

# Automatically running reports with the Mascot Daemon Export Extender

Richard Jacob

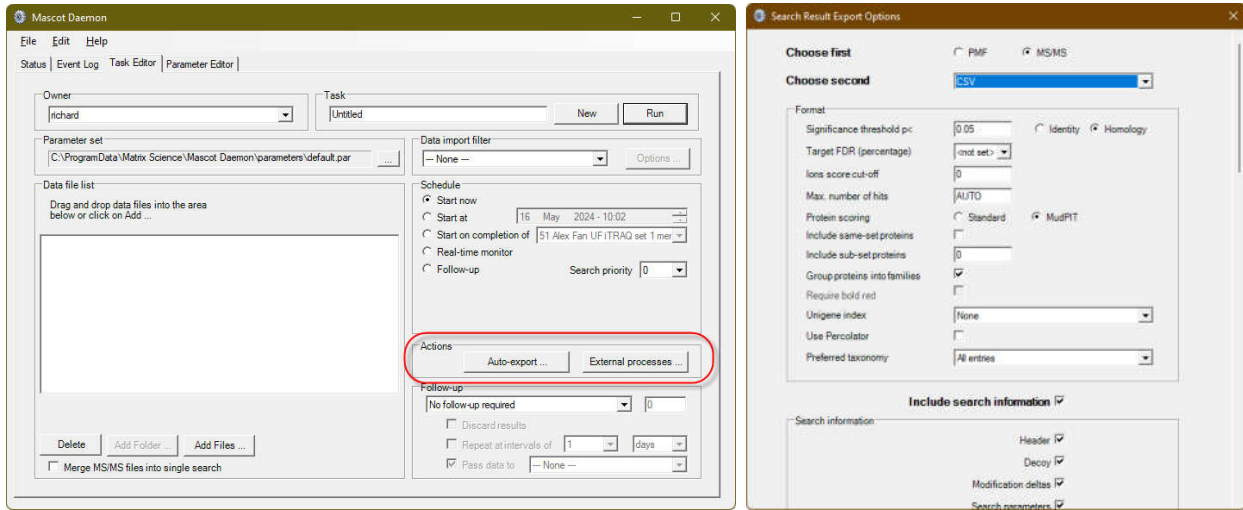
**MASCOT**

: Mascot Daemon Export Extender

© 2024 Matrix Science

 **MATRIX  
SCIENCE**

# Mascot Daemon Actions



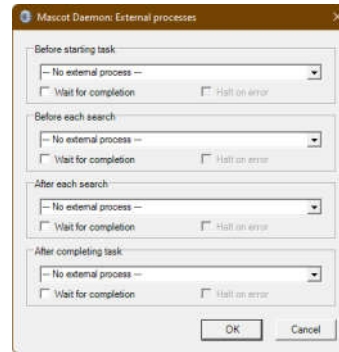
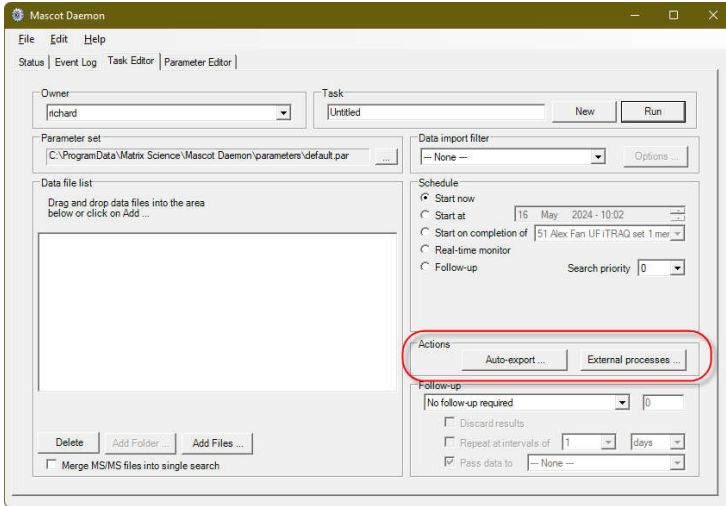
**MASCOT** : Mascot Daemon Export Extender

© 2024 Matrix Science



One of the overlooked features in Mascot Daemon is the actions section. Here you can set up Mascot Daemon to export search results automatically to different formats.

