lccFindReplace Manual

Contents

Description

This document describes how to use the lccFindReplace

lccFindReplace
is a Command Line program created to find and possibly replace content in files. Can find parts of lines or multiple lines, and can replace parts of lines or multiple lines.

The program is controlled by a Logic File and/or command line parameters.

All keys can be provided through the Logic File and/or the command line parameters at the same time. See note in the 'lcc:find' and 'lcc:replaceWith' about multi-line exception.

Program Logic

- Reads the Logic File / command line parameters
- Parse each 'lcc:rootPath' location for all Directories and Files
- If 'lcc:fileNameFilter'(s) provided and File Name does not match, skip file
- If 'lcc:fileExtension'(s) provided and File Extension does not match, skip file
- If a 'lcc:replace' is provide, will search for the value 'within' any lines of the files
    - If matches found, will replace content 'within' the line
- If a 'lcc:replace' is provided with 'lcc:multiLineValueStart', will look for entire line(s) that match
    - If matches found, will replace the entire line(s)

The following reports can also be requested/produced:

USE KEY	REPORT DESCRIPTION
lcc:reportFilesMatchedFind	Lists files that contain at least one of the 'lcc:find' values.
lcc:reportFilesValid	Lists files that match all of the filters, ex: extension, name.
lcc:reportFilesValidExtension	Lists files that match one of the file extension filters.
lcc:reportFilesValidNameFilter	Lists files that match one of the file name filters.
lcc:reportDirectoriesValidNameFilter	Lists directories that match one of the directory name filters.
lcc:reportFindsMatched	Lists all files + [tab] + lines from all files that match at least one of the 'lcc:find' values.
lcc:reportFilesValidEncoding	Lists files + [tab] + file encoding that match all of the filters, ex: extension, name.
lcc:reportFilesMatchedFindEncoding	Lists files + [tab] + file encoding that match at least one of the 'lcc:find' values.
lcc:reportFilesValidModifiedDate	Lists files + [tab] + file last modified date that match all of the filters, ex: extension, name.
lcc:reportFilesMatchedFindModifiedDate	Lists files + [tab] + file last modified date that match at least one of the 'lcc:find' values.

Installation

- copy the lccFindReplace.exe to a folder
- create Logic File
- run lccFindReplace.exe

Using

Prerequisites: configure at least one Logic File.

The program will default to using a Logic File called:
lccFindReplace-logic.txt
in the same folder as the program. You can change to use by specifying a different one on the command line.

While running, the program listens for the ESC key to be pressed. If pressed, the program will close all log files and stop running.

Starting

- run lccFindReplace.exe

Starting with an alternate Logic File

- Syntax: lccFindReplace.exe lcc:logicPath [Logic File]
- Example: lccFindReplace.exe lcc:logicPath ourLogic.txt

Logic File Description/Syntax

A Logic File is a Tab delimited text file. Any lines not recognized as a valid Key/Value pair, will be ignore and can be used as remarks/other.

The Logic File uses the syntax.
Syntax: **[Key]** *[tab]* **[Value]** … *[tab]* **[Value]**
Example: lcc:key value

Any extra tabs in a line after the expected ones are considered remarks and will be ignored. This is a nice way to document specific Key settings (see Log Levels in the Logic File example(s) for reference). Also, if you place a tab before a line, that will essentially make it a remark and will be ignored, which makes using/not using logic without removing quicker.

Any line not starting with an expected key is ignored, which makes placing remarks/formatting the logic easy.

Values Upgrade 20190103:
You can now place 'lcc:userInput' into any value to allow the user to manually input any value within the Logic File on execution. This can be used on any column.

Syntax:
Syntax: **[Key]** *[tab]* **[lcc:userInput …prompt…]** … *[tab]* **[lcc:userInput …prompt…]**

The '...prompt...' is what will display to the user to fill-in.

