
    }
)
```

6.4. 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 logging to the standard output: “[*LOG_LEVEL*]: *MESSAGE*”.

6.4.1. CHANGING LOGGING LEVEL

There is example of setting log levels in the 2nd and 3rd sample codes in each program language.

- C:

```
cm_logger_set_global_logger_min_level(CM_LOGGER_LEVEL_WARNING);
```

- C++:

```
cm::log::GlobalLogger::setMinLevel(cm::log::LogLevel::WARNING);
```

- C#:

```
GlobalLogger.setMinLevel(LogLevel.Warning);
```

6.4.2. SET LOGGING CALLBACK

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

- C:

```
void log_func(const char* message, int message_size, int log_level, void*
userdata) {
    printf("MY LOGGER [%d]: %s %s\n", log_level, message, (const
char*)userdata);
}

void logger_cleanup_func(void* cleanup_info) {
    printf("MY LOGGER CLEANUP %s\n", (const char*)cleanup_info);
}

// In the main code:

CM_LOG_CALLBACK_PARAMS logger;
memset(&logger, 0, sizeof(logger));
logger.userdata = "USERDATA"; // this pointer will be one of the parameters
of every 'log_func' call
logger.log_func = log_func;
logger.cleanup_func = logger_cleanup_func;
logger.cleanup_info = "CLEANUP_INFO"; // this pointer will be the parameter
of the 'cleanup_func' callback
cm_logger_set_default_log_callback(&logger);
```

- C++:

```
cm::log::GlobalLogger::setLogCallback( [](std::string_view message,
cm::log::LogLevel logLevel) {
    std::cout << "MyLog: " << message << std::endl;
});
```

- C#:

```
GlobalLogger.setLogCallback((message, level) => { Console.WriteLine("MyLog: " +
message); });
```

In this callback there is option to write logs to file, or turn off the logger.

7. RESULT CLASSES

7.1. IMAGE CLASSES

7.1.1. C

```

/*
 * Plane descriptor struct for CM_IMAGE
 */
typedef struct CM_PLANE {
    uint8_t* data; // data pointer
    long size; // plane size in bytes
    long linestep; // offset between 2 lines in bytes
    void* _priv; // private pointer for implementation
} CM_PLANE;

```

```

/*
 * Pixel formats for CM_IMAGE
 */
typedef enum CM_PIXEL_FORMAT {
    YUV420P = 0, // planar YUV 4:2:0, 12bpp, (1 Cr & Cb sample per 2x2 Y
samples)
    RGB24 = 2, // packed RGB 8:8:8, 24bpp, RGBRGB...
    BGR24 = 4, // packed RGB 8:8:8, 24bpp, BGRBGR...
    GRAY8 = 9 // Y, 8bpp
} CM_PIXEL_FORMAT;

```

```

/*
 * Represents a bitmap in memory
 */
typedef struct CM_IMAGE {
    long width; // width of image in pixels
    long height; // height of image in pixels
    CM_PIXEL_FORMAT format; // pixel format of the image
    CM_PLANE* planes; // array of planes
    int planeSize; // number of planes
} CM_IMAGE;

```

```

/*
 * Frees the image resource.
 */
int cm_image_free(CM_IMAGE* image);

```

```

/*
 * Metadata of an image
 */
typedef struct CM_IMAGE_INFO {
    long long index; // frame index in a video
    long long timestamp; // epoch unix timestamp in ms
} CM_IMAGE_INFO;

```

```
/*
 * Image wrapper used in the SDK's callback parameters.
 * The image data can be obtained by getting a CM_IMAGE descriptor with the
 cm_imageproxy_clone_image function.
 *
 * It has a similar interface of an image with extra metadata.
 */
typedef struct CM_IMAGE_PROXY {
    long width; // width of image in pixels
    long height; // height of image in pixels
    CM_PIXEL_FORMAT format; // pixel format of the image
    CM_IMAGE_INFO info; // metadata structure
    CM_IMAGE_HANDLE _handle; // private pointer for implementation
} CM_IMAGE_PROXY;
```

```
/*
 * Clones the image resource if the image still exists behind the proxy.
 */
int cm_imageproxy_clone_image(CM_IMAGE** image, const CM_IMAGE_PROXY* proxy);

/*
 * Clones only the image proxy.
 */
int cm_imageproxy_clone(CM_IMAGE_PROXY** newProxy, const CM_IMAGE_PROXY* proxy);

/*
 * Frees the image proxy resource.
 */
int cm_imageproxy_free(CM_IMAGE_PROXY* proxy);
```



7.1.2. C++

```

/*
 * Plane descriptor struct for cm::Image
 */
struct cm::PlaneView {
    uint8_t* data = nullptr;    // data pointer
    std::size_t size = 0;      // plane size in bytes
    std::size_t linestep = 0;  // offset between 2 lines in bytes
};

```

```

/*
 * Represents a bitmap in memory
 */
class cm::Image {
public:

    // ...

    std::size_t width() const;    // width of image in pixels
    std::size_t height() const;  // height of image in pixels
    PixelFormat format() const;  // pixel format of the image

    std::vector<PlaneView> planes() const; // array of planes

    // ...
};

```

```

/*
 * Metadata of an image
 */
class cm::ImageInfo {
public:

    // ...

    long long index() const;      // frame index in a video
    long long timestamp() const;  // epoch unix timestamp in ms

    // ...
};

```

```

/*
 * Image wrapper used in the SDK's callback parameters.
 * The image data can be obtained by getting a cm::Image descriptor with the
 * image() function.
 *
 * It has a similar interface as an image with extra metadata.
 */
class cm::ImageProxy {
public:

    // ...

    std::size_t width() const;    // width of image in pixels

```

Page 59/121



```
std::size_t height() const;           // height of image in pixels
PixelFormat format() const;          // pixel format of the image
std::shared_ptr<cm::Image> image() const; // clones the image resource if
the image still exists behind the proxy (it uses cache if cloned once)
ImageInfo imageInfo() const;         // metadata structure

// ...

};
```



7.1.3. C#

In Carmen namespace:

```
public enum PixelFormat
{
    YUV420P = 0,    //< planar YUV 4:2:0, 12bpp, (1 Cr & Cb sample per 2x2 Y
samples)
    RGB24 = 2,      //< packed RGB 8:8:8, 24bpp, RGBRGB...
    BGR24 = 4,      //< packed RGB 8:8:8, 24bpp, BGRBGR...
    GRAY8 = 9       //<          Y          , 8bpp
};
```

```
/*
 * Metadata of an image
 */
public class ImageInfo
{
    int Index;           // frame index in a video
    System.DateTime DateTime; // epoch unix timestamp in ms
    int Timestamp;      // timestamp in UTC 0 timezone
};
```

```
/*
 * Image wrapper used in the SDK's callback parameters.
 * The image data can be obtained by getting a System.Drawing.Bitmap descriptor
with the Image property.
 *
 * It has a similar interface as an image with extra metadata.
 */
public class ImageProxy
{
    // ...

    int Width;           // width of image in pixels
    int Height;          // height of image in pixels
    Carmen.PixelFormat Format; // pixel format of the image

    System.Drawing.Bitmap Image; // clones the image resource if the image
still exists behind the proxy (it uses cache if cloned once)
    Carmen.ImageInfo ImageInfo; // metadata structure

    // ...
};
```

7.2. ANPR RESULT CLASSES

7.2.1. C

```

/*
 * License plate result descriptor
 */
typedef struct CM_PLATE {
    char* text;           // text of the license plate in UTF-8 encoding
    char* country;       // nationality of the license plate in ISO-3166-1 alpha-
3 with extended codes
    char* state;         // state code of the license plate
    char* category;      // license plate category
    int type;            // type value used by Carmen ANPR
    int textColor;       // main font color
    int bgColor;        // main background color
    int stripColor;      // Dedicated Area color if exists, otherwise main
background color
    int plateTypeRaw;    // unused, deprecated, will be removed
    int plateSize;      // license plate frame width in millimeters
} CM_PLATE;

```

```

/*
 * Information related to the license plate recognition process on an image
 */
typedef struct CM_PLATE_DETECTION {
    CM_PLATE plate;      // properties of the license plate
    CM_POINT* polygon;  // the license plate frame's polygon described by pixel
coordinates
    int polygonSize;    // number of points of the license plate frame's polygon
    float confidence;   // confidence of the recognition process's result
} CM_PLATE_DETECTION;

```

7.2.2. C++

```

/*
 * License plate result descriptor
 */
class cm::anpr::Plate {
public:

    // ...

    const std::string& text() const;           // text of the license plate in UTF-
8 encoding
    const std::string& country() const;       // nationality of the license plate
in ISO-3166-1 alpha-3 with extended codes
    const std::string& state() const;         // state code of the license plate

    int type() const;                         // type value used by Carmen ANPR
    int plateTypeRaw() const;                 // unused, deprecated, will be
removed
    const std::string& category() const;      // license plate category
    int plateSize() const;                    // license plate frame width in
millimeters

    int textColor() const;                    // main font color
    int bgColor() const;                      // main background color
    int stripColor() const;                   // Dedicated Area color if exists,
otherwise main background color

    // ...

};

```

```

/*
 * Information related to the license plate recognition process on an image
 */
class cm::anpr::PlateDetection {
public:

    // ...

    const Plate& plate() const;               // properties of the license
plate
    const std::vector<Point>& polygon() const; // the license plate frame's
polygon described by pixel coordinates

    float confidence() const;                 // confidence of the recognition
process's result

    // ...

};

```

7.2.3. C#

In Carmen.Anpr namespace:

```

/*
 * License plate result descriptor
 */
public class Plate{

    // ...

    String Text;           // text of the license plate in UTF-8 encoding
    String Country;       // nationality of the license plate in ISO-3166-1 alpha-3
with extended codes
    String State;        // state code of the license plate
    String Category;    // license plate category
    int Type;           // type value used by Carmen ANPR
    Color TextColor;    // main font color
    Color BgColor;      // main background color
    Color StripColor;   // Dedicated Area color if exists, otherwise main
background color
    int PlateTypeRaw;   // unused, deprecated, will be removed
    int PlateSize;     // license plate frame width in millimeters

    // ...

};

```

```

/*
 * Information related to the license plate recognition process on an image
 */
public class PlateDetection{

    // ...

    Carmen.Anpr.Plate Plate;    // properties of the license plate
    List<Point> Polygon;        // the license plate frame's polygon described
by pixel coordinates
    float Confidence;          // confidence of the recognition process's
result

    // ...

};

```


7.3. MMR RESULT CLASSES

7.3.1. C

```
/*
 * Attributes of a vehicle
 */
typedef struct CM_MMR_DATA {
    char* category; // vehicle category (e.g. BUS, CAR, etc.)
    char* make;     // vehicle make (e.g. TOYOTA, TESLA, etc.)
    char* model;   // vehicle model (e.g. COROLLA, YARIS, etc.)
    int color;     // vehicle color (RGB)
} CM_MMR_DATA;
```

```
/*
 * Information related to the Make&Model Recognition process
 */
typedef struct CM_MMR_DETECTION {
    CM_MMR_DATA mmr; // Make & Model recognition result

    float makeConfidence; // make confidence [0.0 - 1.0]
    float modelConfidence; // model confidence [0.0 - 1.0]
    float colorConfidence; // color confidence [0.0 - 1.0]
    float categoryConfidence; // category confidence [0.0 - 1.0]
} CM_MMR_DETECTION;
```



7.3.2. C++

```
/*
 * Attributes of a vehicle
 */
class cm::mmr::MmrData {
public:

    // ...

    const std::string& category() const;    // vehicle category (e.g. BUS, CAR,
etc.)
    const std::string& make() const;       // vehicle make (e.g. TOYOTA, TESLA,
etc.)
    const std::string& model() const;     // vehicle model (e.g. COROLLA,
YARIS, etc.)
    int color() const;                    // vehicle color (RGB)

    // ...