# Mascot Daemon Actions



The External Processors feature gives you the potential to do a lot more and is the key to automation. It can call other programs at different points in a Mascot Daemon task. I recently used it in a blog post to preprocess peak lists before searching when working with TMT complementary ions. A script is called in the Before each search box which opens the peak list and transposes the complementary ions to the reporter ion region. Today I am going to be talking about a script that has been written to make it easier to start processes after each search.

## Automatically exporting Mascot Distiller results

- **Mascot Distiller exports can be initiated on the command line**

- <Full path to Distiller executable> <full path to project file> -batch -quantout <full path to output file> -quantreport <quant python report name> [additional report parameters]

Following on from the last talk about Mascot Distiller reports we are going to look at generating them automatically. At the moment we need to open each file manually and then select the report to generate.

Mascot Distiller does have a command line feature that allows you to generate reports and do some other processing without opening the data. This is a lot quicker than by using the Graphics User Interface.

If you have a lot of files you want to generate reports for you can automate the process. If you know which reports you want to generate then it would be great to be able to do that as part of the processing in Mascot Daemon.

The basic command line to generate a Mascot Distiller report looks like this:

And can be called from the Mascot Daemon External Processes dialog

## Automatically exporting Mascot Distiller results

- **When expanded out**

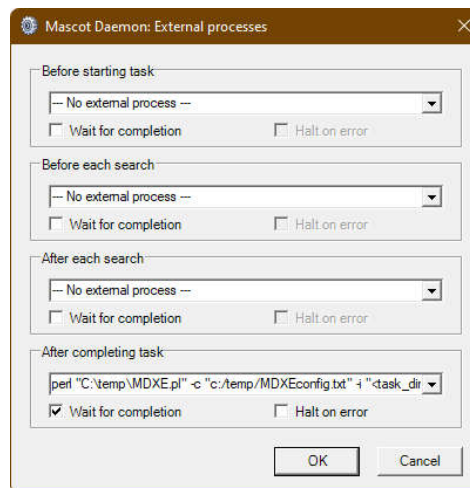
- "C:\Program Files\Matrix Science\Mascot Distiller\MascotDistiller.exe"  
"C:\ProgramData\Matrix Science\Mascot Daemon\MGF\166 Test  
\f4c0612f0fac0df8\_\_mydata.raw.-1.rov" -batch -quantout  
"C:\ProgramData\Matrix Science\Mascot Daemon\MGF\166 Test\  
f4c0612f0fac0df8\_\_mydata.raw.-1.rov.csv" -quantreport table-  
peptides.py -exportFormat CSV

However, when we fill in those options the command line can be quite unwieldy. This leads to the chance of typos and other formatting errors which would mean rerunning the task again in order to correctly generate the report or going back and processing the files manually.

So doable but a bit fussy. Also, what if you want to generate two reports. . .

## Mascot Daemon Export Extender script - MDXE.pl

- Perl script
- Called by Mascot Daemon
- Runs after all the tasks are complete
- Automate Mascot Distiller report generation
- Can run independently
- Can run other tasks



To make the process easier we have released a script that can help.

You need to call the script from the Mascot Daemon External processes tab, after a task completes. It has its own command line but you only pass it the minimum of information.

It comes with a lot of built-in flexibility so it can be used for more than just exporting Mascot Distiller reports.

## MDXE.pl arguments

- **-i input directory**
  - Data directory, Mascot Daemon tag “<task\_directory>”
- **-c config filename**
  - Path to config file
- **-l Enable logging**
  - The optional logging mode records a log to the task directory. The script records the input and output files names along with the expanded command line for each file.