For example, if you wanted to allow a user to provide the 'find' and 'replaceWith' values on execution, you could have lines like:
lcc:find lcc:userInput Find
lcc:replaceWith lcc:userInput Replace With

If you want to Prepend or Postpend a value before/after the user's supplied value, use keys 'lcc:replaceWithUserInputPrepend', 'lcc:replaceWithUserInputPostpend'

Logic File

lcc:debugLevelLevel *(optional)(none to many per Logic File)*

Specifies what log information will be logged/shown. Provide a new line for each level desired.

Debug Levels

1 Logic File Settings: displays settings loaded from Logic File.
2 File Matches Extension: display files that match a requested extension.
3 Traversing File: display each file being traversed.
4 File Size: display each file's size.
5 Source Line (original): display each line from each file.
6 Check If Source Line Matches: display when doing replace check, line # and matches made.

7 Replacing Value: display when replacing a value.
8 Compare Source With Multiline: display when comparing source with multiline.
9 Multiline Match With Source: display when a line is matched from a multiline search.
10 Matched All Multiline: display when all multiline lines match.
11 Replacing Multiline With: display when replacing multiline.
12 Saving Changes – Backup File Exists, Deleting: display when creating backup and backup file already exists, deleting.
13 Saving Changes – Backup Up File: display when backing up file.
14 Saving Changes To File: display when saving file.
15 Matches Found Per File: display how many replacements made per file.
16 Program Finished - Stats: display final stats on all files.
17 Traversing Directory: display when traversing a directory.
18 Logic File Keys/Values: display all of the Logic File keys/values provided.
19 Show Each Match: display when a Find is found.
20 File Name Filter Matches: display when a File name matches a File Name Filter.
21 Valid File: display if a File passes all tests, i.e. extension, file name filter
22 Directory Name Filter Matches: display when a Directory name matches a Directory Name Filter.
23 File Matched Find: display when a File contains a Find.
24 Show Encoding For Valid File: display File Encoding for Valid Files.
26 Directory Name Filter Exclude Matches: display Directory when a Directory contains a Directory Name Filter Exclude
27 File Name Filter Exclude Matches: display File when a File contains a File Name Filter Exclude.
28 Show Encoding For Matched Find File: display File Encoding for Matched Finds.

Syntax: lcc:debugLevel *[tab]* [#]

Example: lcc:debugLevel 2
Example #2 (multiple): lcc:debugLevel 2
lcc:debugLevel 4

lcc:logPath *(optional, but highly recommended)(one per Logic File)*

The path/root name of the Log File. The program will place the '.log' extension automatically and will also append a Year/Month date.

lcc:rootPath *(mandatory)(one to many per Logic File)*

What directory path to start searching for files.

Syntax: lcc:rootPath *[tab]* [Path]
Example: lcc:rootPath F:\folder
Example #2 (multiple): lcc:rootPath F:\folder
lcc:rootPath \\server\share$\folder

lcc:pathExclude *(optional)(one to many per Logic File)*

What directory path to exclude. This is a wildcard, any path with this in it will be excluded.

Syntax: lcc:pathExclude *[tab]* [Path]
Example: lcc:pathExclude \thisfolder
Example #2 (multiple): lcc:pathExclude \thisfolder
lcc:pathExclude someFolder

lcc:subFolders *(optional)(one per Logic File)*

Specifies where sub-folders will be parsed.

Valid Values

- YES

Syntax: lcc:subFolders *[tab]* [YES]
Example: lcc:subFolders YES

lcc:skipBackups *(optional)(one per Logic File)*

By default, the program will make a backup of a file before replacing the content. Providing this key will skip those backups.

Valid Values

- YES

Syntax: lcc:skipBackups *[tab]* [YES]
Example: lcc:skipBackups YES

lcc:saveChanges *(optional)(one per Logic File)*

By default, the program will run in 'simulation' mode, i.e. no real changes made, just information about what it found and would change. Providing this key will force the changes to be made!