};
```

```
/*
 * Information related to the Make&Model Recognition process
 */
class MmrDetection {
public:

    // ...

    const MmrData& mmrData() const;       // Make & Model recognition result

    float makeConfidence() const;        // make confidence [0.0 - 1.0]
    float modelConfidence() const;       // model confidence [0.0 - 1.0]
    float colorConfidence() const;       // color confidence [0.0 - 1.0]
    float categoryConfidence() const;    // category confidence [0.0 - 1.0]

    // ...

};
```



7.3.3. C#

In Carmen.Mmr namespace:

```
/*  
 * Attributes of a vehicle  
 */  
public class MmrData{  
  
    // ...  
  
    String Category; // vehicle category (e.g. BUS, CAR, etc.)  
    String Make; // vehicle make (e.g. TOYOTA, TESLA, etc.)  
    String Model; // vehicle model (e.g. COROLLA, YARIS, etc.)  
    Color Color; // vehicle color (RGB)  
  
    // ...  
  
};
```

```
/*  
 * Information related to the Make&Model Recognition process  
 */  
public class MmrDetection {  
  
    // ...  
  
    Carmen.Mmr.MmrData MmrData; // Make & Model recognition result  
  
    float MakeConfidence; // make confidence [0.0 - 1.0]  
    float ModelConfidence; // model confidence [0.0 - 1.0]  
    float ColorConfidence; // color confidence [0.0 - 1.0]  
    float CategoryConfidence; // category confidence [0.0 - 1.0]  
  
    // ...  
  
};
```

7.4. EVENT CLASSES

7.4.1. C

```

/*
 * Information related to the license plate recognition process on an image with
 a reference to that image
 */
typedef struct CM_PLATE_ON_IMAGE {
    CM_PLATE_DETECTION detection; // properties of the license plate
    CM_IMAGE_PROXY* image; // reference to the image
} CM_PLATE_ON_IMAGE;

```

```

/*
 * Information related to the Make&Model Recognition process on an image with a
 reference to that image
 */
typedef struct CM_MMR_ON_IMAGE {
    CM_MMR_DETECTION detection; // Make & Model recognition result
    CM_IMAGE_PROXY* image; // reference to the image
} CM_MMR_ON_IMAGE;

```

```

/*
 * Vehicle descriptor
 */
typedef struct CM_VEHICLE {
    CM_PLATE* plate; // the license plate of the vehicle
    CM_MMR_DATA* mmrData; // attributes of the vehicle
} CM_VEHICLE;

```

```

/*
 * Result of processing one vehicle passage
 */
typedef struct CM_EVENT {
    char* uuid; // unique id of the event
    char* channelName; // name of the channel which contained
the passage
    char* channelSessionId; // the id of the channel which contained
the passage
    float confidence; // calculated confidence to the whole
event processing
    long long timestamp; // the epoch unix timestamp of the
passage in milliseconds
    CM_VEHICLE* vehicle; // calculated information of the vehicle

    CM_PLATE_ON_IMAGE* plateDetections; // all license plate detection results
from the passage
    int numPlateDetections; // number of license plate detection
results
    CM_MMR_ON_IMAGE* mmrDetections; // all Make & Model recognition results
from the passage
    int numMmrDetections; // number of Make & Model recognition
results
    CM_IMAGE_PROXY* images; // all images saved from the passage
    int imagesSize; // number of images saved from the
passage
} CM_EVENT;

```

Page 68/121



```
/*  
 * Frees the event resource.  
 */  
CMV_C_SDK_DLL_API int cm_event_free(CM_EVENT* event);
```



7.4.2. C++

```

/*
 * Information related to the license plate recognition process on an image with
 a reference to that image
 */
class cm::Event::PlateOnImage {
public:

    // ...

    const cm::anpr::PlateDetection& detection() const; // properties of the
license plate
    std::shared_ptr<cm::ImageProxy> image() const; // reference to the
image

    // ...
};

```

```

/*
 * Information related to the Make&Model Recognition process on an image with a
 reference to that image
 */
class cm::Event::MmrOnImage {
public:

    // ...

    const cm::mmr::MmrDetection& detection() const; // Make & Model recognition
result
    std::shared_ptr<cm::ImageProxy> image() const; // reference to the image

    // ...
};

```

```

/*
 * Result of processing one vehicle passage
 */
class cm::Event {
public:

    // ...

    const cm::Vehicle& vehicle() const; // calculated information of
the vehicle

    const std::string& uuid() const; // unique id of the event
    const std::string& channelName() const; // name of the channel which
contained the passage
    const std::string& channelId() const; // the id of the channel
which contained the passage

    long long timestamp() const; // the epoch unix timestamp
of the passage in milliseconds
    float confidence() const; // calculated confidence to
the whole event processing
};

```

```
    const std::vector<PlateOnImage>& plateDetections() const;           // all
license plate detection results from the passage
    const std::vector<MmrOnImage>& mmrDetections() const;           // all
Make & Model recognition results from the passage
    const std::vector<std::shared_ptr<cm::ImageProxy>>& images() const; //
images from the passage

    // ...

};
```



7.4.3. C#

In Carmen namespace:

```

/*
 * Vehicle descriptor
 */
public class Vehicle {

    // ...

    Carmen.Anpr.Plate Plate;           // the license plate of the vehicle
    Carmen.Mmr.MmrData Attributes;     // attributes of the vehicle

    // ...

};

```

```

/*
 * Result of processing one vehicle passage
 */
public class Event {

    /*
     * Information related to the license plate recognition process on an image
     with a reference to that image
     */
    class PlateOnImage {

        // ...

        Carmen.Anpr.PlateDetection Detection; // properties of the license
plate
        ImageProxy Image;                    // reference to the image
    };

    /*
     * Information related to the Make&Model Recognition process on an image with
     a reference to that image
     */
    class MmrOnImage {

        // ...

        Carmen.Mmr.MmrDetection Detection; // Make & Model recognition result
        ImageProxy Image;                 // reference to the image
    };

    // ...

    String Uuid;                          // unique id of the event
    String ChannelName;                    // name of the channel which contained the
passage
    String ChannelSessionId;              // the id of the channel which contained the
passage

    long Timestamp;                        // the epoch unix timestamp of the passage in
milliseconds
    System.DateTime DateTime;              // the timestamp of the passage in UTC 0
timezone
    float Confidence;                      // calculated confidence to the whole event

```

Page 72/121




```
processing

    Vehicle Vehicle;           // calculated information of the vehicle

    List<PlateOnImage> PlateDetections; // all license plate detection results
from the passage
    List<MmrOnImage> MmrDetections;    // all Make & Model recognition results
from the passage
    List<ImageProxy> Images;          // images from the passage

    // ...

