

Lightgrep™ - Fast Search for Forensics

Lightgrep for EnCase[[1]](#footnote-2) provides you with new options for search in a familiar environment: Keywords that are compatible with Perl (PCRE); filters for search terms to help you find the real information you're looking for faster; text extraction for Office 2007 and similar document formats.

Table of Contents

[Getting Help 5](#_Toc327313635)

[Minimum Requirements 5](#_Toc327313636)

[Known Issues 5](#_Toc327313637)

[Limitations and General Information 5](#_Toc327313638)

[Licensing 7](#_Toc327313639)

[Setup & Installation 8](#_Toc327313640)

[File Locations 13](#_Toc327313641)

[Search Tab 15](#_Toc327313642)

[Options 15](#_Toc327313643)

[Lightgrep Output Folder 15](#_Toc327313644)

[Create Bookmarks 16](#_Toc327313645)

[Create Excel Report 16](#_Toc327313646)

[One Bookmark Per File 16](#_Toc327313647)

[Use Export Folder 16](#_Toc327313648)

[Search only selected items 16](#_Toc327313649)

[Search for Keywords 16](#_Toc327313650)

[Search for Entities 16](#_Toc327313651)

[Ignore all items in loaded hash library (search slack) 16](#_Toc327313652)

[Ignore items with hash category ‘KNOWN’ (search slack) 16](#_Toc327313653)

[Search Status 17](#_Toc327313654)

[Search (Button) 17](#_Toc327313655)

[Keywords 18](#_Toc327313656)

[Options 18](#_Toc327313657)

[Import From EnCase 18](#_Toc327313658)

[Add Keyword List 20](#_Toc327313659)

[Import / Export XML 21](#_Toc327313660)

[Save Settings 21](#_Toc327313661)

[Keyword Tree 21](#_Toc327313662)

[Editing Keywords 21](#_Toc327313663)

[Supported Pattern Syntax 23](#_Toc327313664)

[Individual Code Points 23](#_Toc327313665)

[Breakout Bytes (\zhh) 23](#_Toc327313666)

[Metacharacters 24](#_Toc327313667)

[Character Classes 24](#_Toc327313668)

[Character Class Metacharacters 24](#_Toc327313669)

[Named Character Classes 25](#_Toc327313670)

[Repetition 25](#_Toc327313671)

[Greedy and Reluctant Matching 25](#_Toc327313672)

[Conjunction, Alternation, Grouping 26](#_Toc327313673)

[Comparison With EnCase 26](#_Toc327313674)

[Entity Search 28](#_Toc327313675)

[Options 29](#_Toc327313676)

[Entities 29](#_Toc327313677)

[Entity Filter 29](#_Toc327313678)

[Validation Lists 29](#_Toc327313679)

[Import / Export Entities 29](#_Toc327313680)

[Add Sample Entities 29](#_Toc327313681)

[Compile Filters 30](#_Toc327313682)

[Save Settings 30](#_Toc327313683)

[Editing Entities 30](#_Toc327313684)

[Advanced Tab 32](#_Toc327313685)

[Options 33](#_Toc327313686)

[File Signature List 33](#_Toc327313687)

[Import From EnCase 33](#_Toc327313688)

[Reset Defaults 33](#_Toc327313689)

[Run Lightgrep automatically 34](#_Toc327313690)

[Lightgrep executable path 34](#_Toc327313691)

[Excelutil executable path 34](#_Toc327313692)

[Number of Threads 34](#_Toc327313693)

[Log Level 34](#_Toc327313694)

[Single File Bookmark Threshold 34](#_Toc327313695)

[Save Settings 34](#_Toc327313696)

[Starting a Search 35](#_Toc327313697)

[Monitoring Search Progress 36](#_Toc327313698)

[Reviewing Output 37](#_Toc327313699)

[Bookmark Output 37](#_Toc327313700)

[Excel Output 38](#_Toc327313701)

[Appendix A - Regular Expressions 40](#_Toc327313702)

[Example 1 40](#_Toc327313703)

[Appendix B - Entity Filters 42](#_Toc327313704)

[EntityHit Class Reference 42](#_Toc327313705)

[Members 42](#_Toc327313706)

[Usage Notes 42](#_Toc327313707)

[Example 1 42](#_Toc327313708)

# Getting Help

* We’re here to help, and we want you to be successful using Lightgrep! If you have any comments, questions, or need to report a bug, please email us at [info@lightboxtechnologies.com](mailto:info@lightboxtechnologies.com).

# Minimum Requirements

* EnCase Forensic or Enterprise version 6.16.2 (32-bit or 64-bit) or greater
* Windows 7 (64-bit) – the application has not been tested with Windows XP or Vista, but should function without issue on the 64-bit version of either operating system
* 4 GB of RAM preferred minimum
* Microsoft .Net Framework 3.5 SP1 is required for Excel output

# Known Issues

* 1.0: This version of Lightgrep is 64-bit, which requires your Windows installation to be 64-bit. EnCase can be either 32-bit or 64-bit.
* 1.0: Sample values in the Edit Keyword or Edit Entity dialogs are only displayed in UTF-16LE regardless of encodings selected in the dialog. The displayed samples could appear incorrect in some cases due to the translation from certain encodings. This only affects the displayed samples and does not affect the actual keywords or search hits.
* 1.0: Not compatible with EnCase version 7.
* If you have an extremely large number of search hits, the search hits file in the Output Folder can grow fairly large. The file will likely reach 4 GB in size after approximately 150 million hits, depending on the keywords you’re using. If the Output Folder is on a FAT32 drive or similar which has a relatively small file size limitation, the search may fail. We suggest that you always use NTFS for the Output Folder file system.

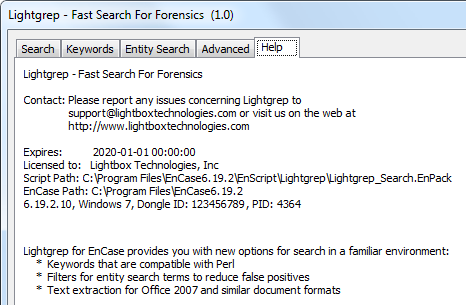
# Limitations and General Information

* We are unable to output search results to the EnCase “Search Hits” tab, because this is not allowed by the EnScript API. Only EnCase’s built-in search functionality can save search hits to this location. Because of this limitation in the EnCase product, we save search hits to the “Bookmarks” tab. It should be noted that Search Hits in EnCase are simply Bookmarks which contain references to the keyword that was searched for, so you’re not missing any data that would otherwise be present in the EnCase search hits results.
* Guidance Software has previously stated that Sweeping Bookmarks in EnCase v6 have a limit of 16,384 bytes. In our testing, however, Sweeping Bookmarks work fine for much larger sizes when created by an EnScript. Lightgrep does not have a technical limit on the length of search hits. Any limits within EnCase may present a practical limitation and cause issues on extremely large search hits when the desired output format is an EnCase Bookmark.
* We filter a few entries by default before searching. Spcifically, any entries with a Physical Size of 0 bytes is filtered out; there’s no reason to search these because they have no content. Also, we filter out the NTFS bad cluster map file using the following criteria: file name in EnCase entry view is "$BadClus·$Bad" and the entry description contains both “Internal” and “Stream”.
* Entries without a starting extent cannot be bookmarked. These usually are files containing Invalid Cluster in the Description field, although some files marked as Internal, such as the streams $UsnJrnl·$J and $BadClus·$Bad, also fall into this category.
* While Lightgrep is able to search entries flagged as Overwritten, bookmarks containing hits may point to the overwriting file, not to the overwritten entry that you originally selected. This is a limitation of the bookmarking function in EnCase v6.
* The “Lightgrep Output Folder” location cannot be set to a protected location in Windows Vista or Windows 7. Protected locations include, but may not be limited to, those locations represented by the following Windows environment variables: *SystemRoot*, *ProgramFiles(x86)*, *ProgramW6432* (e.g. C:\Windows, C:\Program Files, C:\Program Files (x86)). Lightgrep for EnCase will attempt to automatically identify if you are trying to save output to one of these locations and ask you to correct it.
* Lightgrep stores settings in two files: *Lightgrep\_Search.ini* and *Lightgrep\_Search.xml*. The .ini file stores simple options that have been selected in the GUI; the .xml file stores Entities and Keywords. The .xml file uses approximately 75 MB of space per 100,000 keywords.
* Lightgrep uses a constant and relatively small amount of memory while searching, but can require more memory up front while parsing the keyword list depending on the number of keywords. 4 GB of RAM with no other applications running should allow you to parse and search approximately 100,000 fixed string keywords. A minimum of 8 GB of RAM is required to parse and search 1 million fixed string keywords, and more could be required if using GREP expressions. When searching 1 million or more keywords, it can take several minutes before the search begins while the search engine parses the list of keywords.
* The first time you run a search, you may be presented with an alert from your firewall. Lightgrep uses a local network socket to communicate with EnCase. You should Allow Access for lightgrep.exe if prompted to insure there aren’t any problems with searching as shown below.