There are three main arguments:

## MDXE configuration file

- Text file
- Uses tokens to define different parts of the command line
- Examples for all the default Mascot Distiller reports
- Can only use one configuration file per use
- But you can have multiple configuration files for different reporting needs

The complexity is hidden in the configuration file. MDXE parses the configuration file to determine what it is going to do.

Internally it uses tokens that are replaced at run time. These tokens cover input and output file names and paths to other programs or executable.

It ships with examples commands for all the default Mascot Distiller reports.

You can only use one configuration file per run but you can create multiple configuration files for different needs.



## Prerequisites for Mascot Daemon Export Extender

- **Need a local copy of Perl installed**
  - Strawberry Perl
  - “perl” needs to be on the path
- **The script can be downloaded from:**
  - <https://www.matrixscience.com/downloads/MDXE.zip>
- **If installed on the same computer as Mascot Server, version 2.5 or earlier, then ActiveState Perl will already be installed**

You do need the Perl scripting language installed on the same computer, typically the Mascot Daemon computer, as where you are going to use MDXE. We currently recommend using Strawberry Perl and as part of that installation Perl should be added to the path.

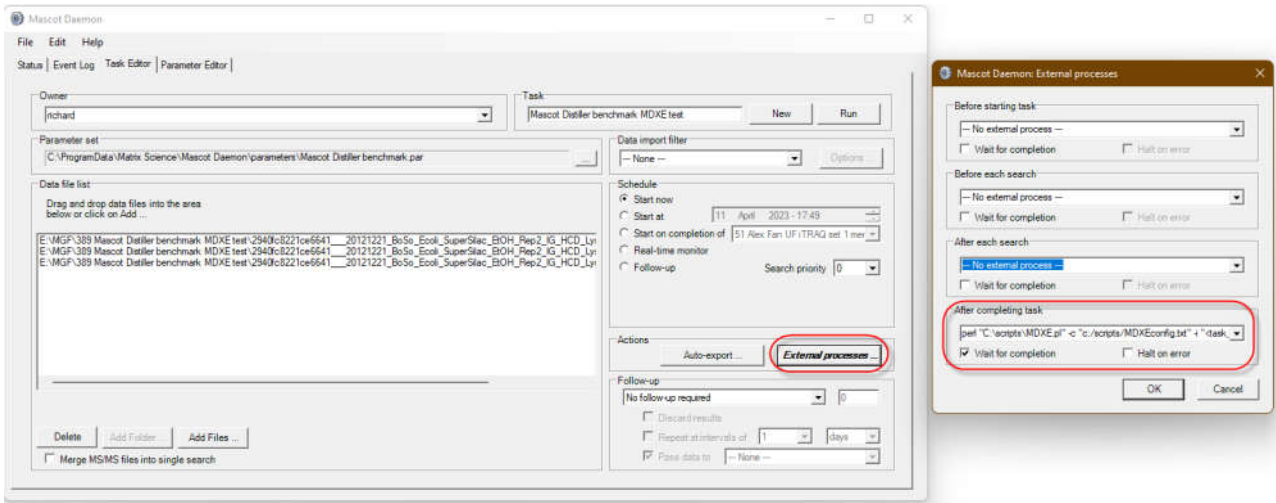
You can download the script from our website

Download [MDXE.zip](#), extract and save MDXE.pl and MDXEconfig\_factory\_default.txt to a directory on the Mascot Daemon computer. For example, C:\scripts.

Make a copy of MDXEconfig\_factory\_default.txt. Enable (uncomment) or disable (comment out) the commands for the Distiller reports you need. It's best to have different configuration files for different users and workflows.

Older versions of Mascot Server also require a copy of Perl so you won't need to install it again in that case.

# Using MDXE in Daemon



```
perl "C:\scripts\MDXE.pl" -c "C:\scripts\My_MDXEconfig.txt" -i "<task_directory>" -l
```

**MASCOT**

: Mascot Daemon Export Extender

© 2024 Matrix Science



In order to use it set up an external task with a command line like this:

## Documentation

- **Blog post**
  - <https://www.matrixscience.com/blog/mascot-daemon-export-extender.html>
- **Configuration file**
  - The included configuration file includes descriptions of all the parameters and examples for all the built in Mascot Distiller reports.