};
```



8. 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



9. KNOWN ISSUES

- All C functions return with error code -1 if error occurs inside. Later there will be more dedicated error codes



ADI DEMO

1. INTRODUCTION

The ANPR Demo for Images (ADI) is a program that was developed by Adaptive Recognition Hungary 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: **7.4.0.19**

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®](#).

! Important

Please note, that this application is not available for ARM and CentOS6 packages 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 three separate sections; the **Result Image**, the **Result Tree** 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).

ANPR Result Tree

- > FOD0121
- > EUI9755
- > EZJ0074

#	Path	Plate Text	Text (AS)	Plate category	Country/State	Confidence	TypeID	ram	racter	Text Background	Text Color	Dedicated Area Color	ANPR Time(ms)	Make&Model	Category	Color	ViewPoint	MMR confidence	MMR Time(ms)
1	C:/l...	FOD0121	FOD0121	COMMON	BRA	44	657027	((1...	29	(255,255,255)	(0,0,0)	(255,255,255)	226	Chevrolet Camaro	Car	YELLOW	front	79	219
2	C:/l...	EUI9755	EUI9755	COMMON	BRL <small>Brazil</small>	59	657027	((3...	27	(255,255,255)	(0,0,0)	(255,255,255)	37	VW Polo	Car	GRAY	front	99	163
3	C:/l...	EZJ0074	EZJ0074	COMMON	BRA	60	657027	((3...	30	(255,255,255)	(0,0,0)	(255,255,255)	38	Mitsubishi Outl...	Car	BLUE	front_side	99	89

Quick overview of the main screen

In the title you can see the following information:

ADI Demo 7.4.0.18 (64 bit) [ANPR: cmanpr-7.3.15.14 : bgd] - [MMR: OFF] - [License: local]

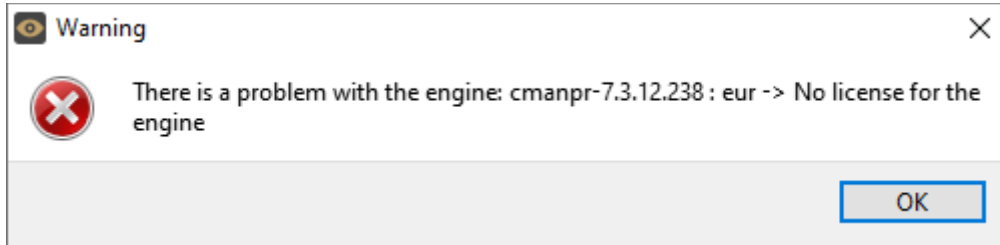
ANPR: the currently used ANPR engine region and version

MMR: the currently used MMR engine region and version

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

Note

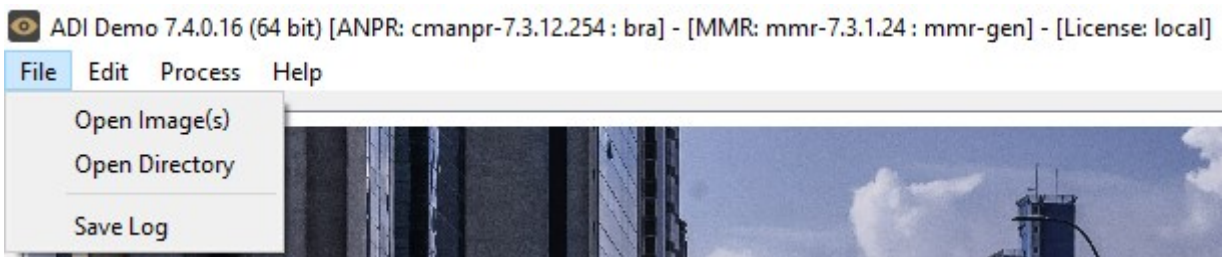
In case of missing hardware key, the following error message will appear:



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

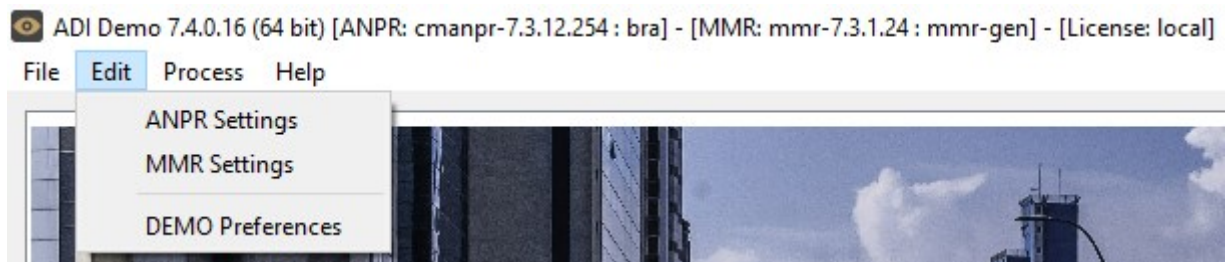
Click to specify a directory containing images for LPR processing.

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:



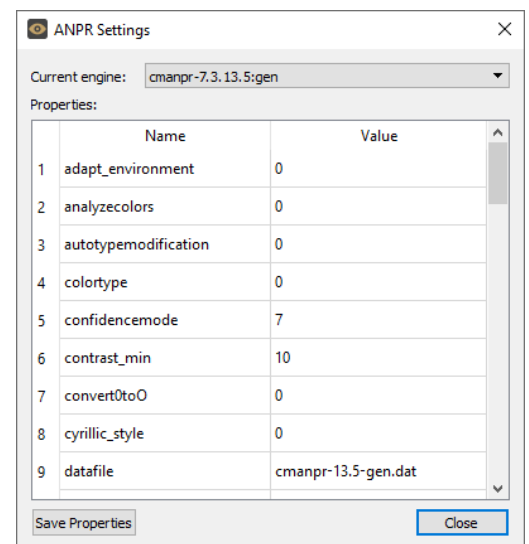
4.1. ANPR / MMR SETTINGS

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.

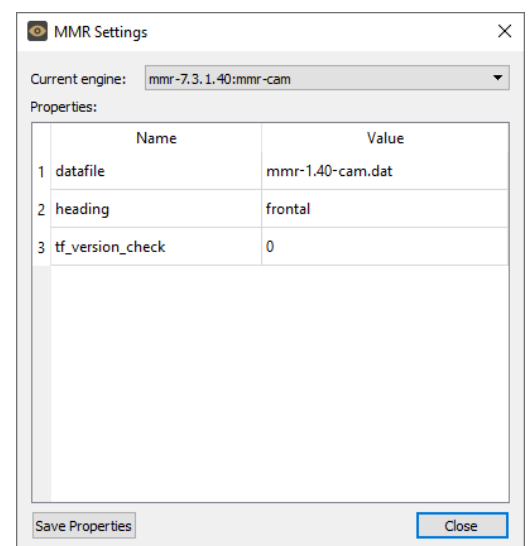
Properties: Lists the property values for the selected engine. To adjust the pre-set values, click on one of the value fields and then input a desired value. A full description of the engine properties can be found in the [CARMEN® ANPR Reference Manual](#).



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. 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.

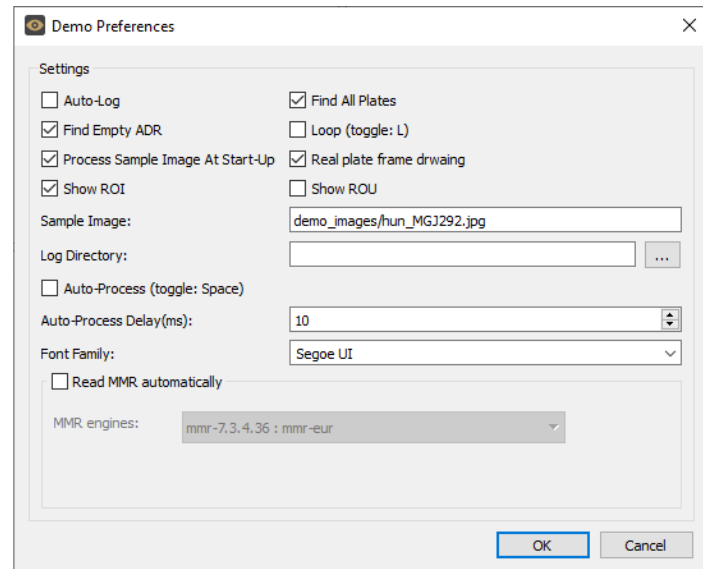
Sample Image:

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

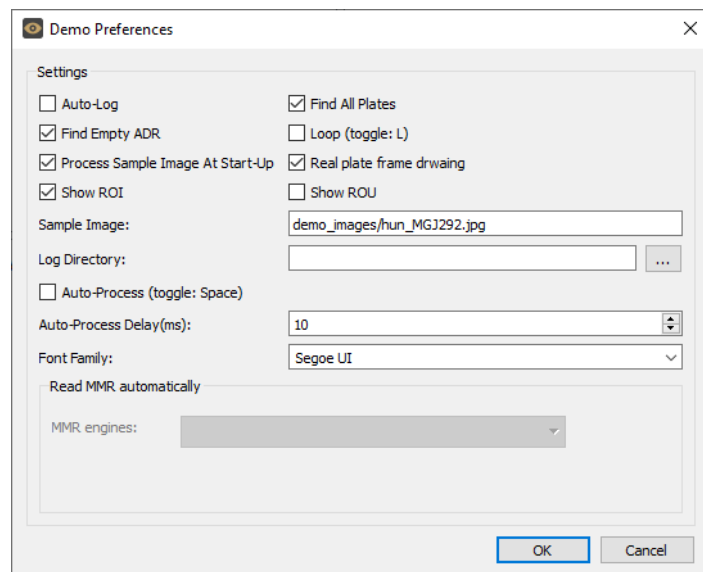
Log Directory:

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 installed

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.

Font Family: Select the font style of the Result Tree (LPR details in the upper right corner of the ADI screen).

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

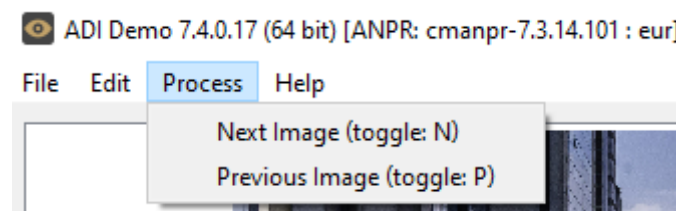
If checked, ADI demo will give you back the MMR results automatically.

Note

MMR engines need separate MMR licenses and must be the same region as ANPR license to be able to work together. GEN MMR engine can work with every ANPR engines (except GEN ANPR engine, which is not capable to run with any MMR engine)

5. PROCESS MENU

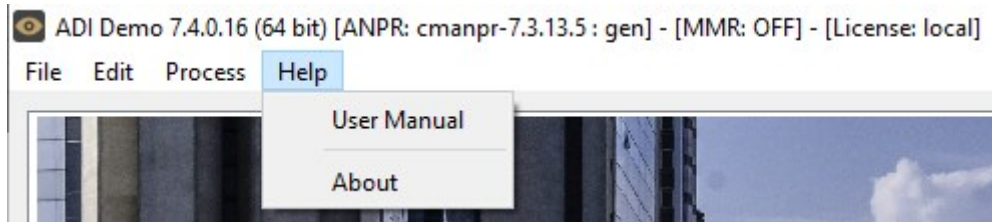
The **Process** menu contains the following menu items:



