HD VISION
high dimensional vision systems

LumiScan AI Object Matching

User Guide

HD VISION
high dimensional vision systems

Imprint

HD Vision Systems GmbH
Carl-Friedrich-Gauß-Ring 5
69124 Heidelberg

Tel.: +49 (6221) 67219-00
Fax: +49 (6221) 67219-01
Email: info@hdvisionsystems.com
WWW: https://hdvisionsystems.com

CEO: PD Dr. Christoph Garbe
Headquarters: Heidelberg
Register court: Amtsgericht Mannheim
Register number: HRB 726917

Version_2024_2_19

Table of Contents

1 Introduction

1.1 About This Manual

This document contains important information about the installation and operation of the LumiScan AI Object Matching application. Please read the manual carefully before using this software.

1.2 Formatting Convention

This guide uses special formatting to highlight certain words and phrases:

- Keywords/important information and buttons are highlighted in bold (e.g. **inference/ready**).
- Links and references are highlighted in green (e.g. info@hdvisionsystems.com).
- File and path names are highlighted in a special font (e.g. `hdvisionsystems/lumiscan-ai-object-matching/<network name>/control/inference/ready`).

1.3 Disclaimer

NOTE: By accessing or using these commercial software products, you expressly agree to the following terms and conditions.

Any attempt to use a debugger to examine, analyze, or tamper with the software provided by HD Vision Systems is strictly prohibited and may have immediate and irreversible consequences.

If the software detects the presence of a debugger, security protocols will be activated to protect the intellectual property, functionality and stability of the software. This may result in, among other things, immediate suspension of the associated Software license, loss of data, and, in extreme cases, forced termination of all instances of the Software operating under the same license, as well as legal consequences.

By using the Software, you acknowledge that you have read, understood and accepted the terms of this Disclaimer (see also General Disclaimer).

2 LumiScan AI Object Matching

2.1 Description

The LumiScan AI Object Matching is an application that uses neural networks to find objects in images. It returns the position and orientation of an object in pixel coordinates. This software runs on ctrlX CORE and its interface is ctrlX Data Layer. It is installed via the App Store and configured via a settings file.

2.2 Requirements

- ctrlX CORE
- Neural Network Models provided by LumiScan Training Framework software

2.3 General Concept of Handshake

The communication process in the LumiScan AI Object Matching application is based on the general concept of handshake. The naming convention for handshake parameters usually consists of the **functionality** and the names **-ready** or **-request**.

Table 1: General Concept of Handshake: Address

Address	Description
<parent address>/<functionality name>/ready	Read-Only, controlled by our app to indicate whether the relevant functionality is allowed to be executed.
<parent address>/<functionality name>/request	Writable. It is used by user to trigger our application to execute relevant functionality.

2.3.1 Functionality Execution

NOTE: To avoid repeating the entire address, in this section the **invariant part** containing the parent address and the **variable functionality name** (inference, reset) have been replaced by **XXX**.

Thus, the address: **hdvisionsystems/lumiscan-ai-object-matching/<network name>/control/inference/ready** will be written as **XXX/ready**.

Accordingly, the address: **hdvisionsystems/lumiscan-ai-object-matching/<network name>/control/inference/request** will be written as **XXX/request**.

In order to execute a functionality, you need to run the following steps (See also image *Functionality Execution Steps*):

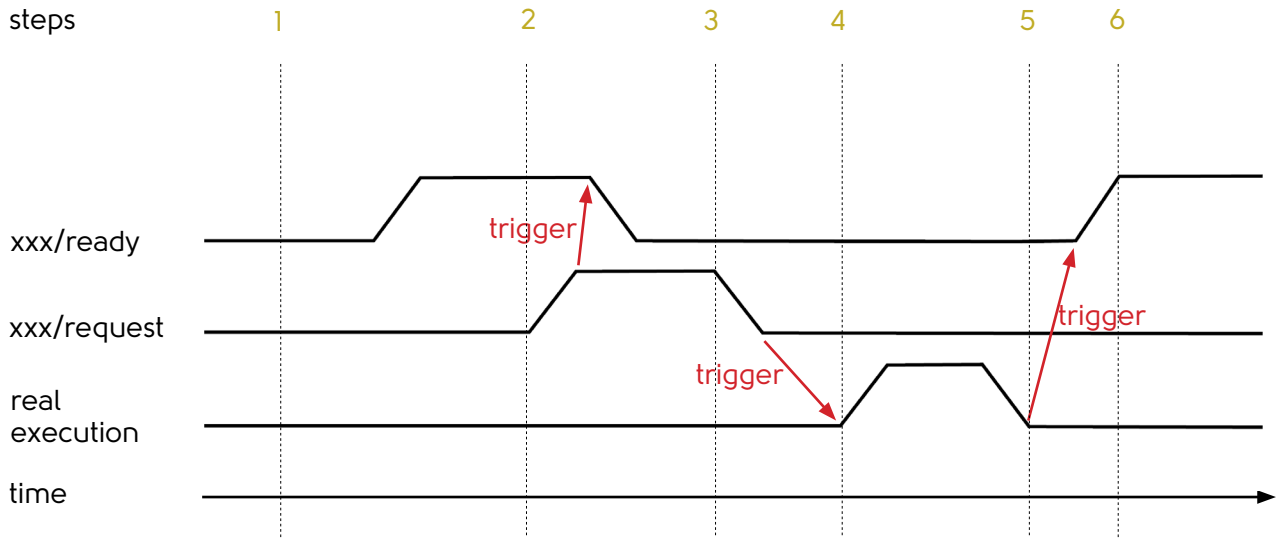
1. Wait until XXX/ready is set to **True**.
2. Set XXX/request to **True**.

