

Pattern Search User Guide

Introduction

The pattern search application is a program that uses regular expressions to find patterns in files. A [regular expression](#) is a standard textual syntax for representing patterns that is used to match text. The pattern search application operates as a stand-alone program that performs its tasks locally.

You can use the pattern search command line interface (*PatternSearch.exe*) or the graphical user interface (PatternSearchUI) to:

- Search for patterns in files in a directory and optionally subdirectories
 - Supported file types include: pdf, docx, xls, xlsx, csv, txt
- Identify image types and quantity embedded in certain file types (pdf and docx)

Command Syntax

The pattern search command syntax is:

```
$ patternsearch -d < directory> [application options]
```

Application Options:

<i>-r, --recursive</i>	<i>Recursively process directory</i>
<i>-c, --csv</i>	<i>Output the results in a CSV format</i>
<i>-v, --verbosity</i>	<i>Verbosity level: B=Basic, M=Moderate, V=Verbose</i>
<i>-i, --imagescan</i>	<i>Scan file to determine if images exists</i>
<i>--help</i>	<i>Display help screen</i>
<i>--version</i>	<i>Display version information</i>

Example #1

Scan the current directory, output verbose results in CSV format, save the results to *out.csv*, and errors to *out.err*

```
$ patternsearch -d . -c -v V > out.csv 2> out.err
```

Example #2

Scan the current directory and all sub directories, output moderate results, save the results to *out.txt*, and errors to *out.err*

```
$ patternsearch -d . -r -v M > out.txt 2> out.err
```

Example #3

Scan the current directory, output basic results, scan file for images, save the results to *out.txt*, and errors to *out.err*

```
$ patternsearch -d . -i -v B > out.txt 2> out.err
```

Sample Output

The command below will produce the following output:

```
$ patternsearch -d . -r -v B > out.txt 2> out.err
```

```
*****
(1/1/2019 1:00) Processing files with the following parameters:
Recursive='true' Directory='.' Verbosity='B'
In the following file types: '.pdf; .txt; .csv; .docx; .xlsx; .xls'
*****
Number of possible patterns found: 1 in a.txt

Number of possible patterns found: 5 in b.pdf

*****
(1/1/2019 1:00) Finished processing files:
Files Processed='2' Processing Time (seconds) ='1'
Searched for the following patterns: 'SSN, Visa'
*****
```

The command below will produce the following output:

```
$ patternsearch -d . -c -r -v B > out.csv 2> out.err
```

```
File Name,Text Size,Possible Match Count
```

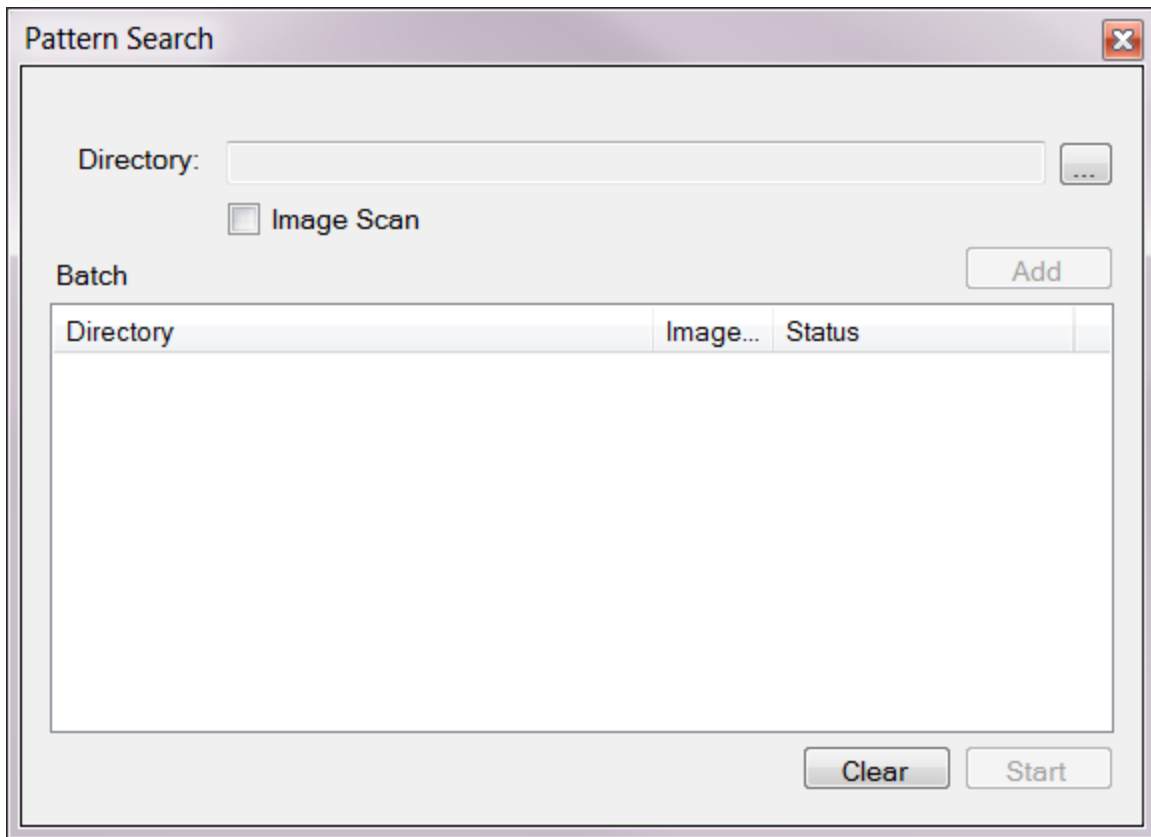
```
a.txt,500,1
```

```
b.pdf,1000,5
```