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.

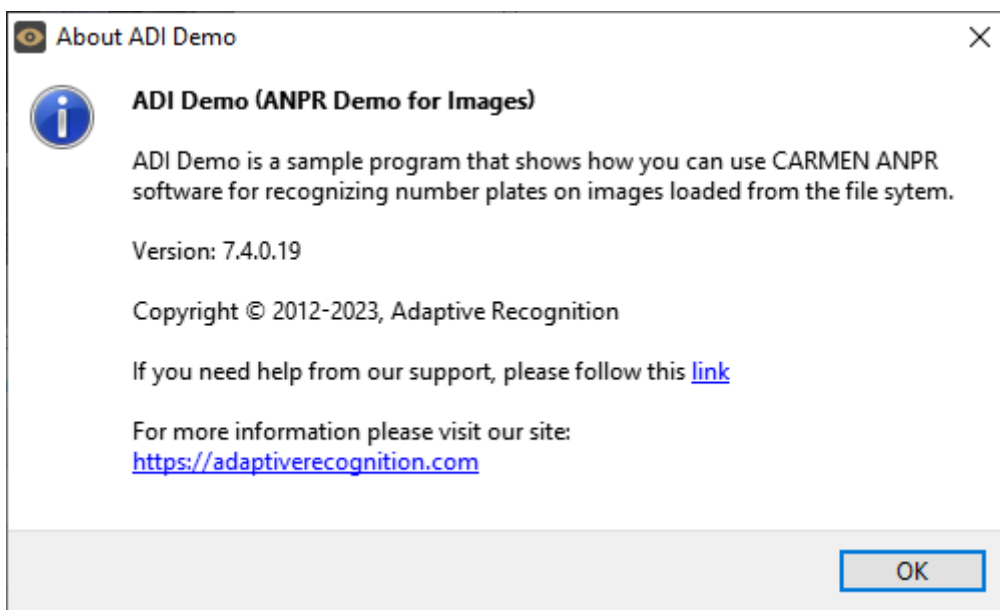
6. HELP MENU

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

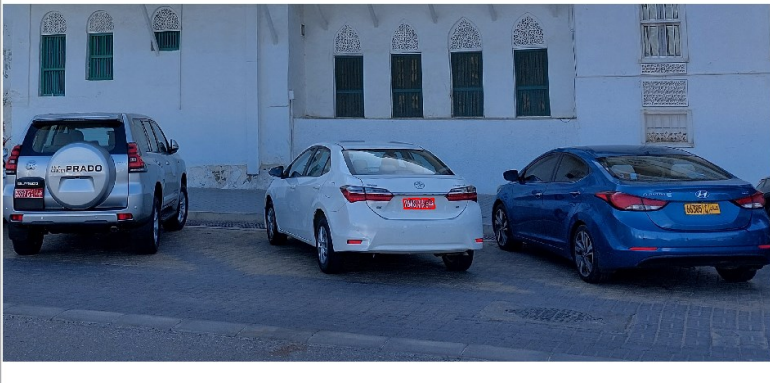
Note

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:

ADI Demo 7.4.0.19 (64 bit) [ANPR: cmanpr-7.3.15.162 : world] - [MMR: OFF] - [License: Local NNC]
File Edit Process Help




ANPR Result Tree

- 7648TA
 - Text(ASCII): 7648TA
 - Plate category: COMMON
 - Confidence: 91
 - Country/State: Oman
 - TypeID: 215096
 - Color(RGB): (255,255,255)
 - BkColor(RGB): (0,0,0)
 - Frame: ((1324,644),(1424,643),(1425,674),(1325,675))
 - > Characters (...)
 - > Tips
- 66385M
 - Text(ASCII): 66385M
 - Plate category: COMMON
 - Confidence: 95
 - Country/State: Oman
 - TypeID: 215053
 - Color(RGB): (0,0,0)

#	Path	Plate Text	Plate Text (ASCII)	Plate category	Country/State	Confidence	TypeID	Frame	Character Size	Text Background	Text Color	Dedicated Area Color	ANPR Time(ms)
1	C:/...	7648TA	7648TA	COMMON	OMN	91	215096	((13...	18	(0,0,0)	(255,255,...	No color	239
2	C:/...	66385M	66385M	COMMON	OMN	95	215053	((22...	22	(255,255,255)	(0,0,0)	No color	151
3	C:/...	5359MS	5359MS	COMMON	OMN	70	215096	((42...	16	(0,0,0)	(255,255,...	No color	207

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.

ADI Demo 7.4.0.19 (64 bit) [ANPR: cmanpr-7.3.15.162 : world] - [MMR: OFF] - [License: Local NNC]
File Edit Process Help



ANPR Result Tree

- 7648TA
 - Text(ASCII): 7648TA
 - Plate category: COMMON
 - Confidence: 91
 - Country/State: Oman
 - TypeID: 215096
 - Color(RGB): (255,255,255)
 - BkColor(RGB): (0,0,0)
 - Frame: ((1324,644),(1424,643),(1425,674),(1325,675))
 - > Characters (...)
 - > Tips
- 66385M
 - Text(ASCII): 66385M
 - Plate category: COMMON
 - Confidence: 95
 - Country/State: Oman
 - TypeID: 215053
 - Color(RGB): (0,0,0)

#	Path	Plate Text	Plate Text (ASCII)	Plate category	Country/State	Confidence	TypeID	Frame	Character Size	Text Background	Text Color	Dedicated Area Color	ANPR Time(ms)
1	C:/...	7648TA	7648TA	COMMON	OMN	91	215096	((13...	18	(0,0,0)	(255,255,...	No color	239
2	C:/...	66385M	66385M	COMMON	OMN	95	215053	((22...	22	(255,255,255)	(0,0,0)	No color	151
3	C:/...	5359MS	5359MS	COMMON	OMN	70	215096	((42...	16	(0,0,0)	(255,255,...	No color	207

9. ANPR RESULT TREE

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

1. License plate result:

Provides a detailed summary of the recognized image:

- **Text(ASCII):** alphanumeric license plate text in ASCII characters
- **Plate type:** category of the license plate
- **Confidence:** overall confidence level (%) of the plate.
- **Country/State:** Country full name / state full name
- **TypeID:** code containing country/state ID of the plate.
- **Color:** color of the dedicated area of the license plate.
- **BkColor:** background color of license plate text.
- **Dedicated Area Color:** color of dedicated area in RGB format
- **Frame:** pixel coordinates of the plate corners.

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
- **MMR Confidence:** overall MMR confidence
- **MMR Time:** MMR processing time in milliseconds

ANPR Result Tree	
~ EUI9755	
Text(ASCII):	EUI9755
Plate catego...	COMMON
Confidence:	59
Country/State:	Brazil
TypeID:	657027
Color(RGB):	(0,0,0)
BkColor(RGB):	(255,255,255)
Dedicated A...	(255,255,255)
Frame:	((317,1754),(485,1749),(484,1799),(315,1803))
Make&model:	VW Polo
Category:	Car
Color:	GRAY
ViewPoint:	front
MMR confid...	99
MMR time:	163
~ Characters (...)	
~ E	
Code:	0x0045
Confide...	99
Color(R...	(0,0,0)
BkColor...	(255,255,255)
Frame:	((324,1771),(342,1771),(341,1797),(323,1798))