NOTE: Make sure there is only one process is accessing XXX/request at a time. The result of execution is **undefined** when:

a. ProcessA set XXX/request to True.
b. ProcessB set XXX/request to False without considering ProcessA.
c. Our application might therefore miss XXX/request from ProcessA.

3. XXX/ready changes to **False**.
4. Set XXX/request to **False**.
5. Start executing relevant functionality.
6. Execution finished.
7. XXX/ready is automatically set to **True** again.*

* Only if the application is configured to use an input image.
If the application is configured to use an external image and there is no image, **XXX/ready doesn't change automatically** and remains set to **False**. In this case check code number. (see also Status Code Values).



Functionality Execution Steps

2.4 Initializing the Inference

2.4.1 Addresses

Table 2: Functionality Addresses

Functionality	Permission	Address
Inference	Read-Only	hdvisionsystems/lumiscan-ai-object-matching/<network name>/control/inference/ready
	Writable	hdvisionsystems/lumiscan-ai-object-matching/<network name>/control/inference/request
Reset	Read-Only	hdvisionsystems/lumiscan-ai-object-matching/<network name>/control/reset/ready
	Writable	hdvisionsystems/lumiscan-ai-object-matching/<network name>/control/reset/request
Status	Read-Only	hdvisionsystems/lumiscan-ai-object-matching/<network name>/control/status/code

2.4.2 Steps To Run Inference with External Image

NOTE: To avoid repeating the entire address, in this section the **invariant part** containing the parent address has been replaced by **XXX**.

Thus, the address: **hdvisionsystems/lumiscan-ai-object-matching/<network name>/control/inference/ready** will be written as **XXX/inference/ready**.

Accordingly, the address: **hdvisionsystems/lumiscan-ai-object-matching/<network name>/control/inference/request** will be written as **XXX/inference/request**.

To start running inference:

1. Requirement: XXX/status/code is 0 (ready*), indicating that the neural network model is set up correctly.
2. Check XXX/inference/ready = True

NOTE: The application will automatically load configurations and setup neural networks. As long las status/code is **0** and XXX/inference/ready is **true**, you can run inference

If XXX/inference/ready is false, either last inference is still running or there's no new external image or application encounter some problem. Please check XXX/status/code and log for more information.

3. Set XXX/inference/request := True
4. Check XXX/inference/ready = False

NOTE: If the application hasn't seen that the variable has changed:

i. Make sure that only one process is accessing XXX/inference/request at a time.
ii. Contact hdvisionsystems.

* The meaning of all status code values is explained in the Status Code Values table at the end of this chapter.

5. Set XXX/inference/request := False
6. Check XXX/status/code = 0.
   You can now check result at `hdvisionsystems/lumiscan-ai-object-matching/<network name>/output/result`

2.4.3 Steps to Run Inference with Input Image

NOTE: To avoid repeating the entire address, in this section the **invariant part** containing the parent address has been replaced by **XXX**.

Thus, the address: **hdvisionsystems/lumiscan-ai-object-matching/<network name>/control/inference/ready** will be written as **XXX/inference/ready**.

Accordingly, the address: **hdvisionsystems/lumiscan-ai-object-matching/<network name>/control/inference/request** will be written as **XXX/inference/request**.

To set input image:

1. Find the following address in the datalayer*: `hdvisionsystems/lumiscan-ai-object-matching/<network name>/input/image`
2. Copy raw image into the following address: `hdvisionsystems/lumiscan-ai-object-matching/<network name>/input/image`
3. Set corresponding image width and height and PixelFormat under the path `hdvisionsystems/lumiscan-ai-object-matching/<network name>/input/image`

NOTE: For the RGB image, data layout is height, width, channel. In other words, the RGB value of a pixel is displayed next to each other in the raw data.

*If the application is configured to use an input image, the specified address should be there.

To start running inference:

1. Requirement: XXX/status/code is 0 (ready*), indicating that the neural network model is set up correctly.
2. Check XXX/inference/ready = True

NOTE: The application will automatically load configurations and setup neural networks. As long las status/code is **0** and XXX/inference/ready is **true**, you can run inference.

3. Set XXX/inference/request := True
4. Check XXX/inference/ready = False

NOTE: If the application hasn't seen that the variable has changed:

i. Make sure that only one process is accessing XXX/inference/request at a time.
ii. Contact hdvisionsystems.