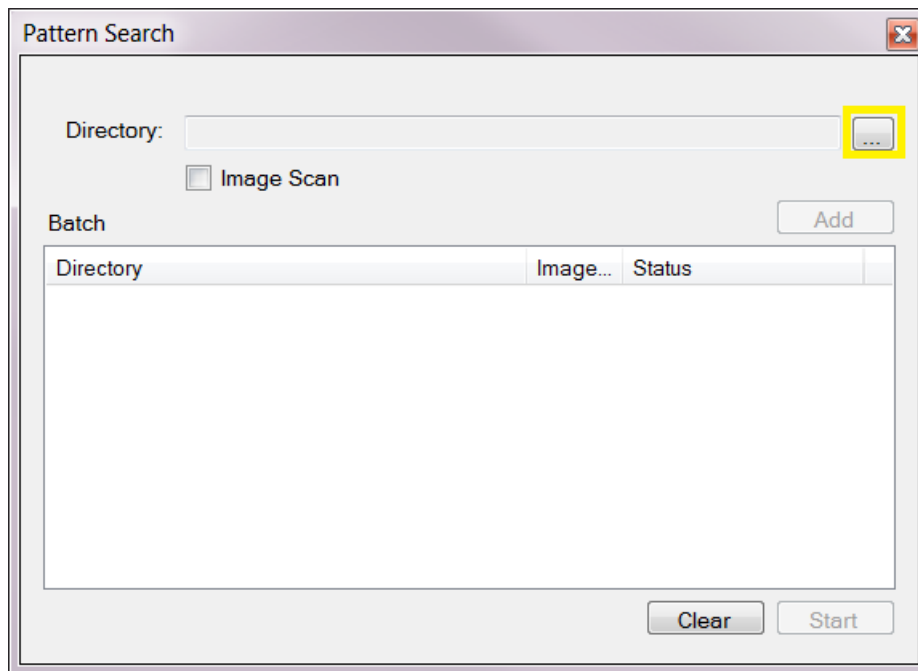
```
Scan Started at: 1/1/2019 1:01, Finished at: 1/1/2019 1:01
Files Processed='2', Processing Time (seconds) ='1'
Patterns searched 'SSN, Visa'
```