> U	
> I	
> 9	
> 7	
> 5	
> 5	
~ Tips	
> E	

2. Characters:

Individual details for each recognized character. Ordered from left to right than up to down.

- **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 result.

- **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 corner



ANPR Result Tree	
▼ EU19755	
Text(ASCII):	EU19755
Plate catego...	COMMON
Confidence:	58
Country/State:	Brazil
TypeID:	657027
Color(RGB):	(0,0,0)
BkColor(RGB):	(255,255,255)
Dedicated A...	(255,255,255)
Frame:	((190,1052),(290,1050),(290,1079),(190,1081))
Make&model:	VW Polo
Category:	Car
Color:	BLUE
ViewPoint:	front
MMR confid...	99
MMR time:	298
> Characters (...)	
> Tips	
▼ EZJ0074	
Text(ASCII):	EZJ0074
Plate catego...	COMMON
Confidence:	60
Country/State:	Brazil
TypeID:	657027
Color(RGB):	(0,0,0)
BkColor(RGB):	(255,255,255)
Dedicated A...	(255,255,255)
Frame:	((1977,1047),(2087,1044),(2090,1077),(1980,1080))
Make&model:	Mitsubishi Outlander Sport ~ ASX

When ADI is not in automatic mode, and it encounters an individual image with multiple plates, it will display all results from that image in the same result tree (if **Find-All-Plates** is checked in Edit -> Demo Preferences).

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:

- **Path:** location of the image file.
- **Plate Text:** alphanumeric license plate text.
- **Plate Text (ASCII):** alphanumeric license plate text in ASCII characters
- **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.
- **Confidence:** overall confidence level of the plate.
- **TypeID:** our internal ID of the currently recognized plate type
- **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 format.
- **Text Color:** character color in RGB format.
- **Dedicated Color:** color of dedicated area in RGB format.
- **ANPR Time(ms):** ANPR processing time in milliseconds.

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 Confidence:** overall MMR confidence
- **MMR Time(ms):** MMR processing time in milliseconds

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

#	Path	Plate Text	Plate Text (ASCII)	Plate category	Country/State	Confidence	TypeID	Frame	Character	Text Background	Text Color	Dedicated Area Color	ANPR Time(ms)	Make&Model	Category	Color	ViewPoint	MMR confidence	MMR Time(ms)
1	C:/I...	FOD0121	FOD0121	COMMON	BRA	44	657027	((1...	29	(255,255,255)	(0,0,0)	(255,255,255)	226	Chevrolet Camaro	Car	YELLOW	front	79	219
2	C:/I...	EUI9755	EUI9755	COMMON	BRA	59	657027	((3...	27	(255,255,255)	(0,0,0)	(255,255,255)	37	VW Polo	Car	GRAY	front	99	163
3	C:/I...	EZJ0074	EZJ0074	COMMON	BRA	60	657027	((3...	30	(255,255,255)	(0,0,0)	(255,255,255)	38	Mitsubishi Outl...	Car	BLUE	front_side	99	89

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:

#,Path;Plate Text;Plate Text (ASCII);Plate
category;Country/State;Confidence;TypeID;Frame;Character Size;Text Background;Text
Color;Dedicated Area Color;ANPR Time(ms);Make&Model;Category;Color;ViewPoint;MMR
confidence;MMR Time(ms);EADR type;EADR frame

ADV DEMO

1. INTRODUCTION

The ANPR Demo for Videos (ADI) is an application that was developed by Adaptive Recognition Hungary Inc. 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:

- .mjppg
- .mjjpeg
- .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 and CenOS6 packages on Linux.

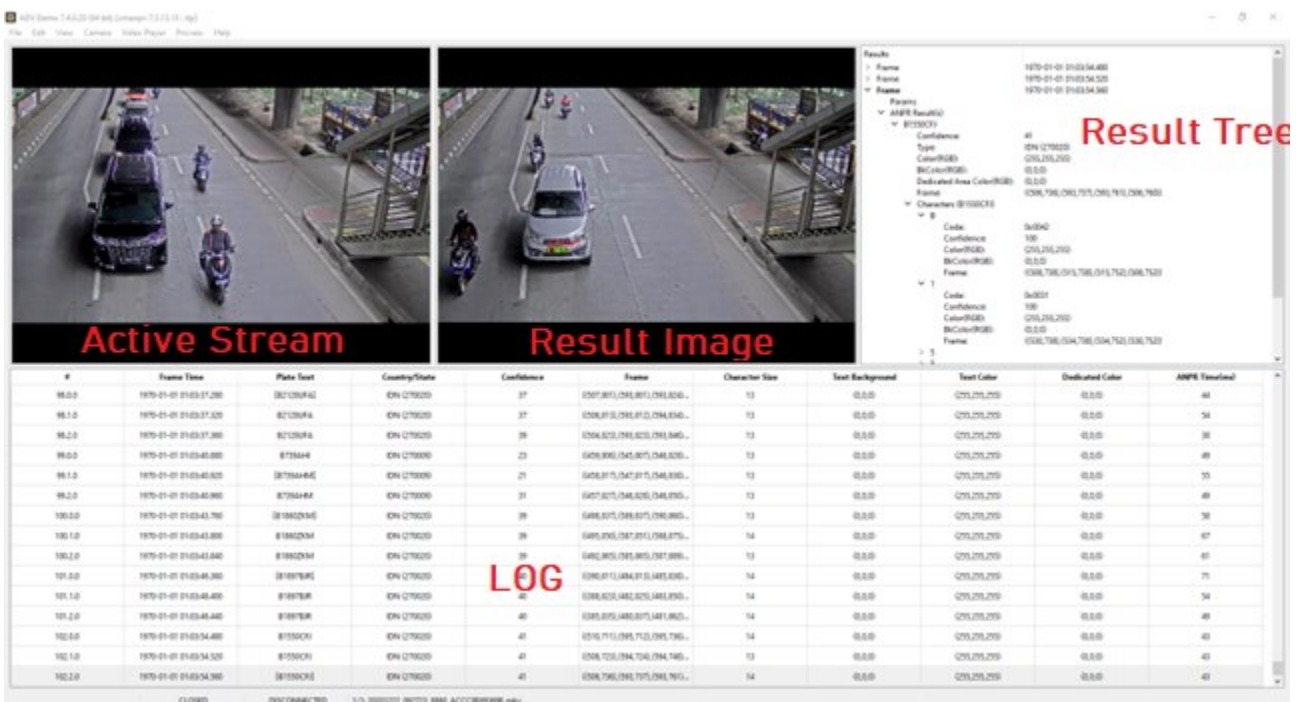
Note

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

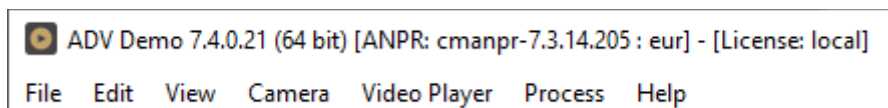
After starting ADV, you will be presented with the main screen of the application. This window is split into four separate sections; the Active Stream, the Result Image, the Result Tree and the Log sections. All four sections provide information in connection with the scanned image. The ADV menu bar 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