5. Set XXX/inference/request := False
6. Check XXX/status/code = 0.
   You can now check result at `hdvisionsystems/lumiscan-ai-object-matching/<network name>/output/result`

* The meaning of all status code values is explained in the Status Code Values table at the end of this chapter.

2.4.4 Status Code Values

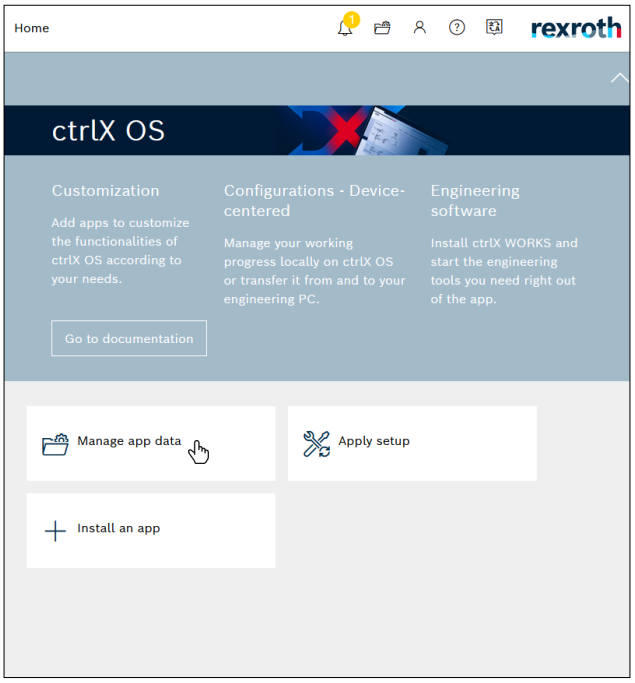
Table 3: Status Code Values Description

Status Code Number	String	Meaning
0	"ready"	The last functionality is successfully executed, the application is ready to execute the next functionality.
1	"settingUp"	The module or function is still setting up.
2	"running"	The function is currently running.
3	"waitingForClient"	Waiting for the client (usually to acknowledge the start of the function).
10001	„noNewExternalImage“	Network is configured to use external image and there’s no new image detected, according to packageIdentifier.
-1	"undefinedError"	Something went wrong and we did not create a special error code for it.
-2	"invalidSetting"	The supplied settings file could not be loaded.
-3	"invalidInput"	One of the input fields was not valid.
-4	"missinglicense"	Make sure you have the correct license installed.

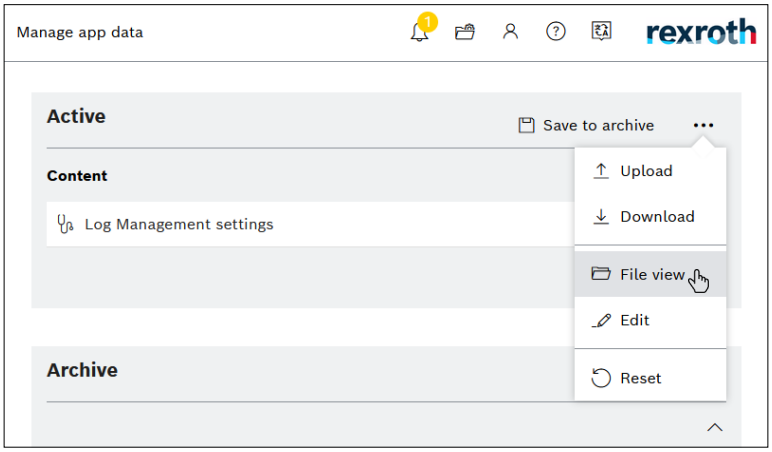
2.5 LumiScan AI Object Matching Folder Structure

2.5.1 Edit with Built-In Website

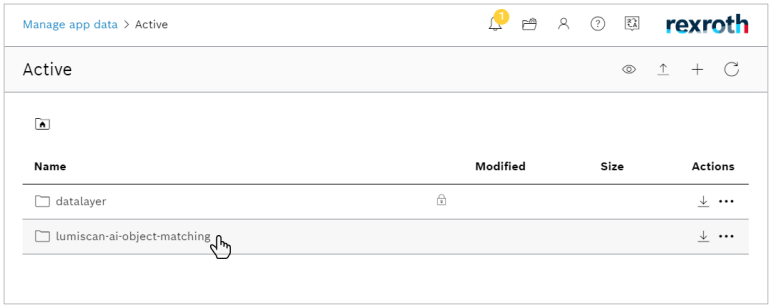
1. Log in to the ctrlX OS website.
2. Go to **Home.**
3. Select **Manage app data.**