Valid Values

- YES

Syntax: lcc:saveChanges *[tab]* [YES]
Example: lcc:saveChanges YES

lcc:includeBOM *(optional)(one per Logic File)*

By default, the program will not save the Byte Order Mark (BOM). Providing this key with 'YES' will include the BOM when overwriting a file.

BOM ref: https://en.wikipedia.org/wiki/Byte_order_mark

Valid Values

- YES

Syntax: lcc:includeBOM *[tab]* [YES]
Example: lcc:includeBOM YES

lcc:targetFilePrePend *(optional)(one per Logic File)*

By default, when replacing content in a file, that file will be overridden. This key overrides that default to prepend this value to all files with content replaced.

You can also provide '[lcc:YYYYMMDD]' anywhere in this value and it will be replaced with the current Year/Month/Day.

Syntax: lcc:targetFilePrePend *[tab]* […]
Example: lcc:targetFilePrePend ourPrePend
Example #2 (with current date): lcc:targetFilePrePend [lcc:YYYYMMDD]-

lcc:addLineNumbers *(optional)(one per Logic File)*

If provided, the program will prepend each line in a modified by with a Line # and [tab].

Valid Values

- YES

Syntax: lcc:addLineNumbers *[tab]* [YES]
Example: lcc:addLineNumbers YES

lcc:backupExtension *(optional)(one per Logic File)*

By default, when backing up files before changing, the program will add an extension to the original filename with '.bak. This key overrides that default to whatever you supply. It is recommended you start your value with a period.

You can also provide '[lcc:YYYYMMDD]' anywhere in this value and it will be replaced with the current Year/Month/Day.

Syntax: lcc:backupExtension *[tab]* […]
Example: lcc:backupExtension .ourExtension
Example #2 (with current date): lcc:backupExtension .[lcc:YYYYMMDD]-ourExtension

lcc:fileNameFilter *(optional)(one to many per Logic File)*

What files names must contain to be used.

Syntax: lcc:fileNameFilter *[tab]* […]
Example: lcc:fileNameFilter partOfAName
Example #2 (multiple): lcc:fileNameFilter partOfAName
lcc:fileNameFilter anotherPart

lcc:fileNameFilterExclude *(optional)(one to many per Logic File)*

What files names will be excluded if the name contains this value.

Syntax: lcc:fileNameFilterExclude *[tab]* […]
Example: lcc:fileNameFilterExclude excludeMe
Example #2 (multiple): lcc:fileNameFilterExclude excludeMe
lcc:fileNameFilterExclude excludeYou

lcc:directoryNameFilter *(optional)(one to many per Logic File)*

What directory names must contain to be used. Sub-Directories will still be traversed, but, Files within ones that do not contain this filter will be skipped.

Syntax: lcc:directoryNameFilter *[tab]* […]
Example: lcc:directoryNameFilter partOfAName

Example #2 (multiple): lcc:directoryNameFilter partOfAName
lcc:directoryNameFilter anotherPart

lcc:directoryNameFilterExclude *(optional)(one to many per Logic File)*

What directory paths will be excluded if the path contains this value.

Syntax: lcc:directoryNameFilterExclude *[tab]* […]
Example: lcc:directoryNameFilterExclude excludeMe
Example #2 (multiple): lcc:directoryNameFilterExclude excludeMe
lcc:directoryNameFilterExclude excludeYou

lcc:fileExtension *(mandatory)(one to many per Logic File)*

What files will be parsed. Only files ending in the extension provided will be audited.

Syntax: lcc:fileExtension *[tab]* […]
Example: lcc:fileExtension .txt
Example #2 (multiple): lcc:fileExtension .txt
lcc:fileExtension .htm

lcc:find *(mandatory)(none to many per Logic File)*

What content to find. The find is case-insensitive, i.e. "Dog", "DOG" and "dog" would find match for 'dog'. However the 'lcc:replaceWith' key 'is' case-sensitive.