Quick overview of the main screen

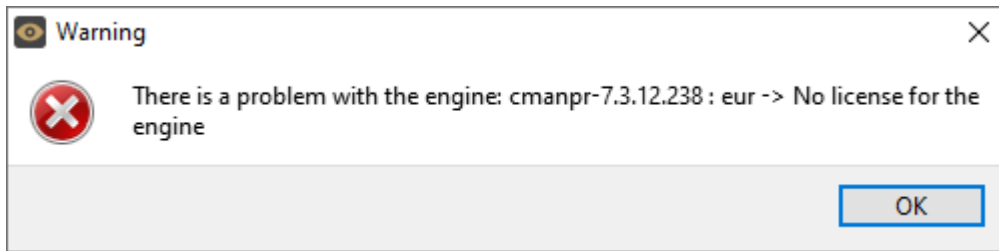
In the title you can see the following information:



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)

Note

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



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 Tree:** 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 tree where it recognized a license plate.

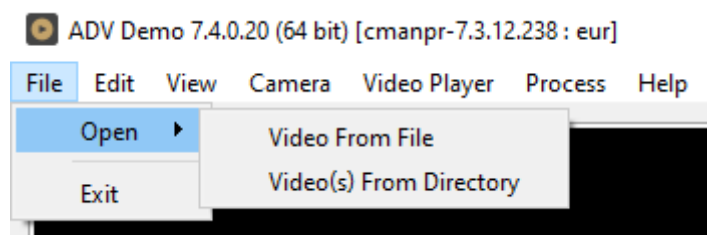
The lower part of the screen contains only one section.

- **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

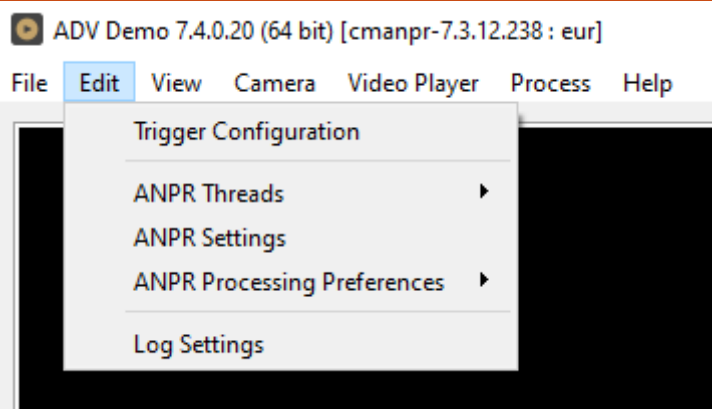
! Important

Trigger configuration largely depends on the location of the cameras that monitor front or rear license plates and the system configuration. This 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.

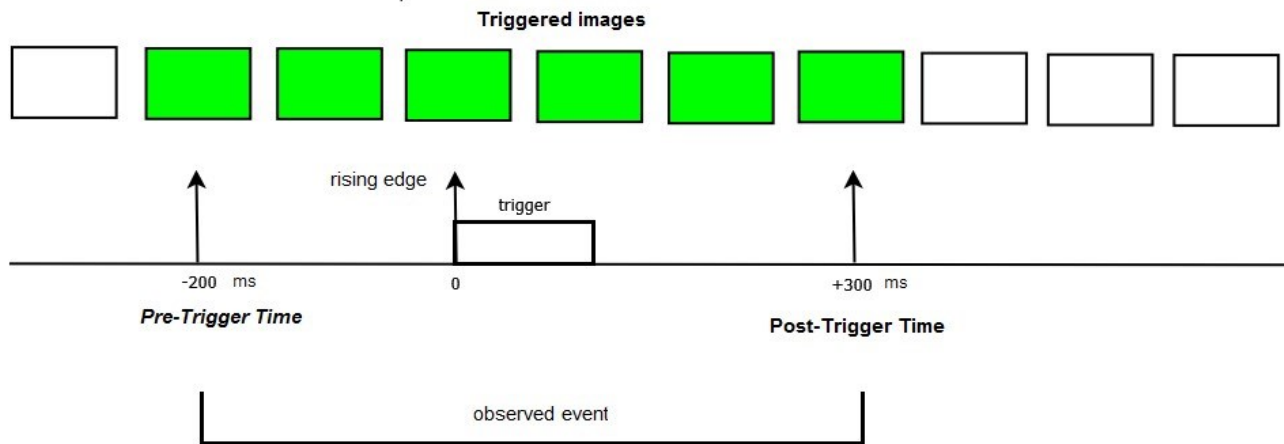
Trigger configuration allows the user to determine a time interval when the system anticipates only those frames from the continuous video frame sequence that are relevant for ANPR. From these relevant images, the engineer must decide what to select as triggered images. Triggered images should be those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



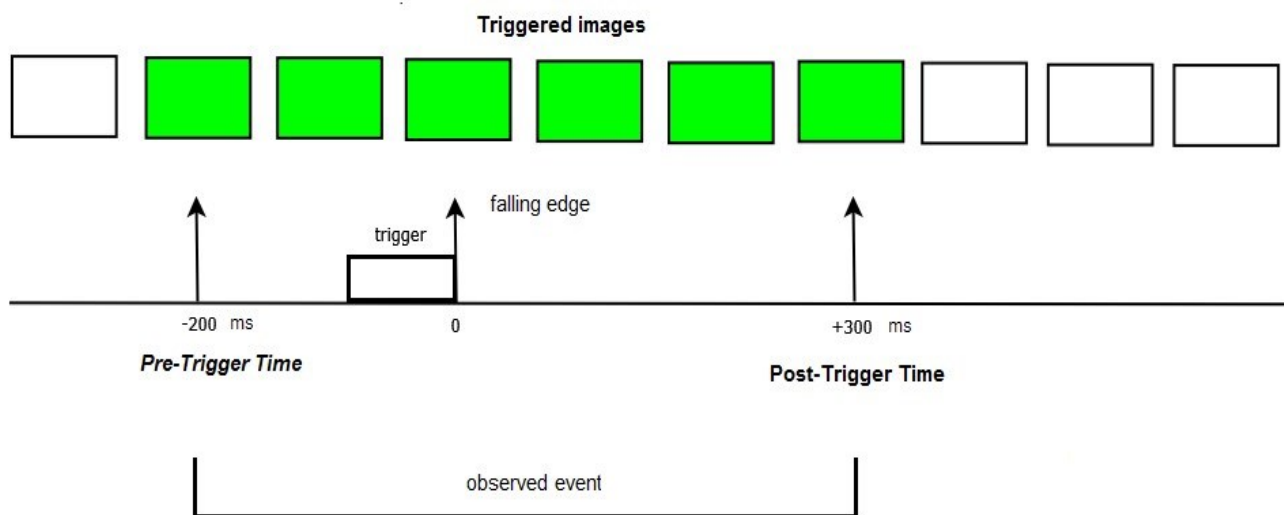
The diagrams below demonstrate how Pre-Trigger Time and Post-Trigger Time are used to define what the triggered images are within an observed event. The application will request the triggered images from the camera for ANPR processing. Since the camera buffers the images internally, even negative offset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

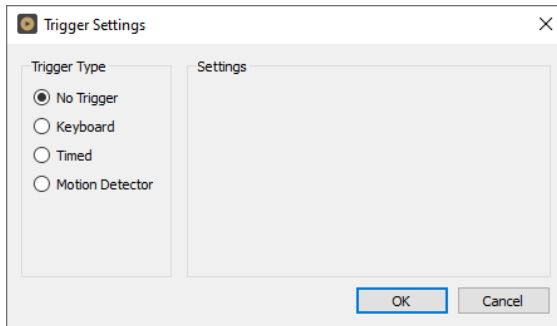
The following examples demonstrate some possible trigger scenarios:

1. Trigger on the rising edge

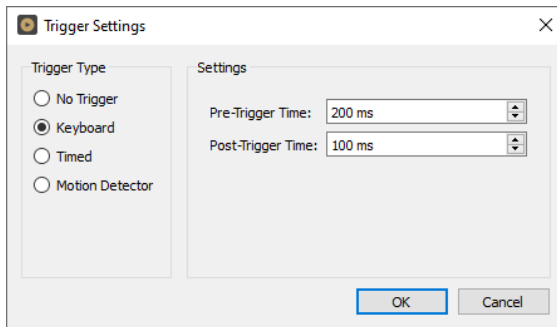


2. Trigger on the falling edge

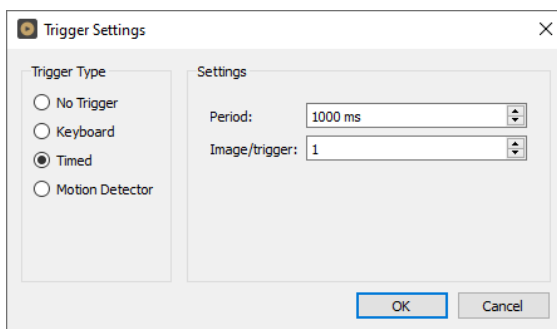




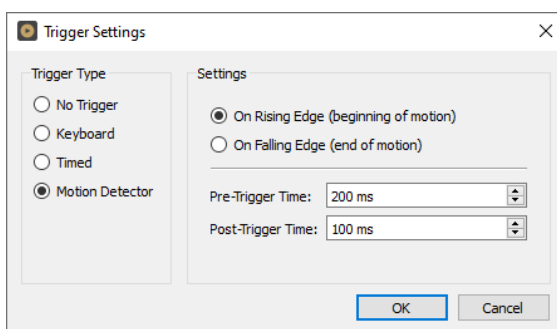
No Trigger: The engine processes all incoming frames that are not dropped from the buffer.



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.

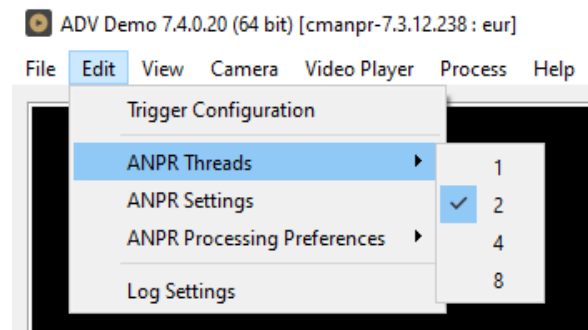
In **No Trigger** and **Timed** modes, the system sends individual images for processing, while in **Keyboard** and **Motion Detector** modes, it sends image packages based on the pre-set time intervals.

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

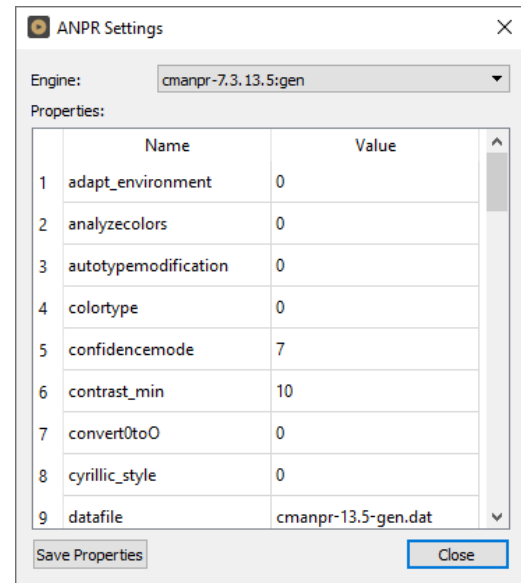
4.3. ANPR SETTINGS

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.

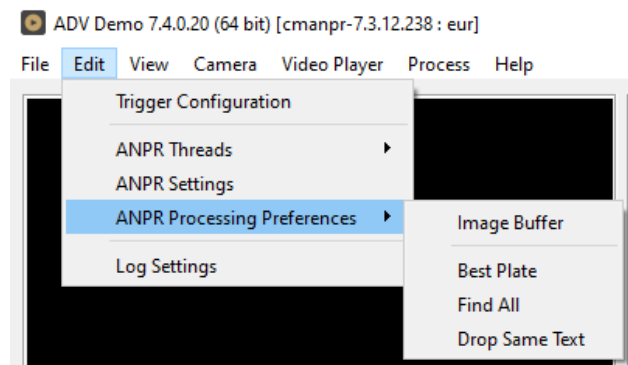
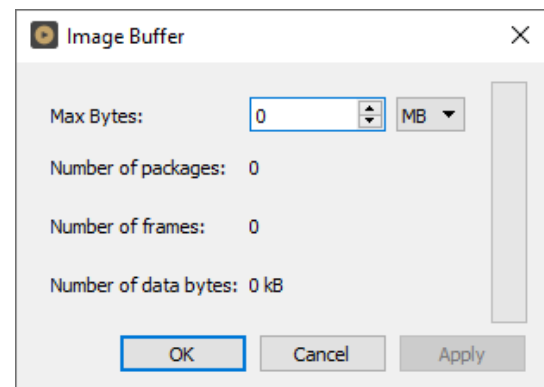


Image Buffer:

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.

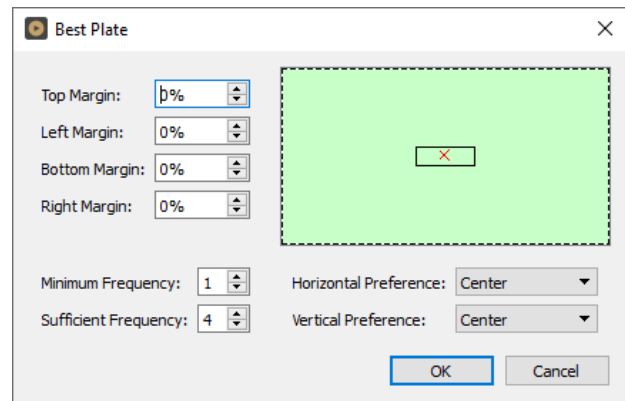


Best Plate:

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:

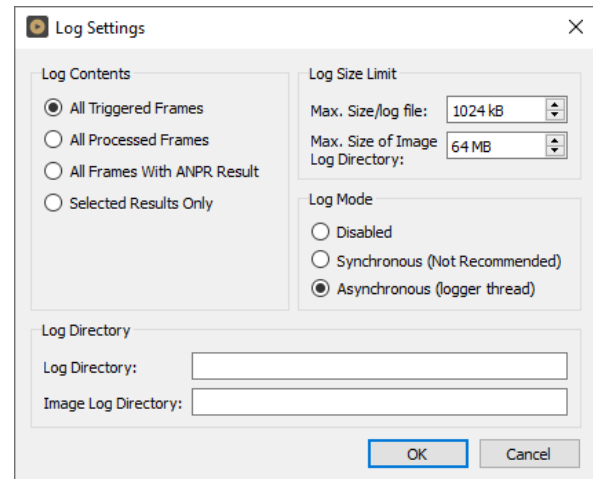
If checked, the system searches for all license plates on the image.

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.



5. VIEW

5.1. FONTS

Set Tree Font...

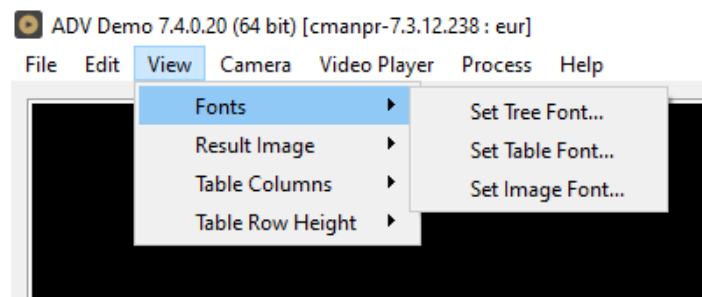
Change the font used to display the Result Tree (the detailed ANPR data found in the upper right corner of the main screen).

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.



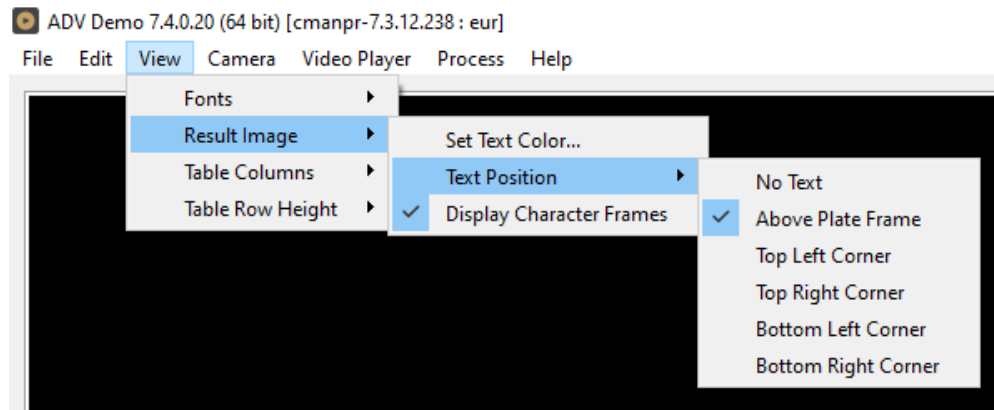
5.2. RESULT IMAGE

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

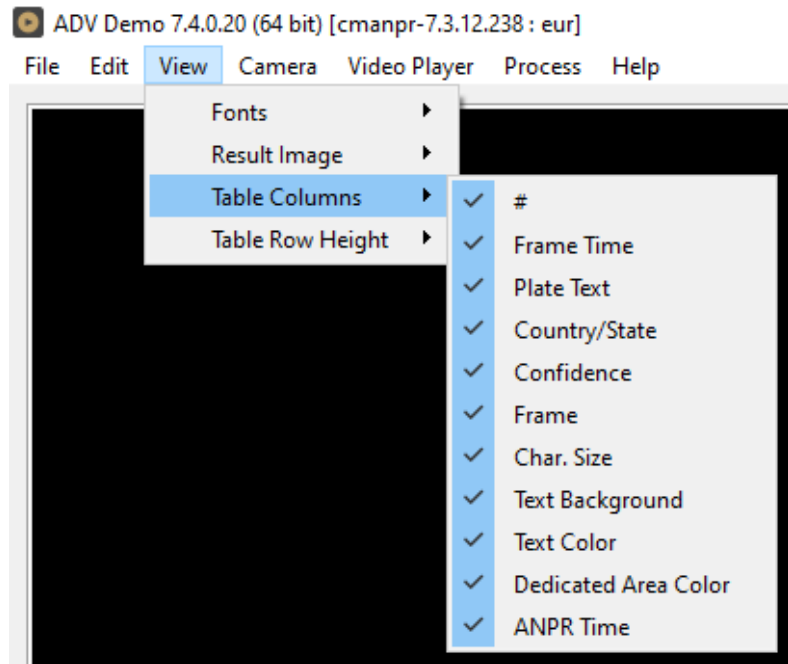
Display Character Frames:

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