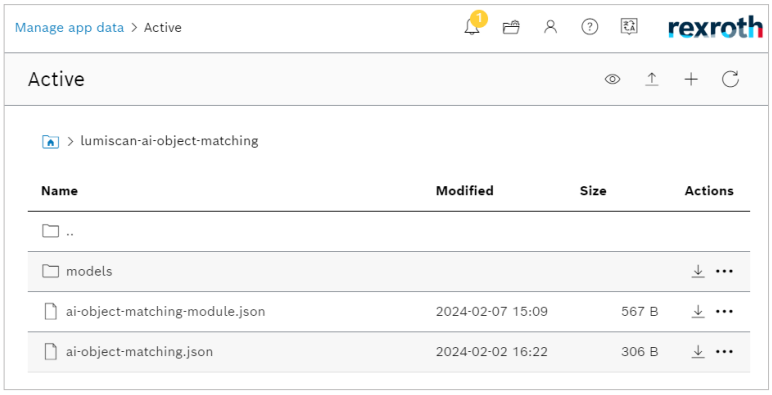
3. Click the three dots and select **File View.**



4. Go to the relevant application folder.

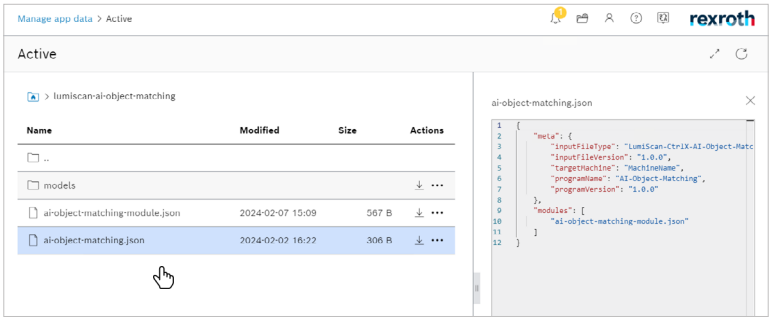


5. There are three files by default (You can add new files later).



- **ai-object-matching.json** contains the **application configuration**. It is used to specify how many neural network modules to load.
  Currently, only one module is supported.
- **<file name x>.json** (here as an example: **ai-object-matching-module.json**) This file contains the **module configuration**. It is used to configure how the network module evaluates image data.
- **models** contains two subfiles with detection parameters and neural network model.

6. Left-clicking on the desired file name will automatically open the contents of the file in the right window.



NOTE: More detailed information about each file can be found in the corresponding sections later in this manual.

2.6 Application Configuration File

This application can run multiple neural network object detection modules. Each of the modules requires a corresponding network module configuration (.json).

The general settings of the Lumiscan AI-Object Matching application are located in the **ai-object-matching.json** file. This file is used to specify how many neural network modules to load.

An example of application configuration JSON settings is shown below.

```
{
    "meta": {
        "inputFileType": "LumiScan-CtrlX-AI-Object-Matching",
        "inputFileVersion": "1.0.0",
        "targetMachine": "MachineName",
        "programName": "AI-Object-Matching",
        "programVersion": "1.0.0"
    },
    "modules": [
        "ai-object-matching-module.json"
    ]
}
```

The description of all the parameters mentioned above can be found in the following Table 4.

Table 4: Application Configuration File: JSON Settings Description

Parameter	Description
meta	Place to set meta information: inputFileType and inputFileVersion are required.
inputFileType	In this case it must be: `LumiScan-CtrlX-AI-Object-Matching`
inputFileVersion	Currently, only version 1.0.0 is supported.
targetMachine	Optional. Enter the name of the machine in use here to indicate that the current application configuration is being used for this particular machine.
programName	Optional. This parameter indicates by which application the current file is used. In this case: `AI-Object-Matching`
programVersion	Optional. Currently used program version.
modules	Contains created model configuration files.

NOTE: After changing the parameters, the LumiScan AI Object Matching application should be restarted.

2.7 Module Configuration File

The module configuration settings are located in the **<file name x>.json** file (in this example, **ai-object-matching.module.json**), and are used to configure how the network module evaluates image data.

An example of module configuration JSON settings is shown below.

```
{
    "meta":
    {
        "inputFileType": "LumiScan-CtrlX-AI-Object-Matching-
Module",
        "inputFileVersion": "1.0.0",
        "targetMachine": "Example",
        "programName": "AI-Object-Matching",
        "programVersion": "0.0.1"
    },
    "name": "example",
    "neuralNetworkConfig": "models/model.json",
    "externalImage": {
        "image": "hdvisionsystems/lumiscan-cam-driver/default-
camera/output/image",
        "packageIdentifier": "hdvisionsystems/lumiscan-cam-dri-
ver/default-camera/output/package-identifier",
        "pipeLength": 10
    }
}
```

The description of the parameters mentioned above can be found in the following Table 5.