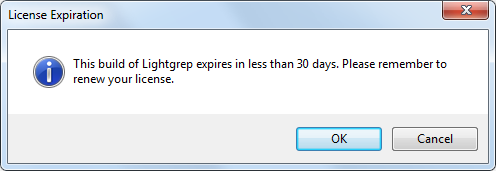


# Licensing

Lightgrep for EnCase is tied to your EnCase dongle ID and is licensed for a 12 month period. The Help tab will show the current expiration date, along with other useful diagnostic information about your installation.

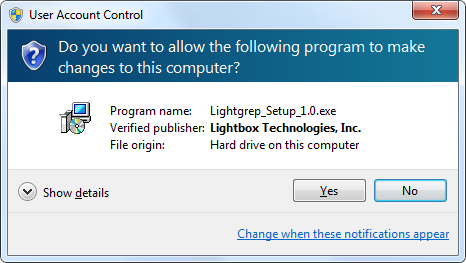


When you are nearing the expiration, a warning message will appear to inform you that the license will expire soon.

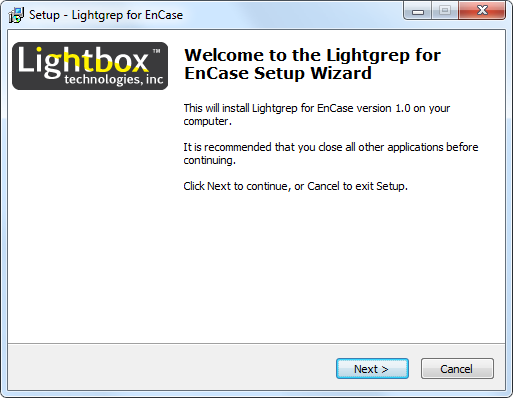


# Setup & Installation

Run Lightgrep\_Setup\_1.0.exe and, if User Account Control is enabled on Windows Vista or Windows 7, you will see a dialog asking for approval to run the installer. Select Yes to continue.

****

The Welcome dialog notes the version of Lightgrep Search for EnCase that you are installing. Click Next on the Welcome dialog to proceed.

****

Please read the License Agreement, select “I accept the agreement,” and click Next to continue. If you do not accept the agreement, you cannot continue with installation.

****

Choose the location to install the Lightgrep executable. The installer should choose the best location for you. Click Next to continue.

****

Select the location of your EnCase 6 directory. The installer will copy the Lightgrep\_Search EnPack and EnLicense into the appropriate EnScript and License folders based on this selection.

****

Choose the Start Menu folder to place Lightgrep shortcuts in. The installer will create shortcuts for the Lightgrep End User License Agreement (EULA), User Manual, and Uninstall application.

****

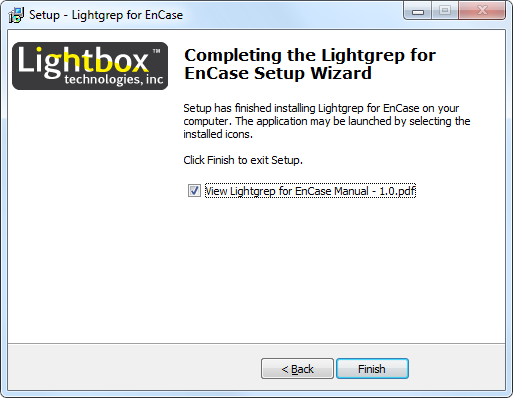
On the Ready to Install dialog, the installer will show you the destination locations you previously selected. Confirm these selections by clicking Install.

****

After installation is complete, an informational message is presented. Click Next.

****

The installation is complete! Leave the box checked to open the Lightgrep Search for EnCase user manual, and click Finish to complete the installer.

****

Run Lightgrep Search for EnCase by opening EnCase and navigating to the EnScript pane, usually in the lower right hand corner of EnCase. Expand the Lightgrep folder and double-click on Lightgrep\_Search to run the application.

****

## File Locations

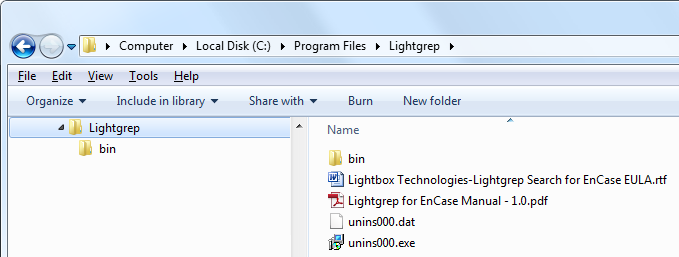
Several files are installed with Lightgrep Search for EnCase. *Lightgrep\_Search.EnPack* is the EnScript application code that allows Lightgrep to integrate with EnCase. It is installed in the *[EnCase6]\EnScript\Lightgrep* folder.



*Lightgrep\_Search.EnLicense* is the EnCase license file, which links Lightgrep Search to your dongle ID(s) and also contains the expiration date, unique to your account. This file is placed in the *[EnCase6]\License* folder.



The main *Lightgrep* folder is typically placed under *Program Files* and, at the top level, contains the EULA, user manual, and the uninstall application.

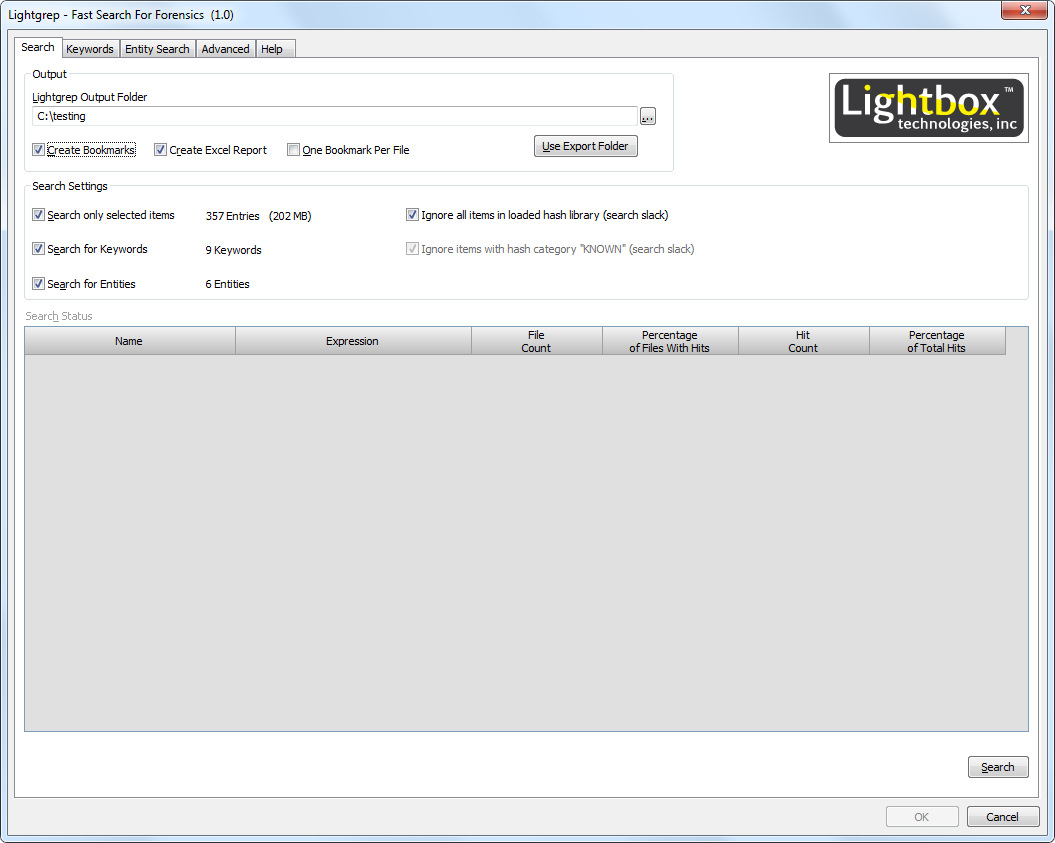


Finally, the *bin* folder, located under the *Lightgrep* folder, contains the Lightgrep executable, the excelutil helper application to create Microsoft Excel output, and a Microsoft library file, which is also required for Excel output.