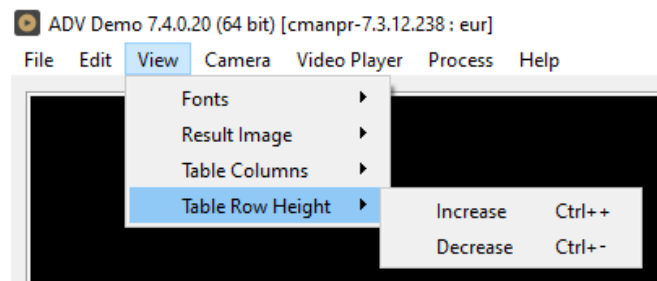


5.4. TABLE ROW HEIGHT

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

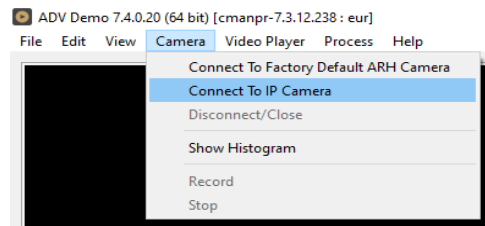
Hotkeys for Increase: **Ctrl + +**

Hotkeys for Decrease: **Ctrl + -**



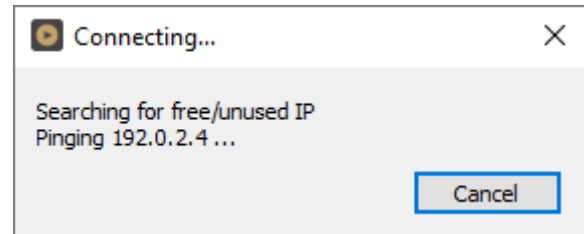
6. CAMERA MENU

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

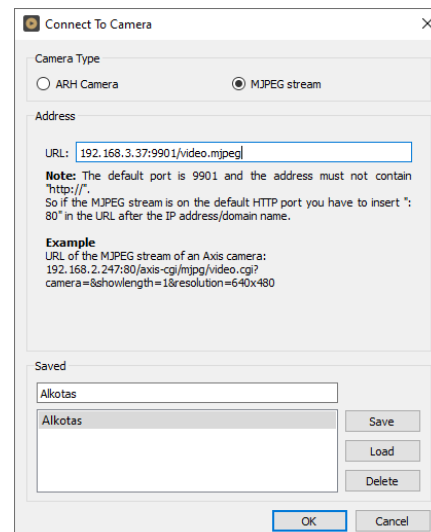
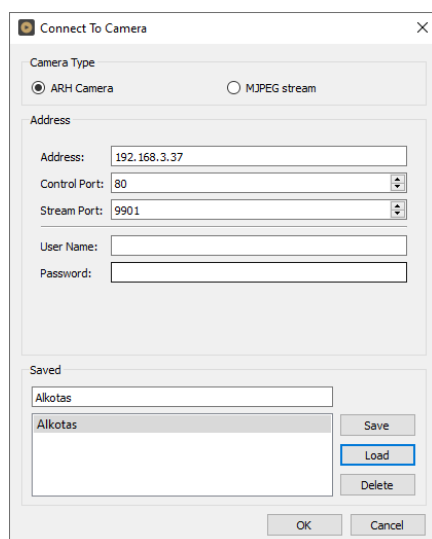
You must set up the network adapter with the same subnet range as the factory default IP subnet of the ARH cameras. Please avoid IP address conflicts by selecting a number different from the camera's default "3" in the last section of the IP address.

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

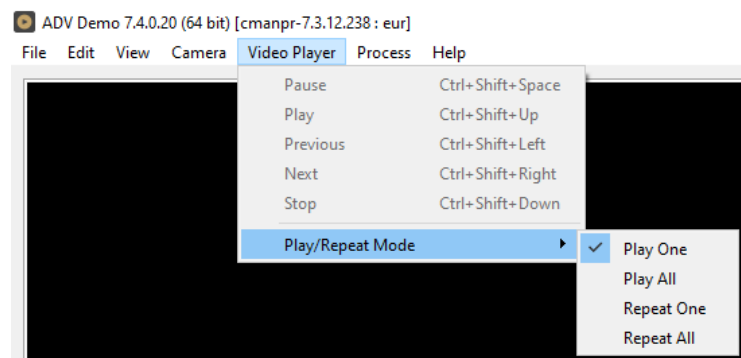
Allows the user to control playback and switch between multiple video streams (if more than one video stream has been loaded).

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:

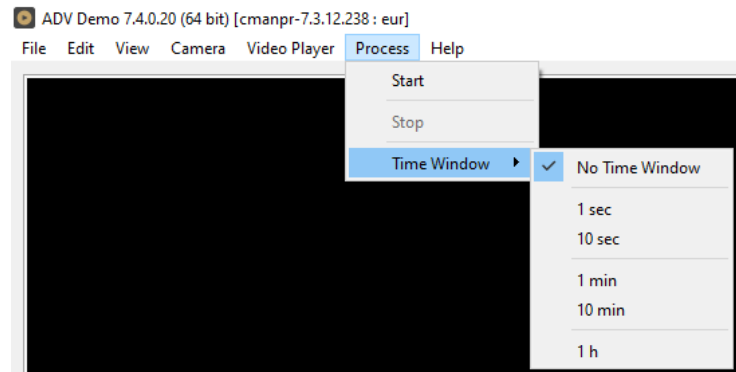
Allows the user to initiate the ANPR process.

Stop:

Allows the user to manually stop the ANPR process.

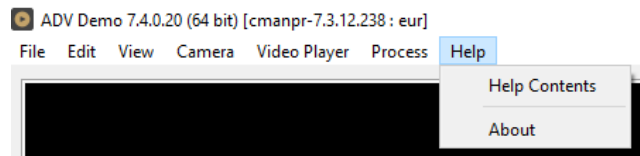
Time Window:

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



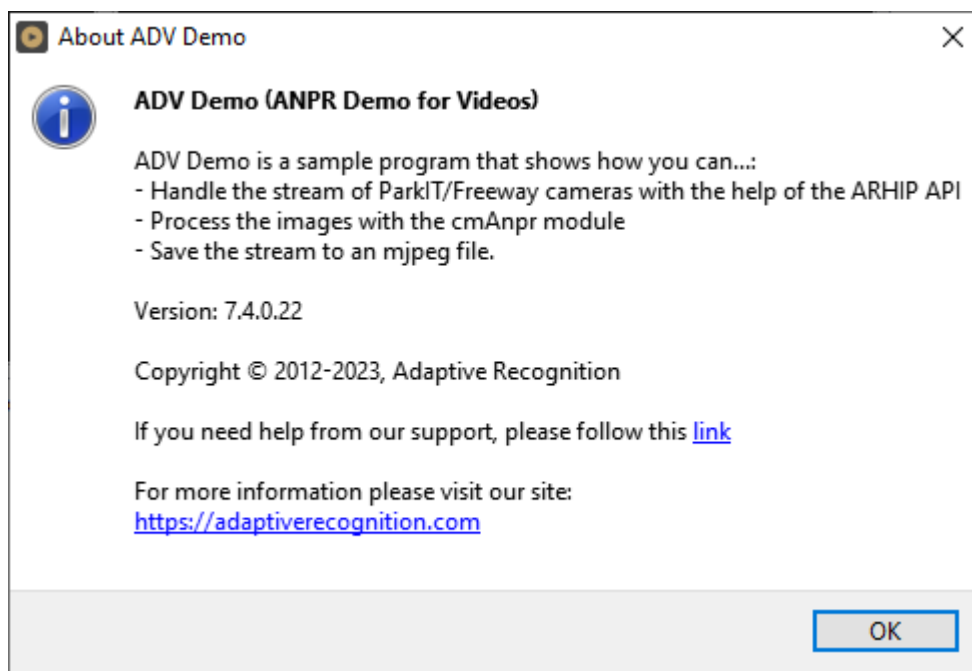
9. HELP MENU

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),(324,357))
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
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 16:26:23.242	[ARH001]		90	((306,331),(423,331),(423,360),(306,360))	25	(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 Hungary 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: **7.3.1.8**

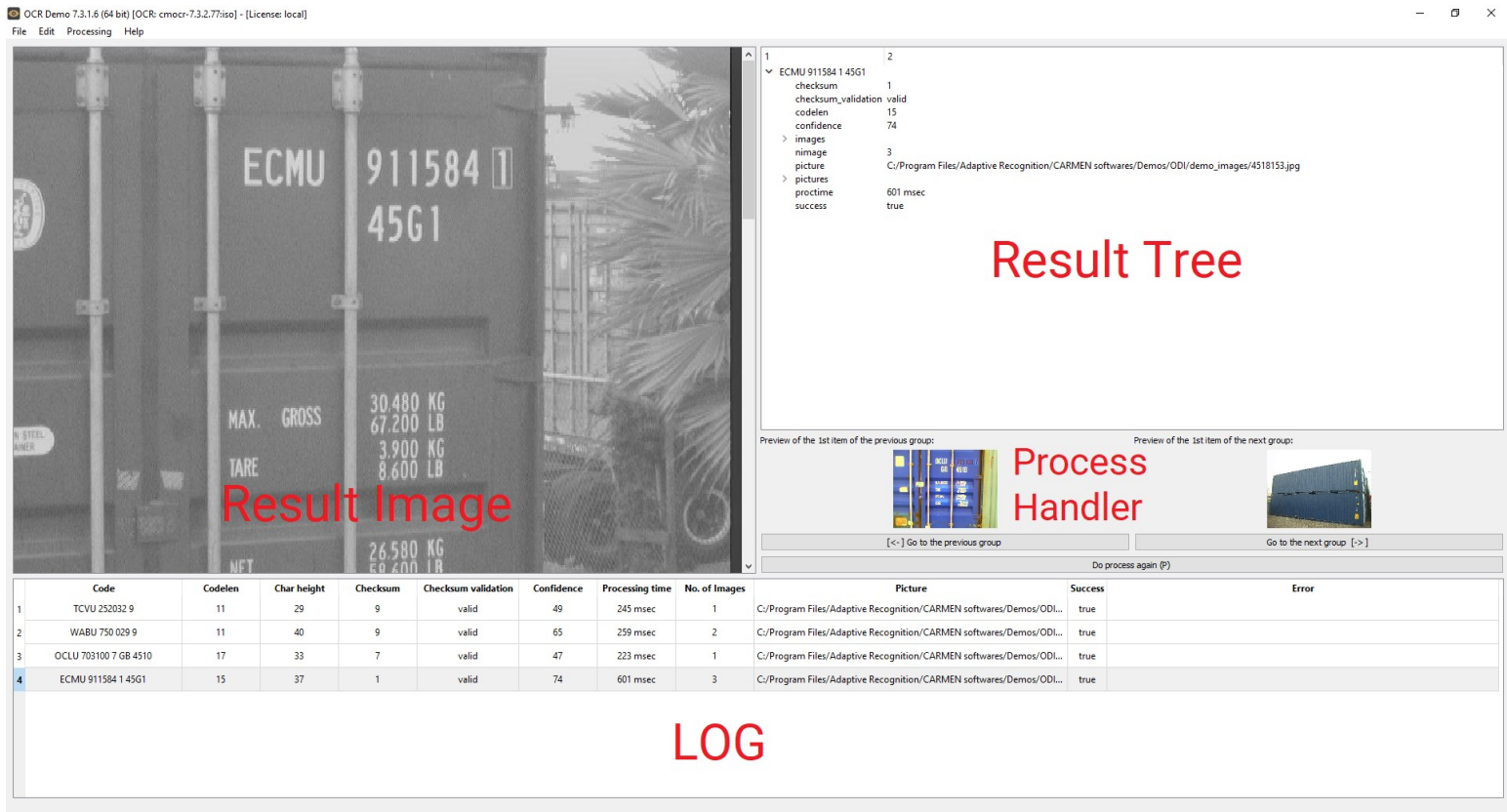
Not all images are adequate for OCR, the input images have to meet a specific set of criteria in order for the engine to be able to recognize them. Please study the following document to learn more about these requirements: [Imaging for Carmen®](#).

! Important

Please note, that this application is not available for ARM and CenOS6 packages 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 three 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:

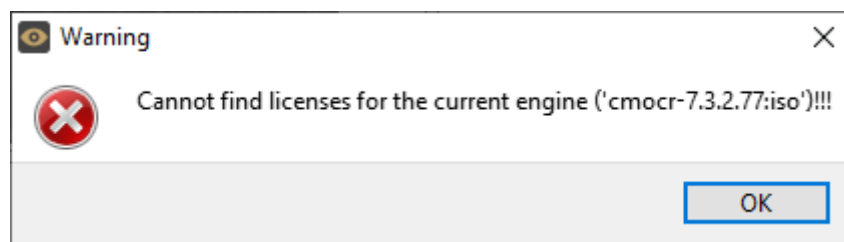
OCR Demo 7.3.1.6 (64 bit) [OCR: cmocr-7.3.2.77:iso] - [License: local]

OCR: the currently used OCR engine type and version

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

Note

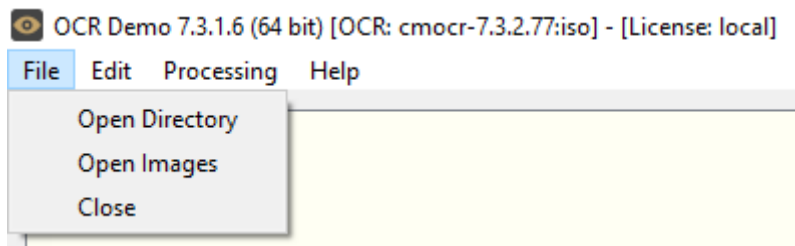
In case of missing hardware key, the following error message will appear:



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

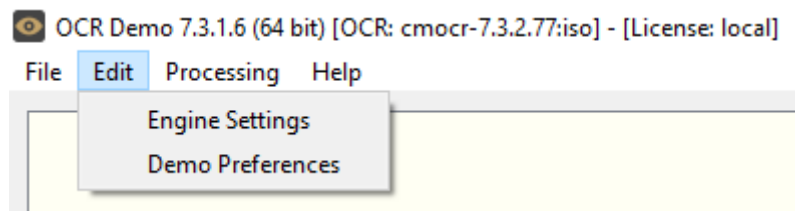
Click to select specific images for OCR processing.

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.

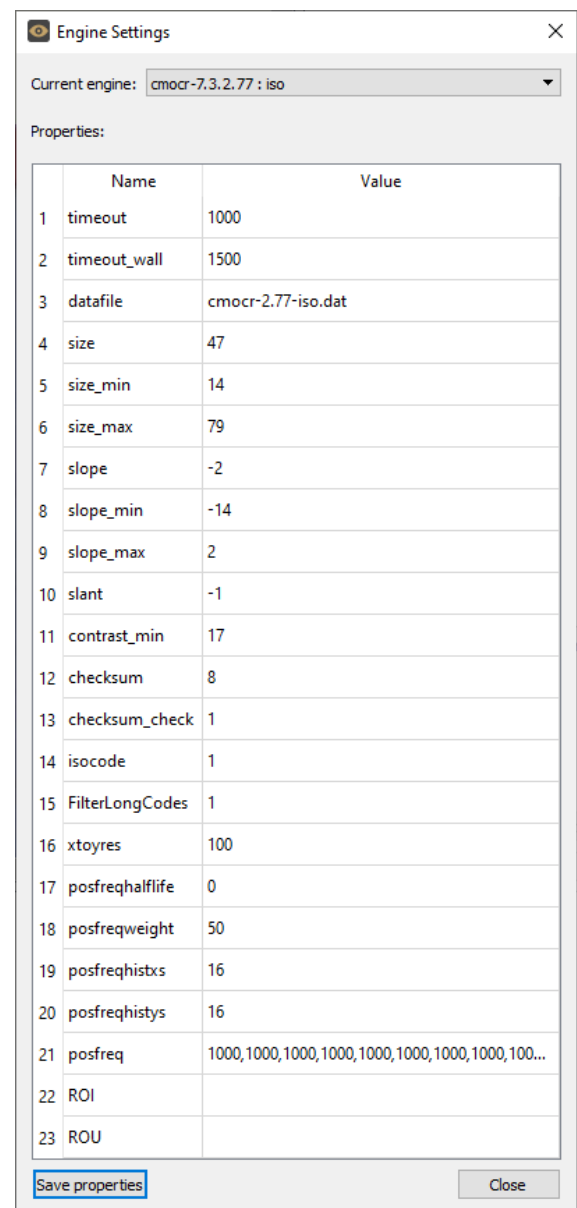
By clicking this menu item, the following pop-up window will appear:

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.

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:

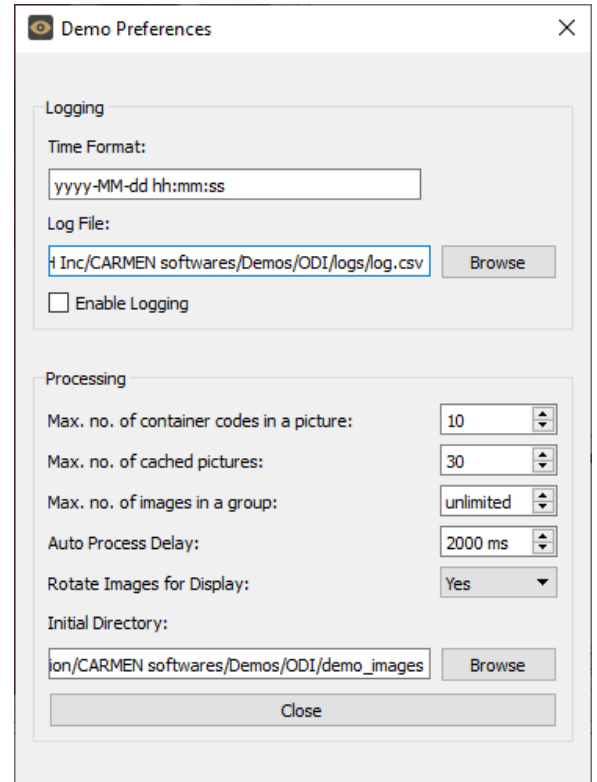
Set how many milliseconds should wait between 2 OCR processes.

Rotate Image for Display:

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

Initial Directory:

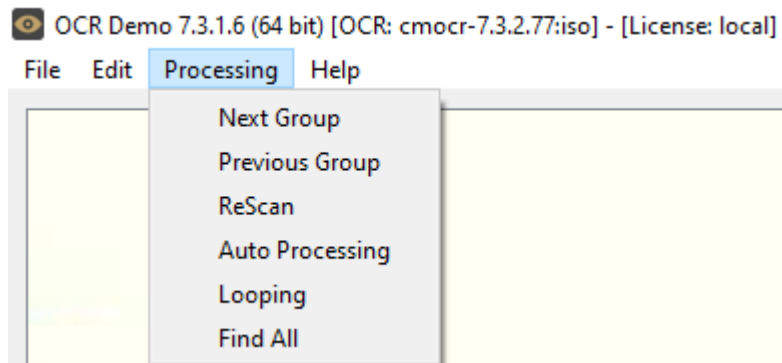
Browse from which directory ODI should search for an image for processing on start-up.



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:

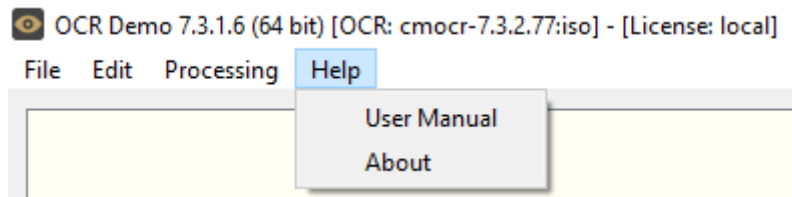
If checked, the program continuously repeats processing of the specified image sequence/folder.

Find All:

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

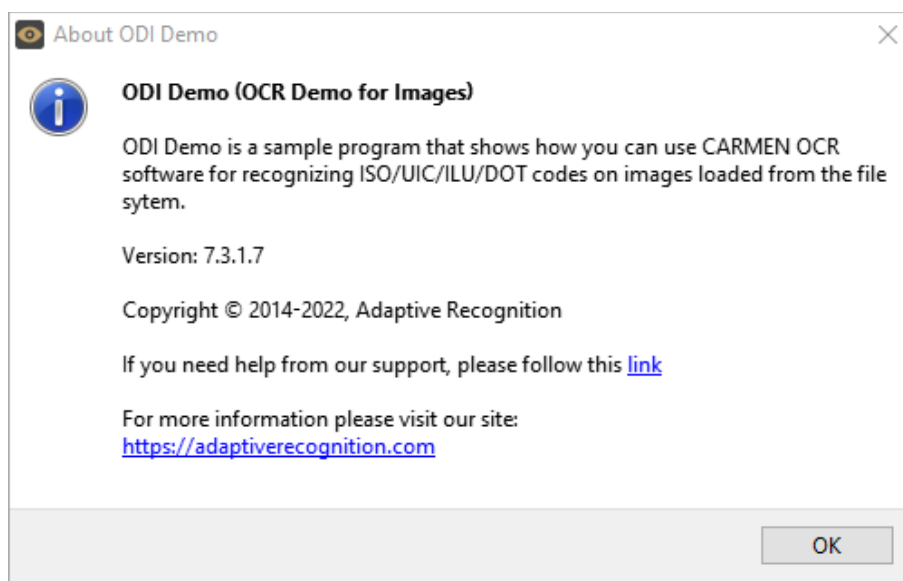
6. HELP MENU

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.