There are two options:

- Provide a value to search for. This value can be inside any portion of any line.
- Provide the value 'lcc:multilineValueStart'. Then on the next line(s), provide the lines you want to find. When finished, end with another line of this key, but, with the value 'lcc:multilineValueEnd'.
  Note: the multi-line option is not available in the command line.

Some white characters can be included in the replace value, using the following flags:

- [lcc:CR] : places a Carriage Return character
- [lcc:LF] : places a Line Feed character
- [lcc:tab] : places a Tab character
-
- Example:

- original find value
  dogs and cats live here
- using
  - lcc:find cats
  - lcc:replaceWith [lcc:CR][lcc:LF]cats
- new value
  dogs and
  cats live here

Syntax: lcc:find *[tab]* […]
Example: lcc:find some value we want changed
Example (multiline): lcc:find lcc:multilineValueStart
Some multiline of text, line #1
Some multiline of text, line #2

Some multiline of text, line #3
lcc:find lcc:multilineValueEnd

lcc:replaceWith*(mandatory)(one to many per Logic File)*

What content to replace the content found with 'lcc:find'. This is case-sensitive, i.e. if you said to replace "Dog" with "Cat", "Dog" would change to "Cat", but, "dog" would not.

If you supply a value of: [lccFlag:nothing], this instructs the program to remove the found content, but, not to replace with anything.

There are two options:

- Provide a value to replace with. This value will replace the 'lcc:find' value inside any portion of any line.
- Provide the value 'lcc:multilineValueStart'. Then on the next line(s), provide the lines you want to replace with. When finished, end with another line of this key, but, with the value 'lcc:multilineValueEnd'.
  Note: the multi-line option is not available in the command line.

Some white characters can be included in the replace value, using the following flags:

- [lcc:CR] : places a Carriage Return character
- [lcc:LF] : places a Line Feed character
- [lcc:tab] : places a Tab character

-
- Example:
- original find value
  dogs and cats live here
- using
  - lcc:find cats
  - lcc:replaceWith [lcc:CR][lcc:LF]cats
- new value
  dogs and
  cats live here

You can also provide the following flags as a value:

- **[lccFlag:nothing]** : this flag will cause the program to replace any values found with the paired 'lcc:find' key with nothing, i.e. the value will be deleted, but, not replaced with anything.

Syntax: lcc:replaceWith *[tab]* […]
Example: lcc:replaceWith some new value
Example (multiline): lcc:replaceWith lcc:multilineValueStart
Some new line of text, line 4
Some new line of text, line 5
lcc:replaceWith lcc:multilineValueEnd

lcc:replaceWithUserInputPrepend *(optional)(one to many per Logic File/one per find replace set)*

If you use the 'lcc:userInput' flag to allow user input (ie.. adhoc values), you can use this key to prepend a value to the users supplied valie.

Syntax: lcc:replaceWithUserInputPrepend *[tab]* […]
Example: lcc:replaceWithUserInputPrepend our prepend value

lcc:replaceWithUserInputPostpend *(optional)(one to many per Logic File/one per find replace set)*

If you use the 'lcc:userInput' flag to allow user input (ie.. adhoc values), you can use this key to postpend a value to the users supplied valie.

Syntax: lcc:replaceWithUserInputPostpend *[tab]* […]
Example: lcc:replaceWithUserInputPostpend our postpend value

lcc:replaceLinesWithoutCRLF *(optional)(one per Logic File)*

Normally, the program will create a Carriage Return/Line Feed, or just Line Feed after each line read in. Providing this key will supress those.

Valid Values

- YES

This is handy if you want to remove all Carriage Return and Line Feeds and use the lcc:replaceWith to placed them as you want. For example, if you are parsing a web page that has source code on multiple lines and you want to crunch all the lines together, then make a new line only when it encounter a "<TR" or "/TR>".