Table 5: Module Configuration: JSON Settings Description

Parameter	Description
meta	Place to set meta information: inputFileType and inputFileVersion are required.
inputFileType	In this case it must be: `LumiScan-CtrlX-AI-Object-Matching-Module`
inputFileVersion	Currently, only version 1.0.0 is supported.
targetMachine	Optional. Enter the name of the machine in use here to indicate that the current application configuration is being used for this particular machine.
programName	Optional. This parameter indicates by which application the current file is used. In this case: `AI-Object-Matching`
programVersion	Optional. Currently used program version.
name	A unique identifier for the neural network, which is also used as the network name in the datalayer address.
neuralNetworkConfig	The file path to the neural network model parameters. These parameters are generated by the LumiScan Training Framework.
externalImage	If specified, the network module will create an additional path under `hdvisionsystems/lumiscan-ai-object-matching/<network name>/input` to allow users to upload raw images to the datalayer without the need for an additional application.
image	The file path to the external image. Refer to the requirements in Automatic Copying of External Image.
packageIdentifier	The file path to a number that indicates whether the current image is up to date.
pipeLength	• It must be an integer greater than or equal to 1. • It indicates how many images the application will retain during inference. • The application will monitor changes to the packageIdentifier value. If it changes, the application will copy the image internally to prevent scenarios where the external image is updated twice while the inference is still running.

NOTE: After changing the parameters, the LumiScan AI Object Matching application should be restarted.

2.7.1 Automatic Copying of External Image

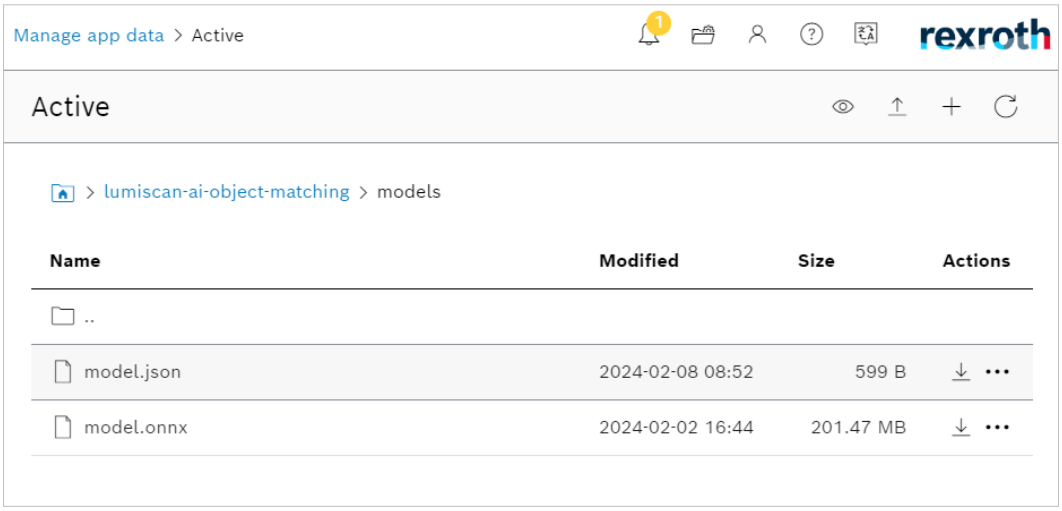
To automatically copy an external image from other applications, these applications must provide an address that contains the following parameters:

Table 6: Parameters of External Image

Address	Type
<address>/data	array of uint8_t
<address>/height	uint32_t
<address>/width	uint32_t
<address>/pixel-type	string, possible values: MONO8, RGB8

<address> is the image field specified in the „externalImage“ in module configuration file.

2.8 Models Folder



The Models folder contains two subfiles:

- **<file name y>.json** (here named **models.json** for example purposes), which contains detection parameters coming from LumiScan Training Framework.
- **<file name y>.onnx** (here referred to as **model.onnx**), which contains the neural network model from LumiScan Training Framework that must be uploaded using WebDAV.*

2.8.1 Model.json File

This file is used to configure which network model to use in the detection parameters.

An example of the JSON settings for this file is shown below.

```
{
    "classList": {
        "0": "nockeIO",
        "1": "nockeVerbogen"
    },
```

* For information how to upload data with WebDAV refer to Upload of Data with WebDAV section.

```
    "thresholdList": {
        "nockeIO": 0.02,
        "nockeVerbogen": 0.02
    },
    "resizedImageShape": {
        "height": 431,
        "width": 571
    },
    "normalizationMean": [
        123.67500305175781,
        116.27999877929688,
        103.52999877929688
    ],
    "normalizationStd": [
        58.39500045776367,
        57.119998931884766,
        57.375
    ],
    "spacialNMSRadius": 3,
    "pathToNetwork": „model.onnx“,
    "provider":"CPU",
    "deviceIndex": 0
}
```

All parameters are generated by the LumiScan Training Framework software.

Of all the parameters mentioned above, only `thresholdList`, `pathToNetwork`, `provider`, `deviceIndex` can be modified and their description is given in Table 7.

Table 7: Model.json File: JSON Settings Description

Parameter	Description
thresholdList	Specify here detection threshold confidence for each detection type (class). • Threshold value should be between 0.0-1.0. • Confidence stands for possibilities of the detected object. • 1.0 stands for 100%-sure the detection object is correct.
pathToNetwork	The path to trained neural network model used for object detection **Note: Path to Network should be relative path to current json file.**
provider	Currently, only `CPU` is possible.
deviceIndex	Currently, it must be 0.