# Search Tab

The Lightgrep Search Tab as shown below presents the main program options that should be reviewed before performing a search, and also provides an area to display ongoing output about search statistics.



## Options

### Lightgrep Output Folder

The folder specified here will be used both as a temporary folder location and as the output location for Lightgrep search results. Generally speaking, less than 10 megabytes of space is used by temporary files (not including the search hits files) from the application, depending on the number and size of keywords. The size of results files can vary greatly depending on the number of hits found. As mentioned in the Known Issues section, the search hits file can grow to several GB if your search returns 100 million hits or more.

### Create Bookmarks

This option will create sweeping bookmarks of all search hits in the current EnCase Case. The output is very similar to EnCase search hits. Search hits will be organized into the same folder structure as present in the Keywords tree (see **Error! Reference source not found.**). If you have a folder named “Financial” containing 5 keywords, you will also have a bookmarks folder named “Financial” which will contain one folder for each of the 5 keywords that have hits.

### Create Excel Report

This option will create an Excel 2007 format overview report, containing statistics about the results of the search. The Microsoft .NET Framework 3.5 SP1 is required for output. It is not required to have Microsoft Excel installed in order to produce the report.

### One Bookmark Per File

Selecting this option will cause bookmarks to be created per file instead of per hit. If the total number of hits generated by Lightgrep surpasses the allowed total number of hits as specified on the Advanced tab in the “Single File Bookmark Threshold” setting, this checkbox will be ignored and Lightgrep will always produce one bookmark per file, per keyword.

### Use Export Folder

This button sets the Lightgrep Output Folder to the current Case Export Folder.

### Search only selected items

This option determines whether all Entries or only selected (blue checked) Entries will be searched. Note that Records are not searched directly, but their content can be searched by selecting the appropriate Entries. Information about the number and size of selected Entries is displayed for reference.

### Search for Keywords

When this is selected, Lightgrep will search for the keywords selected in the Keywords tab. See **Error! Reference source not found.**.

### Search for Entities

When this option is selected, Lightgrep will search for entities using the options on the Entity Search tab. Keyword hit results will be restricted based on each entity’s filter code. See the Entity Search tab description for further information.

### Ignore all items in loaded hash library (search slack)

This option will prevent searching of any item that is present in the currently loaded hash library. To learn more about loading hash sets into the active hash library, please consult the EnCase Help file sections named “Hash Sets” and “Rebuilding a Hash Library.” Slack space will always be searched regardless of this setting. Selecting this item disables the following option.

### Ignore items with hash category ‘KNOWN’ (search slack)

This option will prevent searching of any item with a hash category of ‘KNOWN’ (case insensitive) that is present in the currently loaded hash library. To learn more about loading hash sets into the active hash library, please consult the EnCase Help file sections named “Hash Sets” and “Rebuilding a Hash Library.” Slack space will always be searched regardless of this setting. This item will be disabled if the preceding option is selected.

### Search Status

The Search Status window displays statistics about the currently running search. You will see the keyword name, search expression, a count of how many files have hits for each keyword, the percentage of files with hits, the total count of hits produced by that keyword, and the percentage of total hits related to that keyword.

### Search (Button)

The Search button starts the search using all previously selected criteria.

# Keywords

Lightgrep keywords are a little different from EnCase keywords. For one, Lightgrep keywords are PCRE compatible. Lightgrep supports more operators, longer repetition, and longer search hits. Unfortunately this means we can't use EnCase keywords, so you won't see your EnCase keywords here automatically. Supported keyword syntax is discussed in the section Supported Pattern Syntax.

## 

## Options

### Import From EnCase

Clicking this button will open the *Import Keywords from EnCase* dialog as shown below. The leftmost tree pane shows folders in both Global Keywords and Case Keywords from EnCase. Use this pane and the pane on the right to select (blue-check) the keywords you would like to import. See Supported Pattern Syntax for information on how the patterns are imported.

The right tree pane shows Lightgrep Keywords folders. The folder selected (highlighted) here will receive the imported keywords.



At the bottom of the dialog, a checkbox labeled “Retain folder structure when importing” allows you to keep the imported keywords organized as they were in EnCase. The folders named “\*\*GLOBAL KEYWORDS\*\*” and “\*\*CASE KEYWORDS\*\*” are not retained, and are displayed only for informational purposes. In the example from above, the root of Lightgrep Keywords is selected on the right; this will produce folders as shown below.

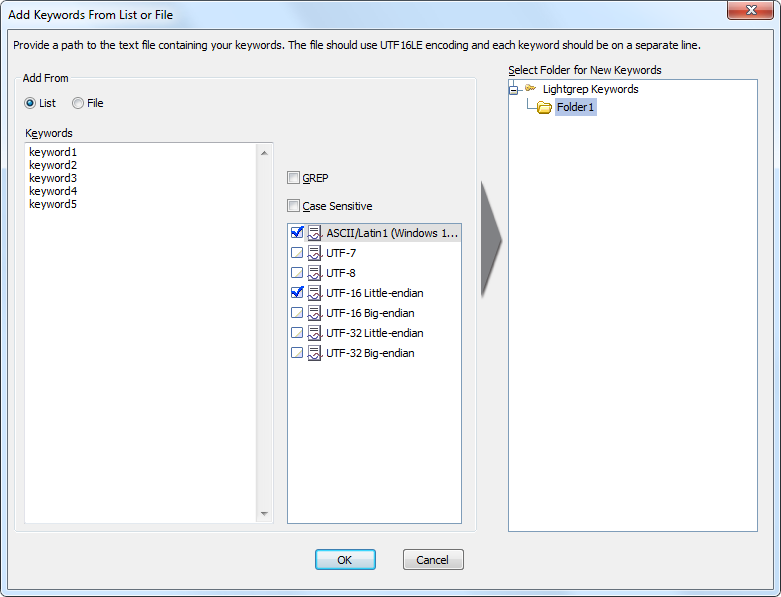


When done, click the OK button and the EnCase keywords will be imported and converted into Lightgrep Keywords. A summary of the import will be presented.

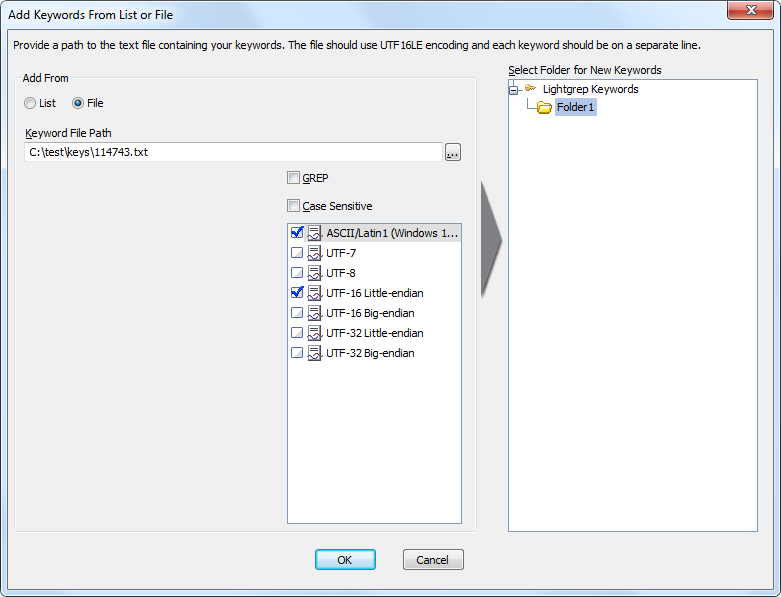


### Add Keyword List

Clicking this button will open the *Add Keywords from List or File* dialog as shown. By default, the “List” option is selected. Copy and Paste a list of keywords, one per line to import. Then, select the proper options for the keywords. The right tree pane shows Lightgrep Keywords folders. The folder selected (highlighted) here will receive the imported keywords.