You can find documentation in the blog post and the configuration file we include with the script.

## Configuration file

- **Uses tokens for variables**

- *<exec1> path and name of the executable to be called*
- *<input1> file extension of the files to be processed*
- *<output1> file extension of the files to be created*
- *<taskDir1> path to the data directory*
- *<command1> command that MDXE will run with tags in the place of any variables*

Within the configuration file we use tokens to for variables. This keeps the length of the commands down and makes them more readable.

We can reuse the tokens in multiple commands too.

The tokens cover the input, output, starting directory and final commands themselves.

## Example: call Mascot Distiller to create a protein quantitation report

- `exec1=C:\Program Files\Matrix Science\Mascot Distiller\MascotDistiller.exe`
- `input1=.rov`
- `output1=proteins.csv`
- `command1=<exec1> <input1> -batch -quantout <output1> -quantreport table-proteins.py -exportFormat CSV`

Here is an example for a Mascot Distiller quantitation report that generates a table of proteins.

We set the executable, input file extension, and output file extension.  
Then use the tokens to build the command line.

Each time the command line is called it will replace the tokens with the current file and other parameters.

## At runtime the command is expanded with the dynamic values

- `<exec1> <input1> -batch -quantout <output1> -quantreport table-proteins.py -exportFormat CSV`

```
"C:\Program Files\Matrix Science\Mascot Distiller\MascotDistiller.exe"
```

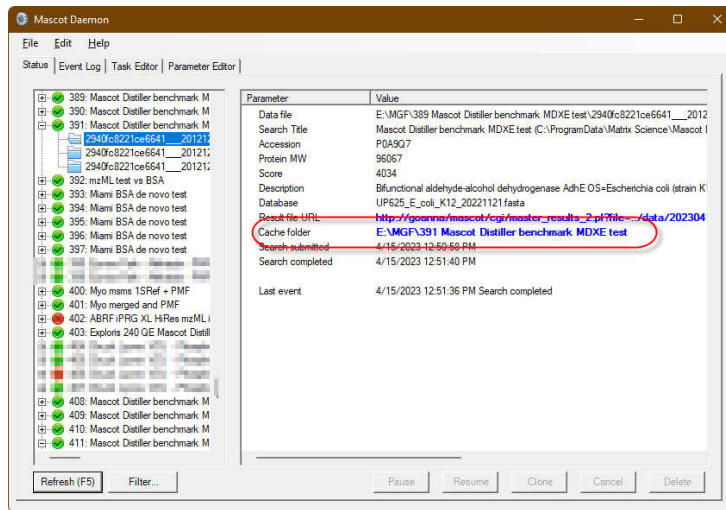
```
"C:\ProgramData\Matrix Science\Mascot Daemon\MGF\166 MDXE  
test\f4c0612f0fac0df8___mydata.raw.-1.rov"
```

```
-batch -quantout
```