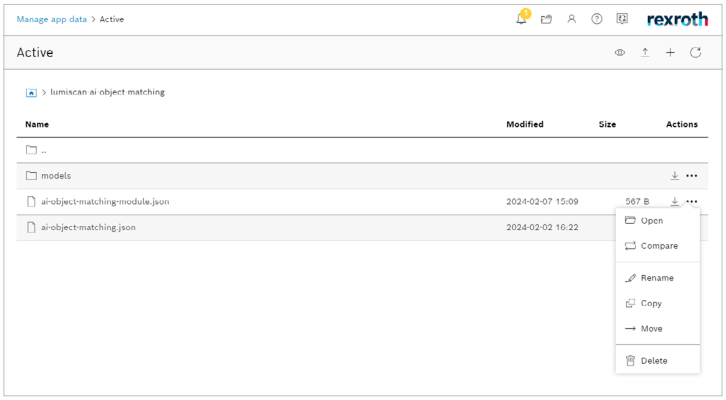
NOTE: After changing the parameters, the LumiScan AI Object Matching application should be restarted.

2.9 Adding New Module

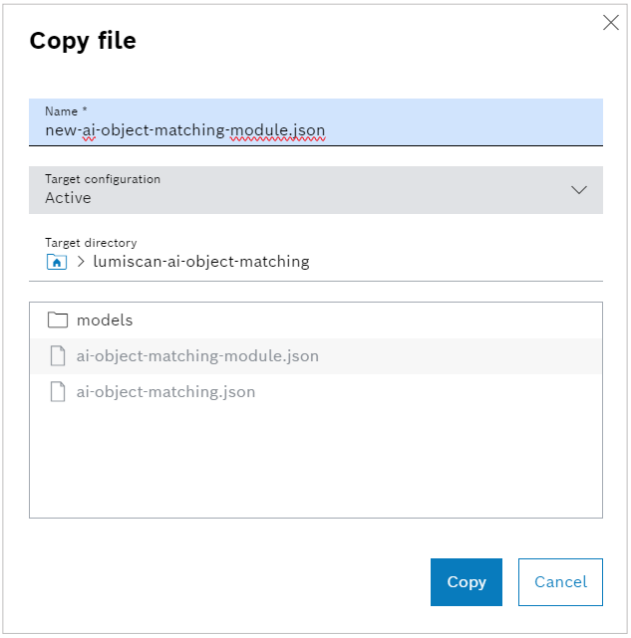
NOTE: Before you create new module you need to download new models folder (see also Models Folder) using WebDAV (Uploading Data with WebDAV).

You can create a new module simply by duplicating an existing one. To do this:

1. Click the triple dots you see to the right of the file name.



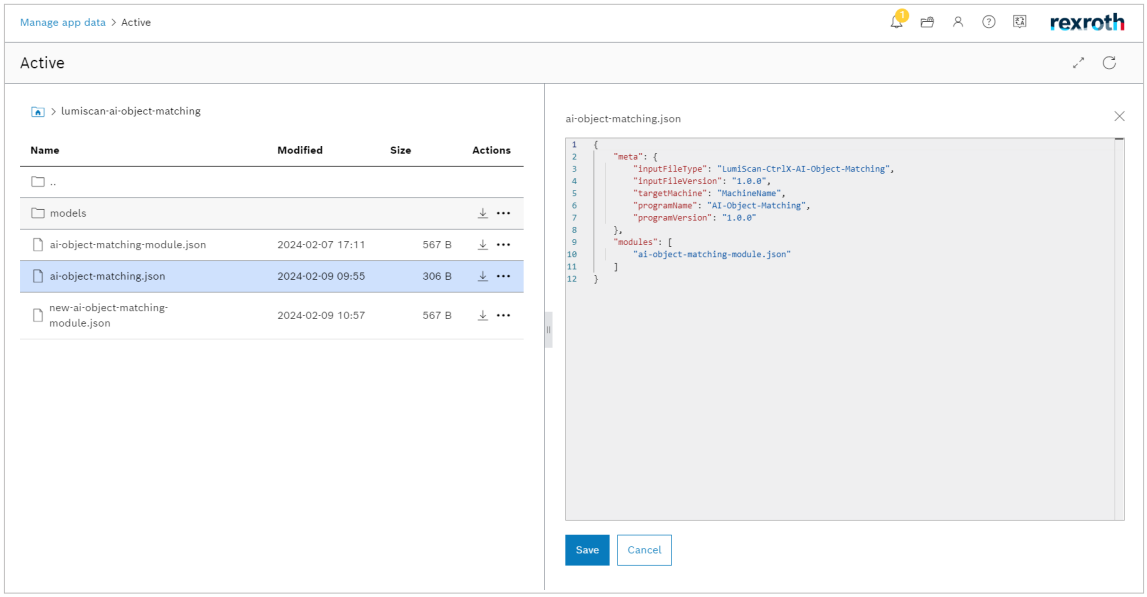
2. Select **COPY**
3. The copy file window appears.



4. Set new file name.

NOTE: You can also rename the new file later by clicking the triple dots to the right of the name and selecting **Rename** from the expanded list.