If you select the “File” option, they will be presented with a path selection box. Choose the path to a text file containing keywords, one per line. Then, select the proper options for the keywords. The right tree pane shows Lightgrep Keywords folders. The folder selected (highlighted) here will receive the imported keywords.



### Import / Export XML

These options allow you to export selected keyword folders to an XML file for later import or for import on another installation of Lightgrep for EnCase. Once exported, you can click the “Import XML” button and import the exported keywords to the folder of their choosing.

### Save Settings

Clicking the “Save Settings” button will save all changed options, including Keywords and Entities, to the script’s storage files. After clicking “Save Settings,” the saved settings will be available the next time you run the script even if you cancel the dialog without running a search.

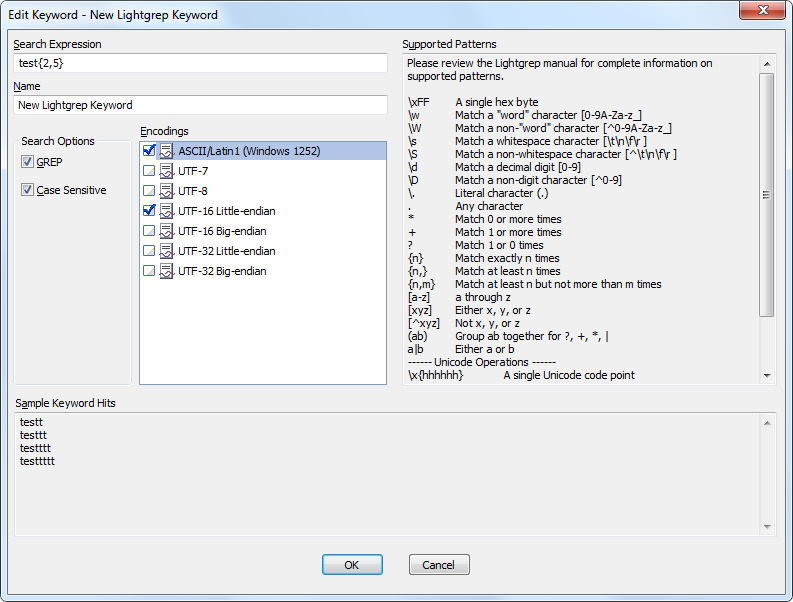
### Keyword Tree

The Keyword Tree gives a familiar interface to view, organize, edit, and delete keywords. Keywords can be organized into folders on the left (tree) side of the interface just as any other dialog within EnCase. The heading / title directly above the keyword tree shows how many keywords are currently selected. **Only keywords which are blue checked will be searched by Lightgrep.**

### Editing Keywords

You can create a new keyword by right-clicking anywhere in the tree and selecting “New” from the context menu or edit an existing keyword either by right-clicking on the keyword and selecting “Edit” or by double clicking on the keyword. Upon creating a new keyword or opening one to edit, you will be presented with the *Edit Keyword* dialog as shown below. The fields in the keyword dialog are:

* Search Expression – the fixed string or GREP pattern to search for
* Name – familiar representation of the keyword
* GREP – search using regular expression pattern matching (see Supported Pattern Syntax)
* Case Sensitive – only search for the exact case shown in the expression (i.e. “test” will not hit on “Test” or “teSt” if this is selected)
* Encodings – blue check the desired encodings to search for
* Supported Patterns – displays supported syntax for GREP patterns
* Sample Keyword Hits – displays sample strings that the expression could hit on, or an error message if invalid; these are not necessarily representative of data that may be present in the currently open Case



## Supported Pattern Syntax

### Individual Code Points

|  |  |
| --- | --- |
| ***b*** | where ***b*** is any literal byte except **0x00** and metacharacters |
| **\a** | **0x07**, bell |
| **\e** | **0x1B**, escape |
| **\f** | **0x0C**, form feed |
| **\n** | **0x0A**, new line |
| **\r** | **0x0D**, carriage return |
| **\t** | **0x09**, horizontal tab |
| **\*o*** | **0*o***, where *o* is an octal digit |
| **\*oo*** | **0*oo***, where each *o* is an octal digit |
| **\*ooo*** | **0*ooo***, where each *o* is an octal digit, less than or equal to 0377 |
| **\z*hh*** | **0z*hh***, where **hh** is a hexadecimal digit. This will match any **single** hex byte, **regardless of encoding**. Also referred to as a “**breakout byte**.” |
| **\xhh** | Matches a single Unicode code point. "**hh**" must be 2 hex digits, in the range [0,FF]. |
| **\x{hhhhhh}** | Matches a single Unicode code point. "**hhhhhh**" must be 1 to 6 hex digits, in the range [0,10FFFF]. |
| **\N{U+hhhhhh}** | Matches a single Unicode code point. "**hhhhhh**" must be 1 to 6 hex digits, in the range [0,10FFFF]. |
| **\N{name}** | Matches a single Unicode code point with the given name. E.g., **\N{LATIN CAPITAL LETTER A}** will match "A". Code point names are defined by the Unicode Standard. |
| **\p{property}** | Matches any Unicode code point having the given property. For example,  **\p{Cyrillic}** matches any code point having Cyrillic as the value of its Script property. |
| **\P{property}** | Matches any Unicode code point not having the given property.  **\P{property}** is equivalent to **[^\p{property}]**. |
| **\*c*** | ***c***, where c is any character not listed above |

### Breakout Bytes (\zhh)

The “breakout byte” operator, **\zhh**, will match on a single byte regardless of encoding. In the encoding “windows-1252”, **\z20** would match on the hexadecimal byte value **20**. Using the encoding “UTF-16LE”, it would match exactly the same single byte, **20**; it would not match **20 00**.

**\zhh** is useful for matching well defined byte sequences, such as file headers. It should usually be used along with encoding “windows-1252”.

As an example, let’s say you want to search for the PNG file header, which is represented by the byte sequence **89 50 4E 47 0D 0A 1A 0A**. You would specify encoding “windows-1252” and your pattern for Lightgrep would be **\z89\z50\z4E\z47\z0D\z0A\z1A\z0A**. In fact, to specify a different encoding value, such as UTF-16LE, would be both unnecessary and incorrect, since the header is a byte sequence and cannot be represented in any other form.

### Metacharacters

Outside of character class specifications, the following characters have special meanings: \, |, (, ), ?, +, \*, ., [. To specify a literal backslash, pipe, left parenthesis, right parenthesis, question mark, plus, asterisk, period, or left square bracket, escape it with a backslash. Additionally, literal left curly braces ({) must be escaped outside of character class specifications if they could otherwise be interpreted as the start of ranged quantifiers, but need not be escaped otherwise.

### Character Classes

[*ccspec*] matches any byte in *ccspec*  
[^*ccspec*] matches any byte not in *ccspec*

Where *ccspec* is one or more of the following:

* a byte,
* a byte escape,
* a range formed by bytes or byte escapes, or
* a named character class (except .).

For example, [a-z] will match letters “a” through “z”, [0-9] will match digits “0” through “9”.