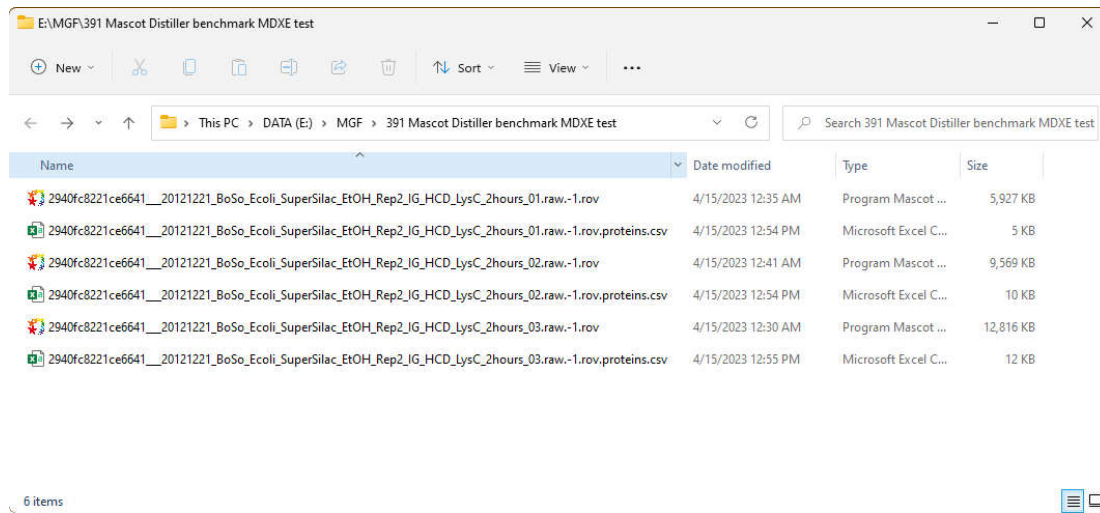
```
"C:\ProgramData\Matrix Science\Mascot Daemon\MGF\166 MDXE  
test\f4c0612f0fac0df8___mydata.raw.-1.rov.proteins.csv"
```

```
-quantreport table-proteins.py -exportFormat CSV
```

Like this:



Once the task is run it will pick up files in the tasks cache or MGF folder, the Mascot Distiller rov files in this case



**MASCOT**

: Mascot Daemon Export Extender

© 2024 Matrix Science



And for each rov file use Mascot Distiller to generate the protein table reports which it saves to the same directory.



## Not just from Mascot Daemon

- As MDXE.pl runs from the command line you can put all the files you wish to process into one folder and process them as a batch
- Configure the command you want to run and set the input file
- `perl "C:\temp\MDXE.pl" -c "C:\ temp\custom_MDXEconfig.txt" -i "G:\Temp\391 MD benchmark MDXE reanalysis" -l`


Lets say you have some old data that you want to generate some reports from. Instead of rerunning the files with this script we can take the existing results, place them in a folder together and process as a batch.

The screenshot shows a Windows File Explorer window titled "G:\Temp\391 MD benchmark MDXE reanalysis". The address bar shows the path "This PC > Data2 (G:) > Temp > 391 MD benchmark MDXE reanalysis". The file list contains 16 items, including CSV and LOG files with names like "2940fc8221ce6641\_\_20121221\_BoSo\_Ecoli\_SuperSilac\_EtOH\_Rep2\_IG\_HCD\_LysC\_2hours\_03.raw.-1.rov.peptides.csv".

Overlaid on the File Explorer is a Command Prompt window with the following command:

```
C:\>perl "C:\temp\MDXE.pl" -c "C:\temp\custom_MDXEconfig.txt"
-i "G:\Temp\391 MD benchmark MDXE reanalysis" -l
C:\>
```

At the bottom of the screenshot, there is a footer with the following information:

**MASCOT** : Mascot Daemon Export Extender © 2024 Matrix Science 

In this case using the Mascot Distiller rov files generating both the table of proteins and the table of peptides reports.

## Not just Mascot Distiller reports

- There is enough flexibility to run many other command line programs.
- Using an AI chatbot it is now easy to create command line programs to do simple tasks

We can go beyond Mascot Distiller reports and run other programs that have multiple inputs. I'm going to give a very simplified example, but I tend to find customers have far more complex and interesting needs.

For this I am going to use an AI chatbot to write a command line script.

write a python script that runs on the command line. The script copies a file from one location to another. Take the input file name and path from the input variable "-i" and move it to the output directory "-o"

To create a Python script that copies a file from one location to another using command line arguments, you can use the `argparse` module for handling the command line arguments and the `shutil` module for the file operations. Below is a complete script that accomplishes this:

```
python Copy code

import argparse
import shutil
import os

def copy_file(input_path, output_dir):
    # Check if the input file exists
    if not os.path.isfile(input_path):
        print(f"Error: The input file '{input_path}' does not exist.")
        return

    # Check if the output directory exists
    if not os.path.isdir(output_dir):
        print(f"Error: The output directory '{output_dir}' does not exist.")
        return

    # Get the base name of the input file
    file_name = os.path.basename(input_path)
```