5. Confirm with clicking on **Copy.**
6. New model configuration settings file is created.
7. Now you can specify the newly created file name in the application settings (ai-object-matching.json):
   a. Left-click on the **ai-object-matching.json** file name.
   b. The JSON settings window will open on the right.

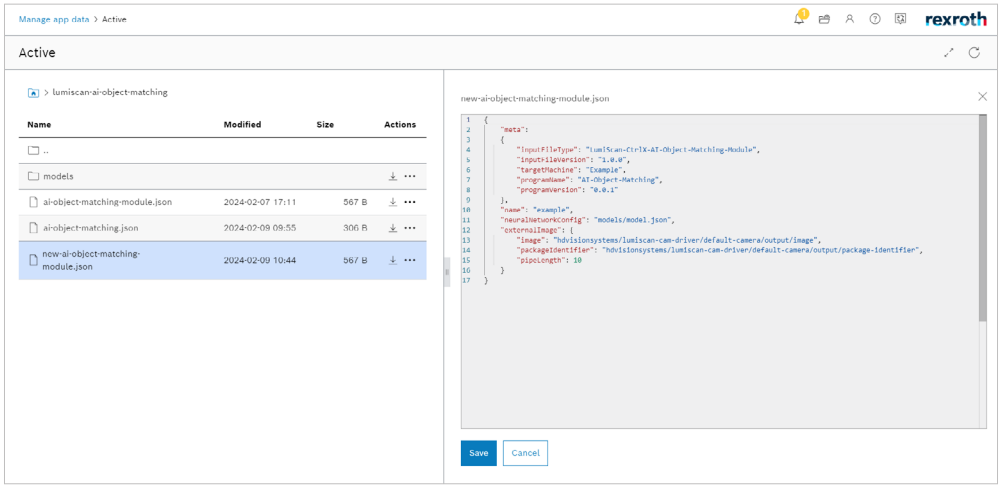


   c. Type the name of the newly created module in the `modules` field.

NOTE: Since we currently support only one module at a time, please remove the old one from the application's JSON settings.

8. Return to the LumiScan AI Object Matching main window, which lists all files.

9. Edit the content of the newly created module (left click on the file name).
10. File settings will appear in the window on the right side of the screen.



11. In the field for the `neuralNetworkConfig` parameter, enter the path to the newly downloaded models folder.

12. Save the changes.

NOTE: After changing the parameters, the LumiScan AI Object Matching application should be restarted.

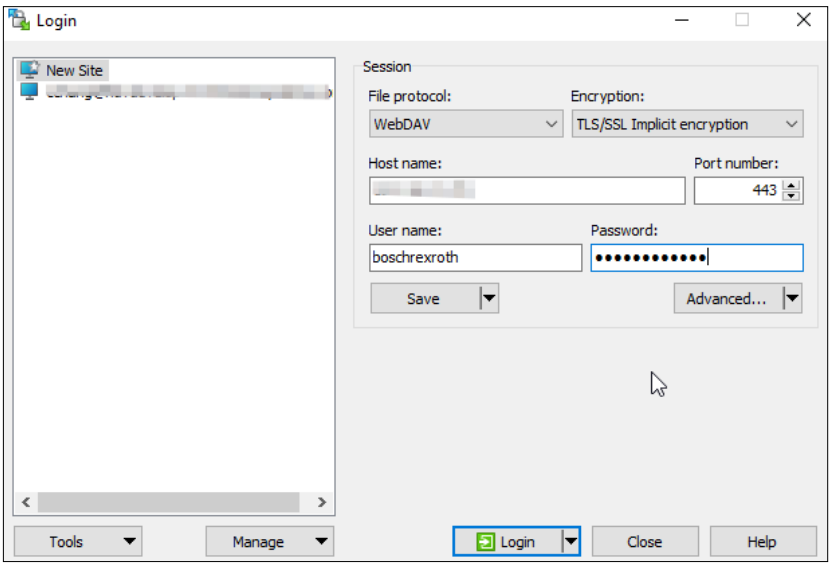
2.10 Uploading Data with WebDAV

NOTE: Uploading a neural network model is required to use the software. Optionally, if the user wants to use his own settings, an application configuration upload is also possible.