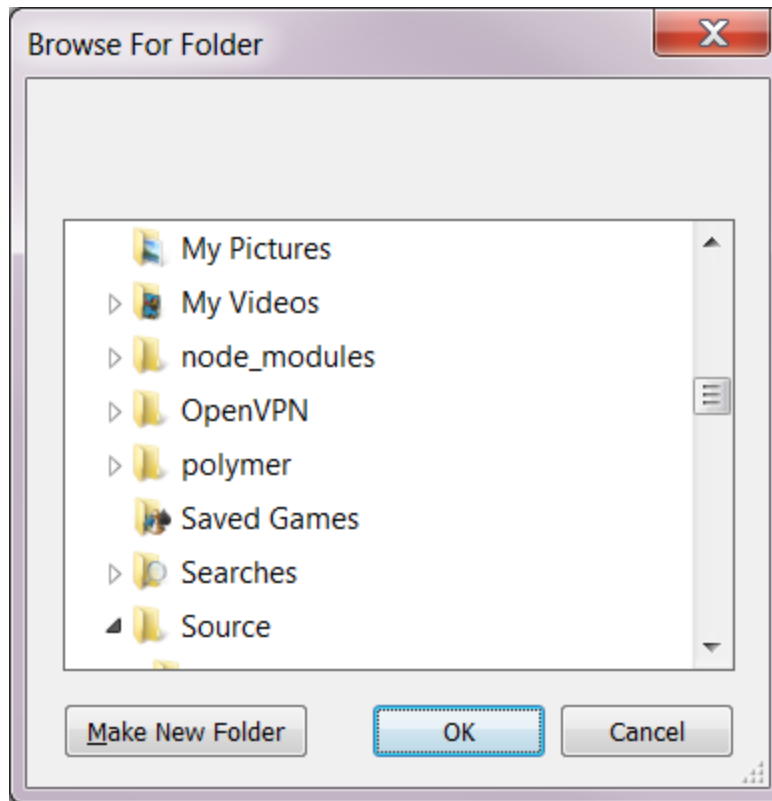
Graphical User Interface

The graphical user interface (GUI) provides the ability to batch multiple directories for processing. The GUI generates a CSV file with verbose output. When creating the batch, there is an option to scan each file in a directory to determine if images exist in the file.



To add a directory to the batch, simply click the **ellipses** and a dialog will appear to select a directory.





Once a directory is identified, simply click the **Ok** button, optionally click the **Image Scan** check box, and click the **Add** button. Repeat the process until a batch is complete. Once the batch is complete, click the **Start** button and the scan will begin. The scan results will be generated with the same name as the directory appended with a .csv extension. For example, if the *Source* directory was scanned, the output file would be *Source.csv*. The file will be located in the same directory as directory being scanned. For example, if the *Source* directory was located in *C:\temp*, the *Source.csv* output file would also be located in *C:\temp*. To create a new batch, simply click the **Clear** button and restart the process.

Configuration

The application relies on the configuration file for information on what to search for and how to filter the results. Patterns are defined as regular expressions and are grouped together in the configuration file and are formatted as name/value pairs. Below is an example of the patterns section in the configuration file:

```
<Patterns>
  <add key="Bank RTN" value="\b((0[0-9])|(1[0-2])|(2[1-9])|(3[0-2])|(6[1])|(7[02])|80)([0-9]{7})\b"/>
  <add key="FOUO" value="(?)\bFOUO\b" />
  <add key="SSN Title" value="(?)\bSSN|Social Security Number"/>
  <add key="SSN" value="\b\d{3}-\d{2}-\d{4}\b" />
  <add key="VISA" value="\b4[0-9]{12}(?:[0-9]{3})?" />
  <add key="MC" value="\b(?:5[1-5][0-9]{2}|222[1-9]|22[3-9][0-9]|2[3-6][0-9]{2}|27[01][0-9]|2720)[0-9]{12}\b" />
  <add key="AMEX" value="\b3[47][0-9]{13}\b" />
  <add key="Discover Card" value="6(?:011|5[0-9]{2})[0-9]{12}" />
  <add key="Phone" value="\b(?:([0-9]{3})[\(\) -.]([0-9]{3})[-.]([0-9]{4})\b" />
  <add key="EMail" value="[a-z0-9]([-a-z0-9._]@([a-z0-9]+\.)+[a-z]{2,5})" />
  <add key="Zip Code" value="[0-9]{5}(?:-[0-9]{4})" />
  <add key="DOB" value="(?)\b\dob\b|date of birth" />
  <add key="VIN" value="([A-Z\d]{3})[A-Z]{2}\d{2}([A-Z\d]{1})([X\d]{1})([AZ\d]{3})\d{5}" />
</Patterns>
```

The filters section in the configuration help filter out information that matches the pattern but should be excluded from the results. For example, you may want to identify all phone number patterns in a file, but are not interested in phone numbers that start with 301-504. The filters section has filters which are also defined as regular expressions and are formatted as name/value pairs. Below is an example of the filters section in the configuration file:

```
<Filters>
  <add key="EMail" value=".gov|.mil" />
  <add key="Phone" value="^301-504" />
  <add key="Zip Code" value="20814" />
</Filters>
```