You can also use breakout bytes (\zhh) within a character class. If you use only bytes in a character class, the domain of that character class is only bytes. If you use only code points, the domain is only code points. If you use both bytes and code points, and your character class is positive (i.e. does not use ^), then both the bytes and the encodings of the code points will be searched. If you use both bytes and code points, and your character class is negative (i.e. uses ^), then the bytes are subtracted from the encodings of the code points. This can be a bit confusing, so please review the following examples:

* [A] matches the code point 'A' (U+41)
* [\z00] matches the byte 0x00
* [\z00-\z10] matches any byte in the range 0x00-0x10
* [A\z00] matches the code point 'A' and the byte 0x00
* [A-\z00] is not permitted, and produces an error - the endpoints of a range must both be code points or both be breakout bytes
* [^\z00] matches any byte except 0x00
* [^A] matches any code point except 'A'
* [^\z00-\zFF] is not permitted, because it matches nothing
* [^A\z00] matches any code point except 'A' and the code point that has the encoding \x00, if any
* [^A\z00-\zFF] matches all code points except 'A', less all of the code points with single-byte encodings. The results of this term depend on your target encoding. It would be empty for ASCII, but not for UTF-8.

### Character Class Metacharacters

* The right square bracket ] must be escaped inside character class specifications unless it is the first member.
* The hyphen - forms ranges inside character class specifications using the characters immediately preceding and following it, except when - is the first or last character in the class specification or if the previous character is already part of a range, in which case - stands for itself.
* A literal \ must always be escaped.
* All other characters stand for themselves.

### Named Character Classes

**.** equivalent to **[0x00-0xFF]**  
**\d** equivalent to **[0-9]**  
**\D** equivalent to **[^0-9]**  
**\s** equivalent to **[\t\n\f\r]**  
**\S** equivalent to **[^\t\n\f\r]**  
**\w** equivalent to **[0-9A-Za-z\_]**  
**\W** equivalent to **[^0-9A-Za-z\_]**

### Repetition

***S*\*** equivalent to***S*{0,}**, matching greedily***S*+** equivalent to***S*{1,}**, matching greedily***S*?** equivalent to ***S*{0,1}*****S*{*n*,}** equivalent to***S*{*n*,*M*}**, matching greedily ***S*{*n*,*m*}** matches at least ***n*** and at most ***m*** repetitions of ***S***, greedily***S*\*?** equivalent to***S*{0,}?**, matching reluctantly***S*+?** equivalent to***S*{1,}?**, matching reluctantly***S*??** equivalent to***S*{0,1}? *S*{*n*,}?** equivalent to***S*{*n*,*M*}?**, matching reluctantly***S*{*n*,*m*}?** matches at least ***n*** and at most ***m*** repetitions of ***S***, reluctantly

Where ***S*** is an atomic pattern, 0 ≤ ***n*** ≤ ***m***, and ***M*** is a largish integer which presently depends on the size of the memory available on the machine where Lightgrep is run. An ***M*** value of 10,000,000 can be searched successfully with less than 24 GB of RAM, though startup time will be extremely slow.

### Greedy and Reluctant Matching

“Matching greedily,” as referenced above means that preference will be given to the longest hit found. Searching for the expression “.+” will match the entirety of the data being searched, provided the data is at least one byte long.

“Matching reluctantly,” on the other hand, will find the shortest match. Searching for “.+?” will match a single byte at the beginning of the data.

As EnCase’s internal search engine limits the length of matches generated by \* and + to 256 bytes, you may not be used to true greedy matching. Take, for example, the sample expression provided in EnCase for “US Phone Numbers (area code)” seen below.

\(?###[) \-\.]\*###[ \-\.]?####[^#]

This expression contains a \* after the character class [) \-\.] which, using the EnCase search engine, means that any of the characters in the set “) -.” could be repeated 0 to 256 times and produce a match. If we convert this to a Lightgrep expression, we get the following:

\(?\d{3}[ ).-]\*\d{3}[ .-]?\d{4}\D

The difference is, when we search this expression with Lightgrep, those same characters could be repeated far more than 256 times, even thousands or millions of times, and still produce a match. It pays to understand the rules of the item you’re searching for, and being precise in your regular expression construction will help searches run with greater speed. How do we fix this? In real world usage, you should only have a maximum of two of these in succession. Here are some sample cases from real usage: “(123) 456-7890”; “(123).456.7890”; “123-456-7890”; “123.456.7890”; “123 456 7890”. Keeping this in mind, it’s probably better to limit the repetition to a max of 2 instead of specifying the greedy \*:

\(?\d{3}[ ).-]{0,2}\d{3}[ .-]?\d{4}\D

Using shorter repetition will insure the search engine doesn’t waste time matching strings that are way too long to be real phone numbers.

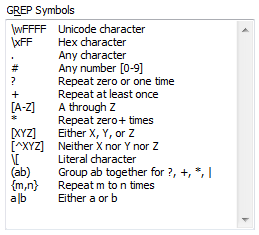
### Conjunction, Alternation, Grouping

***ST*** matches something matching ***S*** followed by something matching ***T******S*|*T*** matches ***S*** or ***T***, preferring matches for ***S***  
**(*S*)** equivalent to ***S***, but is atomic

Individual bytes, character classes, and groups are atomic.

### Comparison With EnCase

You can see that Lightgrep supports many more operators than EnCase GREP does. EnCase uses its own form of GREP syntax, seen below, which is not Perl compatible.

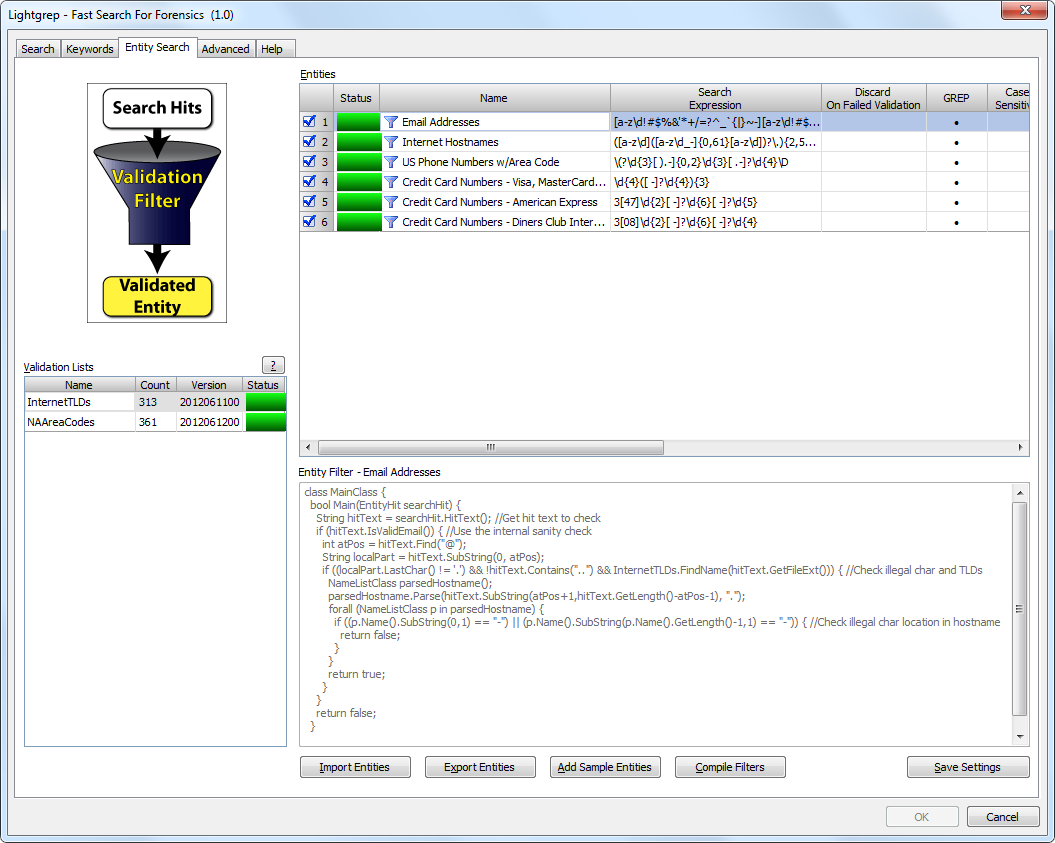


Lightgrep, on the other hand, has millions of tests to validate PCRE compatibility. Here are some notable syntax comparisons:

|  |  |  |
| --- | --- | --- |
| Operator Equivalency in EnCase and Lightgrep | | |
| EnCase | Lightgrep | Explanation |
| \xhh | \xhh | EnCase uses \xhh to denote a single hexadecimal character. In the encoding which EnCase refers to as “ANSI Latin - 1” (windows-1252), \x20 would match on the space character, and in the encoding “Unicode” (UTF-16LE), it would match on 0x20 0x00, also the space character, in UTF-16LE. The same is true for Lightgrep when using \xhh. See “breakout bytes” above for more information on matching single bytes. |
| \whhhh | \x{hhhhhh} | When you perform an import from EnCase keywords, \whhhh will be translated into \x{hhhh}. EnCase accepts four hexadecimal digits which reference a Unicode code point. This covers the Basic Multilingual Plane in Unicode (http://en.wikipedia.org/wiki/Basic\_Multilingual\_Plane), but does not allow referencing the Supplementary Planes. Lightgrep accepts one to six hexadecimal digits for full coverage of all Unicode Planes. |
| # | \d | Equivalent to [0-9] |
| + | {1,255} | When you perform an import from EnCase keywords, + will be translated into {1,255} to prevent accidental lengthy repetitions. EnCase only supports repetition of characters or groups up to 255 times. Lightgrep does not have any limits on repetition. After the import, you may increase the repetition to as much as is necessary. |
| \* | {0,255} | When you perform an import from EnCase keywords, \* will be translated into {0,255} to prevent accidental lengthy repetitions. EnCase only supports repetition of characters or groups up to 255 times. Lightgrep does not have any limits on repetition. After the import, you may increase the repetition to as much as is necessary. |
| \{char} |  | EnCase requires that many characters be escaped in different usages. For example, the pattern [.] will generate an error in EnCase, requiring the . to be escaped. Lightgrep does not require the . to be escaped in this pattern, because its presence within the character class negates its normal usage of matching of any character. Be aware of these differences – it is possible to receive an error message in Lightgrep from “over-escaping.” As an example, Lightgrep will produce an error on \#, because # is not treated as a special character in Lightgrep. |

# Entity Search

Entity Search looks for GREP keywords and then uses filters to verify the results as shown in . The sample filters compare Email addresses and URLs to the list of approved top-level domains to insure validity; check US phone numbers for valid area codes; and compare credit card numbers using Luhn's algorithm. If "Discard On Failed Validation" is selected, the search hit will be ignored completely if the filter returns false, otherwise it will be noted in the bookmark comment. Be aware that filtering search hits for entities can take some time, if the filter code is involved and there are lots of hits. The productivity gain from reducing false positives, however, is well worth the slightly extended filtering time.



Figure

## Options

### Entities

Entities use the same expression rules as regular Lightgrep keywords. You can add new entities, edit and delete existing entities in the same manner as keywords using right click controls.

### Entity Filter

This read-only text box displays the filter code for the currently selected Entity. You must edit the Entity in order to change the filter code.

### Validation Lists

Validation Lists are NameListClass objects (see the EnScript Help within EnCase for an example of working with NameListClass types) that are pre-populated with useful lists of information. For example, the 'InternetTLDs' list is a NameListClass containing every IANA approved top level domain, such as com, edu, uk, etc. The list count and version (a string typically representing the date of the most recent update) are displayed. Double-click on a Validation List name to view its contents.

Validation Lists can be used as NameListClass objects directly in the Main() function of entity filters without initialization:

NameListClass comDomain = InternetTLDs.FindName("com");

Validation Lists are local to the Main() function; they cannot be used outside of Main() without being passed into other functions.

Current Sources:

InternetTLDs  
List of valid Internet top level domains, maintained by IANA (com, edu, uk, etc)  
<http://data.iana.org/TLD/tlds-alpha-by-domain.txt>

NAAreaCodes  
List of active North American area codes, maintained by the North American Numbering Plan Administration  
<http://www.nationalnanpa.com/nas/public/npasInServiceByNumberReport.do?method=displayNpasInServiceByNumberReport>

### Import / Export Entities

These options allow you to export entities to an XML file for later import or for import on another installation of Lightgrep for EnCase. Once exported, you can click the “Import Entities” button and import the exported items.

### Add Sample Entities

Clicking this button will allow you to add the sample entities shown in . Sample entities are provided as a starting pointing and demonstration of the power of entity filtering.

### Compile Filters

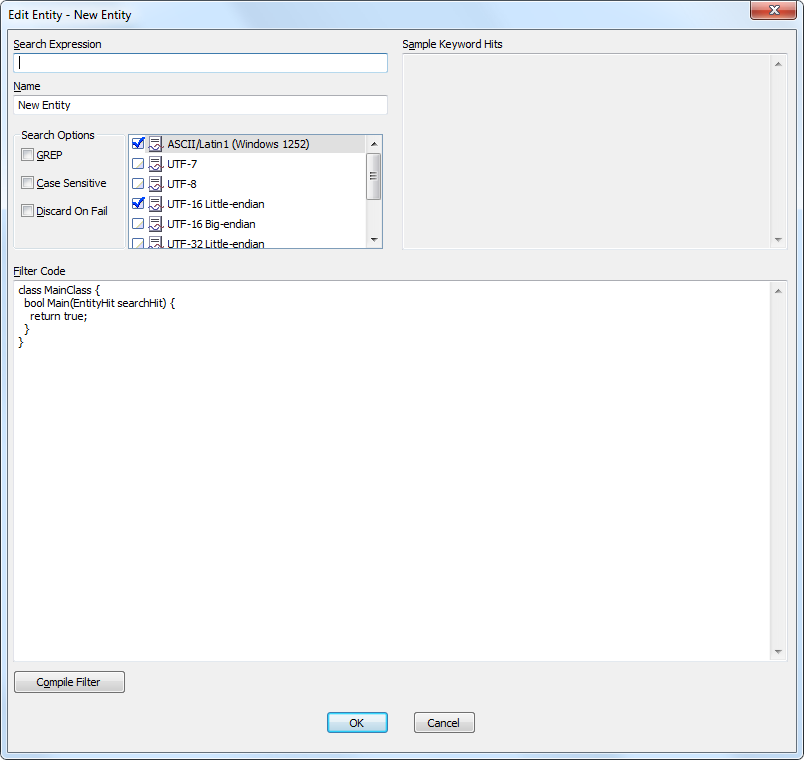
This button will compile all filters in any blue checked entity. If a filter fails to compile, you will see an error message that describes which entity filter could not be compiled.

### Save Settings

Clicking the “Save Settings” button will save all changed options, including Keywords and Entities, to the script’s storage files. After clicking “Save Settings,” the saved settings will be available the next time you run the script even if you cancel the dialog without running a search.

### Editing Entities

If you right-click in the Entity Search tab and create a new entity, or double-click on an existing entity, you’ll see the Edit Entity dialog shown below that lets you edit the entity and the filter.