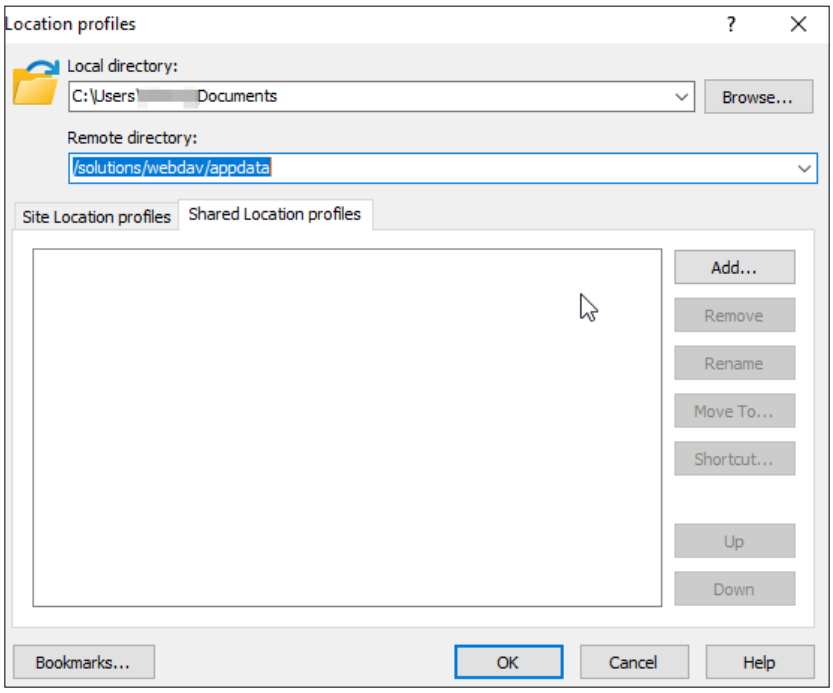
To upload the data files:

1. Go to the WinSCP website.

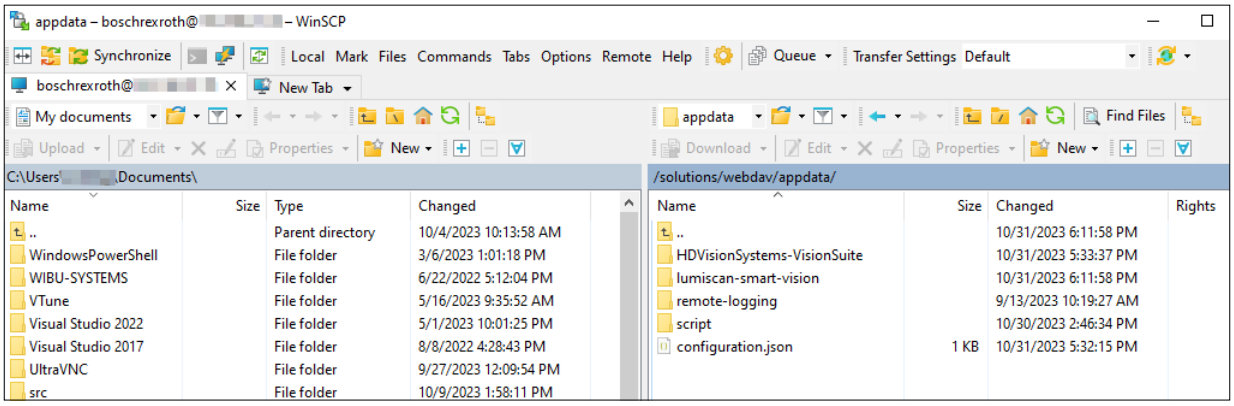
2. Download the latest software version.
3. Install it on your device.
4. Start WinSCP.
5. Login page will open.



6. Select WebDAV as the **File protocol**.
7. Select TLS/SSL for **Encryption**.
8. Enter **user name** and **password**.
9. Navigate to **/solutions/webdav/appdata.**



10. You can see folder structure as below on the right panel.



You can drag and drop data, settings into the corresponding folder with the application name.

3 General Disclaimer

1. The manufacturer is not liable for damage to life, body or health or damage to property resulting from improper use. Please note that operating and/or connection errors are beyond our control. We cannot accept any liability for damage resulting from this.
2. Any damage caused by unauthorized debugging activities to inspect, analyze or manipulate the software provided by HD Vision Systems is the sole responsibility of the user. Any attempt to use a debugger is strictly prohibited and may result in irreversible consequences such as loss of data, interruption of service, and even legal action. The manufacturers, developers and distributors of the software are not liable for any damage or loss resulting from the user's failure to comply with this warning.
3. Claims for damages on the part of the purchaser arising from culpa in contrahendo, breach of secondary contractual obligations and tort are excluded, unless they are based on intent or gross negligence on the part of HD Vision Systems GmbH itself or one of our vicarious agents.
4. Recourse claims in the sense of §12 of the Product Liability Act (Produkthaftungsgesetz) are excluded, unless the party entitled to recourse proves that the defect was caused by HD Vision Systems GmbH and was due to at least gross negligence.
5. If any provision of this Agreement is or becomes invalid or unenforceable, the validity of the remaining provisions shall not be affected. The invalid or unenforceable provision shall be replaced by a valid and enforceable provision whose effects come as close as possible to the economic purpose pursued by the parties with the invalid or unenforceable provision. The foregoing provisions shall apply mutatis mutandis in the event that the contract proves to be incomplete. German courts shall have exclusive jurisdiction over all disputes arising in connection with this disclaimer.
6. For more information about the warranty, please contact the manufacturer of the product.

HD Vision Systems GmbH
Carl-Friedrich-Gauß-Ring 5
69124 Heidelberg

T +49 6221 6721900
F +49 6221 6721901

info@hdvisionsystems.com
www.hdvisionsystems.com