```
▼ ECMU 911584 1 45G1
  checksum          1
  checksum_validation valid
  codelen           15
  confidence        74
  > images
  nimage            3
  picture           C:/Program Files/Adaptive Recognition/CARMEN softwares/Demos/ODI/demo_images/4518153.jpg
  > pictures
  proctime          597 msec
  success           true
```

4. Code result:

Provides a detailed summary of the recognized image:

- **checksum:** The result of the checksum calculation from the code characters
- **checksum_validation:** the result of the checksum validation
- **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

```

    images
    > characters
      checksum      0
      checksum_validation valid
      confidence    99
      imgindex     1
      ncharacter    15
      picture        C:/Program Files/Adaptive Recognition/CARMEN softwares/Demos/ODI/demo_images/4518153.jpg
      text          ECMU911584145G1
    > characters
      checksum      0
      checksum_validation valid
      confidence    99
      imgindex     3
      ncharacter    15
      picture        C:/Program Files/Adaptive Recognition/CARMEN softwares/Demos/ODI/demo_images/4518258.jpg
      text          ECMU911584145G1
    > characters
      checksum      0
      checksum_validation valid
      confidence    97
  
```

5. 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
- **checksum:** the result of the checksum calculation
- **checksum_validation:** result of the checksum validation
- **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

```

    characters
      code      69
      code_char E
      confidence 99
      frame     (243,107)-(262,107)-(262,147)-(243,147)
      code      67
      code_char C
      confidence 100
      frame     (265,107)-(284,107)-(284,147)-(265,147)
      code      77
      code_char M
      confidence 98
      frame     (287,107)-(307,107)-(306,147)-(286,147)
  
```

▼ pictures



C:/Program Files/Adaptive Recognition/CARMEN softwares/Demos/ODI/demo_images/4518153.jpg

C:/Program Files/Adaptive Recognition/CARMEN softwares/Demos/ODI/demo_images/4518196.jpg

C:/Program Files/Adaptive Recognition/CARMEN softwares/Demos/ODI/demo_images/4518258.jpg

6. 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
- **Checksum validation:** the result of the checksum validation process.
- **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	Checksum	Checksum validation	Confidence	Processing time	No. of Images	Picture	Success
2	TCVU 252032 9	11	29	9	valid	49	237 msec	1	C:/Program Files/Adaptive Recognition/CARMEN softwares/Demos/ODI...	true
3	WABU 750 029 9	11	40	9	valid	65	240 msec	2	C:/Program Files/Adaptive Recognition/CARMEN softwares/Demos/ODI...	true
4	TCVU 252032 9	11	29	9	valid	49	233 msec	1	C:/Program Files/Adaptive Recognition/CARMEN softwares/Demos/ODI...	true
5	WABU 750 029 9	11	40	9	valid	65	242 msec	2	C:/Program Files/Adaptive Recognition/CARMEN softwares/Demos/ODI...	true
6	TCVU 252032 9	11	29	9	valid	49	232 msec	1	C:/Program Files/Adaptive Recognition/CARMEN softwares/Demos/ODI...	true
7	WABU 750 029 9	11	40	9	valid	65	234 msec	2	C:/Program Files/Adaptive Recognition/CARMEN softwares/Demos/ODI...	true
8	OCLU 703100 7 GB 4510	17	33	7	valid	47	219 msec	1	C:/Program Files/Adaptive Recognition/CARMEN softwares/Demos/ODI...	true
9	ECMU 911584 1 45G1	15	37	1	valid	74	597 msec	3	C:/Program Files/Adaptive Recognition/CARMEN softwares/Demos/ODI...	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;checksum;checksum_validation;confidence;proctime;nimage;picture;success;err
or



CONTACT INFORMATION

Headquarters:

Adaptive Recognition, Hungary Inc.
Alkotás utca 41 HU-
1123 Budapest Hungary
Phone: +36 1 201 9650
Fax: +36 1 201 9651
Web: adaptiverecognition.com

Service Address:

Adaptive Recognition, Hungary Inc.
Ipari Park HRSZ1113/1 HU
2074 Perbál Hungary
Phone: +36 1 2019650
E-mail: rmarequest@adaptiverecognition.com

Adaptive Recognition Technical Support System ([ATSS](#)) is designed to provide you with the fastest and most proficient assistance to get you back to business quickly.

For further technical information about our products, please visit our official website.

Information regarding hardware, software, manuals, and FAQ are easily accessible for customers who previously registered to enter the dedicated ATSS site. Besides offering assistance, the site is also designed to provide maximum protection while managing your business information and the technical solutions utilized.

New User

If this is your first online support request, please create an account by clicking on this [link](#).

Returning User

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

If you need assistance with login or registration, please contact atsshelf@adaptiverecognition.com.