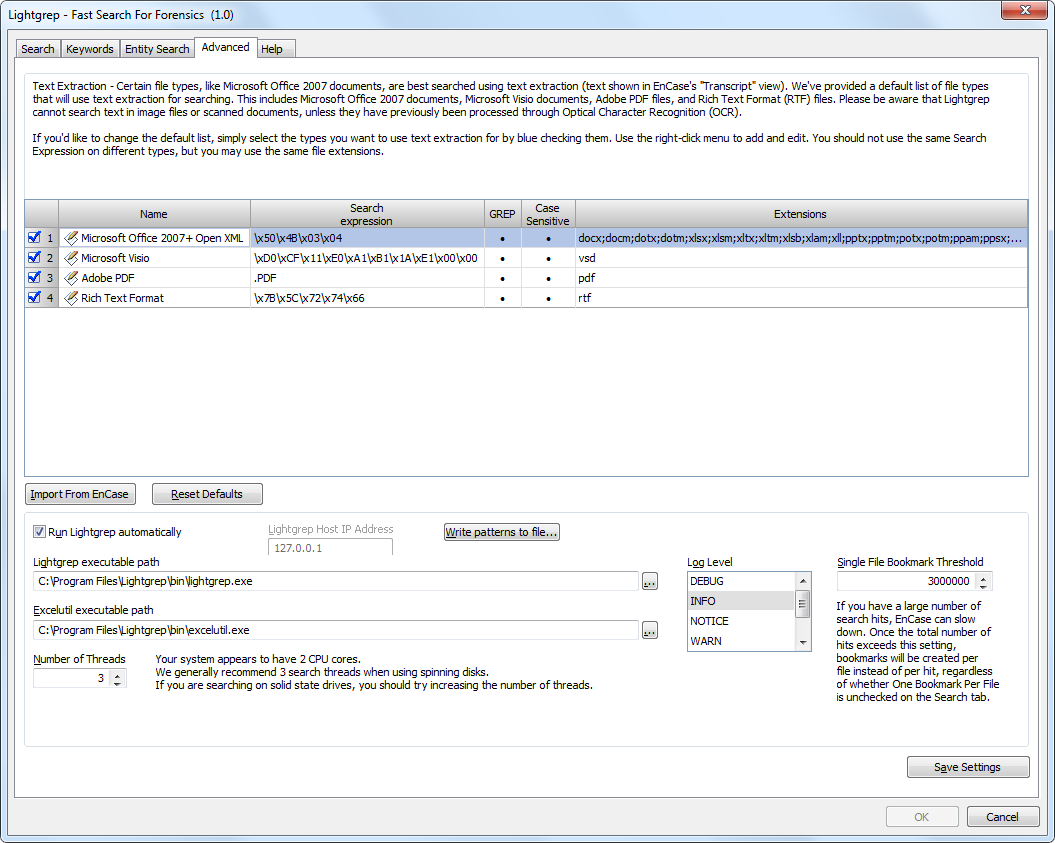
The fields in the Edit Entity dialog are:

* Search Expression – the fixed string or GREP pattern to search for, represents a Lightgrep keyword
* Name – familiar representation of the keyword
* GREP – search using regular expression pattern matching (see Supported Pattern Syntax)
* Case Sensitive – only search for the exact case shown in the expression (i.e. “test” will not hit on “Test” or “teSt” if this is selected)
* Discard On Fail – if checked, the search hit will be ignored completely if the filter returns false; if not checked, a note will be placed in the bookmark comment if the filter returns false
* Encodings – blue check the desired encodings to search for. At least one must be selected.
* Sample Keyword Hits – displays sample strings that the expression could hit on, or an error message if invalid; these are not necessarily representative of data that may be present in the currently open Case
* Filter Code – this area contains an EnScript filter that will be run against search hits from the Search Expression. See Appendix 2 – Entity Filters, for more detailed information.
* Compile Filter – this button will compile the filter in the entity currently being edited. If the filter fails to compile, you will see an error message that describes why it could not be compiled.

# Advanced Tab

Text Extraction - Certain file types, like Microsoft Office 2007 documents, are best searched using text extraction (text shown in EnCase's "Transcript" view). We've provided a default list of file types that will be used text extraction for searching. This includes Microsoft Office 2007 documents, Microsoft Visio documents, Adobe PDF files, and Rich Text Format (RTF) files. Please be aware that Lightgrep cannot search text in image files or scanned documents, unless they have previously been processed through Optical Character Recognition (OCR).

If you'd like to change the default list, simply select the types you want to use text extraction for by blue checking them. You should not use the same Search Expression on different types, but you may use the same file extensions.



## Options

### File Signature List

The File Signature List, as shown in the top portion above, contains a list of file types that will be used to identify files for text extraction. This list contains standard EnCase file signatures. You can add or edit file signatures just as you would in EnCase.

During a search, any file types selected here will have text extracted using EnCase’s Oracle Outside-In technology. Lightgrep will search the extracted text for keywords instead of the binary content of the file. It will also search binary file slack of any files identified for text extraction, so you don’t have to worry about missing anything.

If you deselect any types, you will receive a warning: “You have deselected one or more file types. Any file type that is not selected will be searched using standard native content instead of extracted text.”

### Import From EnCase

Clicking this button will open the *Import File Signatures from EnCase* dialog as shown below. The dialog shows the standard EnCase File Signatures list. Use this window to select (blue-check) the file signatures you would like to import. When done, click the OK button and they will be imported into Lightgrep.



### Reset Defaults

Clicking on the Reset Defaults button will remove any custom file signatures that have been created or imported from EnCase and replace them with the default values.

### Run Lightgrep automatically

This setting should always be checked when performing searches. You should only change this option at the instruction of Lightbox Technologies support for the purposes of debugging.

### Lightgrep executable path

The path to lightgrep.exe on the examiner system. The Lightgrep search executable is run using this path. Usually *C:\Program Files\lightgrep\bin\lightgrep.exe*.

### Excelutil executable path

The path to excelutil.exe on the examiner system. The excelutil executable is run using this path to output the Excel report. Usually *C:\Program Files\lightgrep\bin\excelutil.exe*.

### Number of Threads

The script will attempt to auto-detect how many cores you have on your box. Regardless of the number of cores, we generally recommend 3 search threads when using spinning / magnetic hard disks. If you are searching on solid state drives, and have more than 2 cores, you can try increasing the number of threads.

### Log Level

This represents the level of detail provided in the EnCase Console log. Usually this should be set to INFO. Setting this to DEBUG will cause an extremely large number of log events to be written to the Console, and may cause the Console file to grow to tens of gigabytes in size.

### Single File Bookmark Threshold

If you have a large number of search hits, EnCase can slow down significantly. Once the total number of hits exceeds this setting, bookmarks will be created per file instead of per hit. The default value is *3,000,000*.

### Save Settings

Clicking the “Save Settings” button will save all changed options, including Keywords and Entities, to the script’s storage files. After clicking “Save Settings,” the saved settings will be available the next time you run the script even if you cancel the dialog without running a search.

# Starting a Search

Once you click the “Search” button on the main Search tab, the search process will begin. The first thing you will see in the bottom status bar is “Initial filtering” as shown.



The “Initial filtering” process consists of several phases:

* Writing out the selected keyword and entity expressions to the file *keywords.txt*
* Filtering out entries with a physical size of 0 and also files which are completely sparse
* Sorting the entries selected for search so as to maintain consistently high usage of the tool’s multithreading capabilities
* Checking each individual entry to determine if it should be searched using extracted text (transcript)
* Writing the list of entries to be searched to the file *EntryList.txt*

This process can take anywhere from a few seconds to a few minutes, depending on how many entries are selected, and which options you choose.

Clicking the “Cancel” button at any time during the search will cause the entire search to be aborted.

# Monitoring Search Progress

Once the search begins, you can monitor the progress of the search through the main Search tab as shown below. The following fields will be displayed during a search:

* Keyword or Entity name
* Search Expression
* File Count – the number of files which are responsive to each keyword
* Percentage of Files With Hits – of the total files searched, what percentage contained hits for each keyword
* Hit Count – the number of search hits produced by each keyword (before entity filtering)
* Percentage of Total Hits – of the total search hits found for all keywords, what percentage of hits were produced by each keyword



# Reviewing Output

The “OK” button is enabled when the search completes. Clicking “OK” will save sweeping bookmarks containing the search hit results. If “Create Excel Report” was selected, the Excel overview will also be created. Clicking “Cancel” will end the search without saving bookmarks or the Excel report.



## Bookmark Output

Once the bookmarks have saved, you can review them just as you would other search hits. **Error! Reference source not found.** shows example bookmark output from the URLs entity search.



When reviewing search hits, the folders for Entities will contain a comment noting how many hits failed entity filtering, and whether they were discarded.



## Excel Output

The Excel report provides an overview of results by displaying charts of the top ten keywords by number of hits and file size.

|  |  |
| --- | --- |
|  |  |

A keyword frequency report shows all keywords that were searched, and the resulting hit statistics. These results are identical to those in the Search Status box which are displayed during the search.



The DataCategories tab shows all file extensions for search hits, with the number of files and size of files represented by each file extension. Files with no extension are denoted by the asterisk (\*).



The Excel report worksheets are protected so you don’t accidentally alter the data. If you wish to edit the report, simply go to the Review tab in the Office Ribbon as shown below, and click “Unprotect Sheet.” No password is needed.