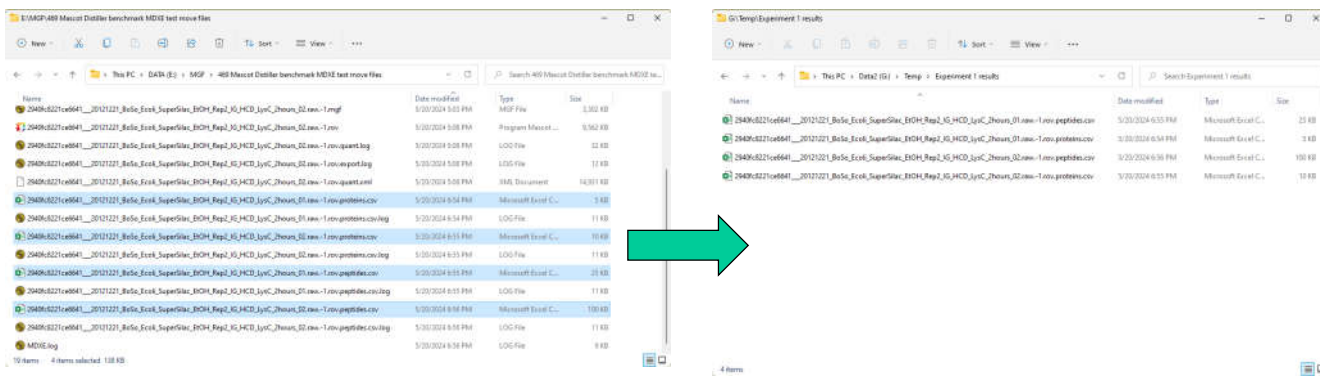
It helps if you can think a bit like a programmer and write a good prompt. You can refine it too, but I didn't need to in this case.

Here is a bit of the output. I saved the program as a text file with extension .py in a scripts directory.

```

exec1=C:\Program Files\Mascot Science\Mascot Distiller\MascotDistiller.exe
exec2=C:\temp\movefile.py
input1=.rov
input2=.csv
output1=.proteins.csv
output2=.peptides.csv
command1=<exec1> <input1> -batch -quantout <output1> -quantreport table-proteins.py -logfile <output1>.log -loglevel 7 -exportFormat CSV
command2=<exec1> <input1> -batch -quantout <output2> -quantreport table-peptides.py -logfile <output2>.log -loglevel 7 -exportFormat CSV
command3=<exec2> -i <input2> -o "G:/Temp/Experiment 1 results/"

```



**MASCOT**

: Mascot Daemon Export Extender

© 2024 Matrix Science



I can then add this script to the MDXE configuration file.

MDXE now uses Mascot Distiller to create the two quantitation reports and then calls the new python script to copy the .csv files generated by Distiller to a new directory.

Technical note. You need to be able to run python scripts from the command line. In order to do this python file type .py needs to be registered and the executable added to the path. On Windows you can do this with the following commands at the command prompt running in administrator mode:

```
C:\> assoc .py=Python
```

```
C:\> ftype Python="C:\python27\python.exe %1 %*"
```

Note that you need to adjust the path to python to match your python installation. Many people use anaconda and have multiple installation of python and you can pick any default python installation for this script.

## Conclusions

- **Mascot Daemon external processes are the key to automation**
- **MDXE can export any Mascot Distiller report(s)**
- **Can call other custom scripts or programs**
- **Works on the command line too**

So in Mascot Daemon the External Process dialog is the key to automation above and beyond what Mascot Daemon can do already.

We have made a utility available, Mascot Daemon Export Extender, that makes it a bit easier to use this feature to generate Mascot Distiller quantitation reports or other tasks. MDXE works on the command line too and can be used for general batch automation.

If you have a task you want to automate after the searches are complete and run into difficulties, please let us know and we may be able to help.