# Appendix A - Regular Expressions

## Example 1

Crafting proper regular expressions (regex) often means disassembling your desired result into smaller pieces and analyzing what terms should go into your search. Take Internet hostnames, for example, which on the surface have a general structure of “www.mydomain.com”. This seems pretty easy to search for, right? Well, there are a few rules for Internet hostnames, defined in RFC 952 (<http://tools.ietf.org/html/rfc952>) with several supporting documents, and then appended to in RFC 1123 (<http://tools.ietf.org/html/rfc1123>). Let’s break down these rules and see if we can write a regular expression to accommodate them.

1. The maximum length of an Internet Hostname is 255 characters
2. Each label (segment) has a minimum length of 1 character and maximum length of 63 characters
3. Labels are separated by a period (.)
4. The only valid characters are a-z, 0-9, and the minus sign (-)
5. Each label must begin and end with a-z or 0-9
6. The - character is only valid when surrounded by a-z or 0-9
7. The top level domain portion adheres to a list maintained by IANA (<http://data.iana.org/TLD/tlds-alpha-by-domain.txt>), the longest of which is 22 characters
8. Microsoft Windows and Android smart phones, among others, support the use of the underscore (\_) in hostnames. So while not technically valid per RFC, and no email can be routed to them, we should support these in our search as they might be valid on internal networks and they are also sometimes used in special DNS entries that aren’t hostnames.

Let’s take a stab at a regex to search for these rules. We know that each label has to start with a-z or 0-9, and has to be at least 1 character in length. We’re going to use a *character class* to specify the allowed characters. In Lightgrep, left and right square brackets (“[]”) are used to denote a character class. Within them, you can use ranges to specify letters (“a-z”) and *named character classes* to specify numbers or digits (“\d”):

[a-z\d]\.[a-z\d]\.[a-z\d]

There’s a start that gets us strings like “a.b.c” and “1.2.3”, so that’s a good start.

Next let’s take care of the maximum length of each segment, which is 63. We’ll also add in the underscore here:

[a-z\d][a-z\d\_-]{0,62}\.[a-z\d][a-z\d\_-]{0,62}\.[a-z\d][a-z\d\_-]{0,62}

Setting the repetition to {0,62} means that with the required first character we previous specified, we can have anywhere from 0 to 62 more characters in each label section, for a total of 1 to 63. Now we have a regex that will find strings like “www.mydomain.com”, but it will also find “www.mydomain.01234567890-123456789-0123456789”.

By adding in the repetition of {0,62}, we’ve now violated rule #6 above. Let’s change that to make sure the dash isn’t the last character in that string. We can also reduce some more bad hits by looking at the list of allowed top level domains, as defined by IANA. Specifically, the shortest is two characters and the longest is 22 characters. Let’s change our previous regex to allow for this:

[a-z\d]([a-z\d\_-]{0,61}[a-z\d])?\.[a-z\d][a-z\d\_-]{0,62}\.[a-z\d][a-z\d-]{1,22}

We’ve changed the last character class to remove the underscore and changed the repetition to be in line with the minimum of 2 and maximum of 22. Now we won’t get any top level domains that are longer than 22 characters, as we had with the previous version.

One final consideration is that while we will now get hits on “space.mit.edu”, we won’t get hits on “earth.space.mit.edu”, because we’ve only allowed for 3 segments. Let’s remove the middle section and add some grouping and repetition to finish off the regex. We’ll specify the minimum of 3 total segments, and a maximum of 6 segments:

([a-z\d]([a-z\d\_-]{0,61}[a-z\d])?\.){2,5}[a-z\d][a-z\d-]{1,22}

Now we’re getting results like “www.mydomain.com” and “www.ag.state.ny.us” – success!

But wait, we’re still getting hits like “35.51.65.00” and “v3.1.0T” and “routines.js.php” which aren’t hostnames / domain names at all. We could keep making our regex more complicated to cover corner cases, but with Lightgrep, we don’t have to. Check out Appendix 2 – Entity Filters, to see how to further reduce false positives in search results without overcomplicating your regex.

# Appendix B - Entity Filters

## EntityHit Class Reference

### Members

|  |  |  |
| --- | --- | --- |
| Name | Type | Comment |
| FileOffset | ulong | Start location of the hit within the file |
| Length | ulong | Length of the hit, in bytes |
| Encoding | String | Encoding name, e.g. “UTF-16LE” |
| Comment | String | User settable comment field, returned to main program and set as comment on Bookmark |
| HitText() | String | Text of the search hit |
| Entry | EntryClass | The EntryClass object containing the hit |
| File | FileClass | The FileClass object containing the hit |

### Usage Notes

The HitText() property should be called sparingly, as it can incur a performance penalty. Every time this property is used, the File object is accessed and read to retrieve the text of the hit. If your filter will reference the value of HitText() more than once, you should assign its value to a variable at the beginning of your filter code, and then reference that variable:

String hitText = searchHit.HitText();

The Comment property is a String variable that the user can set from within a filter. If a value is set for Comment, this value will persist and be set as the Comment on the hit Bookmark.

## Example 1

Entity Filters provide a great way to cut down on false positives. Sometimes it’s simply not possible to design a regular expression that is both human readable and efficient while finding all the results you desire. Entities allow you to create filters to further pare down the results by using an EnScript filter.

Let’s revisit the regex we created in Appendix 1 – Regular Expressions. You’ll recall this was a regex for Internet Hostnames:

([a-z\d]([a-z\d\_-]{0,61}[a-z\d])?\.){2,5}[a-z\d][a-z\d-]{1,22}

When we left off, we were seeing results like “www.mydomain.com” and “www.ag.state.ny.us”, but also incorrect strings like “35.51.65.00” and “v3.1.0T” and “routines.js.php”. How can we reduce the false positives? Use an Entity Filter! Here’s what a default filter looks like when you create a new Entity:

class MainClass {

bool Main(EntityHit searchHit) {

return true;

}

}

Before we move on, let’s dissect the default filter code and go over a little EnScript housekeeping.

* MainClass is a required class in EnScript, and is included by default. Do not alter the name.
* Main is a required function within MainClass, and it must be of type bool.
* An EntityHit object is passed into the Main function. It is named searchHit by default.
* You can set the Comment variable on the EntityHit object in Main, and the value will be passed back out to the main Lightgrep program. This comment will be set as the search hit Bookmark comment.
* The Main function must return true or false.

Now let’s take a look at how we can build a filter to reduce false positives on our Internet Hostnames regex. One thing we can do is check that the last segment (the “com” in “www.mydomain.com”) of the string is a valid top level domain (TLD). Here’s the code:

class MainClass {

bool Main(EntityHit searchHit) {

String hitText = searchHit.HitText();

if (InternetTLDs.FindName(hitText.GetFileExt())) {

return true;

}

return false;

}

}

searchHit is an EntityHit object, which has a property named HitText(). HitText() returns a string containing the text of the search hit. First, we get the value of HitText() and assign it to a new String variable named hitText. Next, we call GetFileExt() on the hitText string. GetFileExt() is part of the built in String class in EnScript. Essentially it looks for the last period (.) in a string and then returns the text after the period; this is a perfect and easy way to get the TLD portion of our hits. If our hit is “www.mydomain.com” then hitText.GetFileExt() will return “com”.

Next we need to compare this to a list of valid TLDs. Fortunately Lightgrep comes preloaded with a list maintained by IANA. This list is stored in a flat NameListClass named InternetTLDs. NameListClass is a built-in class in EnScript that inherits from NodeClass, which is essentially a tree structure. The easiest way to search for matches is by using the NameListClass function FindName().

When we put all of these pieces together, we end up with the code shown here:

InternetTLDs.FindName(hitText.GetFileExt())

Now we’re effectively comparing the last segment of the search hit to the list of valid top level domains! If our segment matches, we return true on the filter; if there’s no match, we return false.

Once we get our filtered hits back from Lightgrep, we can easily sort on the Bookmark Comments field and separate the validated hits to review first.

It’s noted earlier in the Entity Search section, but it’s worth repeating: Be aware that filtering search hits for entities can take some time, if the filter code is involved and there are lots of hits. The productivity gain from reducing false positives, however, is well worth the slightly extended filtering time.

1. EnCase® is a registered trademark of Guidance Software, Inc. [↑](#footnote-ref-2)