This example code would produce that effect:

```
lcc:find                <TR
lcc:replaceWith         [lcc:CR][lcc:LF]<TR
lcc:find                /TR>
lcc:replaceWith         /TR>[lcc:CR][lcc:LF]
lcc:replaceLinesWithoutCRLF         YES
```

Syntax: lcc:replaceLinesWithoutCRLF *[tab]* [YES]
Example: lcc:replaceLinesWithoutCRLF YES

lcc:newLineOnly *(optional)(one per Logic File)*

By default, the program will save each line with a Carriage Return/Line Feed (CRLF), which is the default for Windows files. If you want the lines to end in Line Feed (LF) only, which is the default for Linux files, supply this key.

Valid Values

- YES

Syntax: lcc:newLineOnly *[tab]* [YES]
Example: lcc:newLineOnly YES

lcc:heartbeatDirectoriesTraversed *(optional)(one per Logic File)*

Will report what Directory is being traversed, at every Heartbeat (#) interval. For example, if you provide a value of 5, then every 5th directory will be displayed.

Syntax: lcc:heartbeatDirectoriesTraversed *[tab]* [#]
Example: lcc:heartbeatDirectoriesTraversed 5

lcc:heartbeatDirectoriesTraversedPause *(optional)(one per Logic File)*

If key 'lcc:heartbeatDirectoriesTraversed' supplied, and this is key is provided, the program will pause at each interval, giving you the option to continue or quit.

Valid Values

YES

Syntax: lcc:heartbeatDirectoriesTraversedPause *[tab]* [YES]
Example: lcc:heartbeatDirectoriesTraversedPause YES

lcc:heartbeatFilesMatchedFind *(optional)(one per Logic File)*

Will report what file matches a 'lcc:Find' value at every Heartbeat (#) interval. For example, if you provide a value of 5, then every 5th valid file will be displayed.

Syntax: lcc:heartbeatFilesMatchFind *[tab]* [#]
Example: lcc:heartbeatFilesMatchFind 5

lcc:heartbeatFilesMatchedFindPause *(optional)(one per Logic File)*

If key 'lcc:heartbeatFilesMatchedFind' supplied, and this is key is provided, the program will pause at each interval, giving you the option to continue or quit.

Valid Values

YES

Syntax: lcc:heartbeatFilesMatchedFindPause *[tab]* [YES]
Example: lcc:heartbeatFilesMatchedFindPause YES

lcc:heartbeatFilesValid *(optional)(one per Logic File)*

Will report what file matches all filters, i.e. extension, name filter at every Heartbeat (#) interval. For example, if you provide a value of 5, then every 5th valid file will be displayed.

Syntax: lcc:heartbeatFilesValid *[tab]* [#]

Example: lcc:heartbeatFilesValid 5

lcc:heartbeatFilesValidPause *(optional)(one per Logic File)*

If key 'lcc:heartbeatFilesValid' supplied, and this is key is provided, the program will pause at each interval, giving you the option to continue or quit.

Valid Values

YES

Syntax: lcc:heartbeatFilesValidPause *[tab]* [YES]
Example: lcc:heartbeatFilesValidPause YES

lcc:heartbeatFilesTraversed *(optional)(one per Logic File)*

Will report what file was traversed at every Heartbeat (#) interval. For example, if you provide a value of 5, then every 5th valid file will be displayed.

Syntax: lcc:heartbeatFilesTraversed *[tab]* [#]
Example: lcc:heartbeatFilesTraversed 5

lcc:heartbeatFilesTraversedPause *(optional)(one per Logic File)*

If key 'lcc:heartbeatFilesTraversed' supplied, and this is key is provided, the program will pause at each interval, giving you the option to continue or quit.

Valid Values

YES

Syntax: lcc:heartbeatFilesTraversedPause *[tab]* [YES]
Example: lcc:heartbeatFilesTraversedPause YES

lcc:reportFilesMatchedFind *(optional)(one per Logic File)*

If provided, will create a Report file (text) at the value (path) supplied containing the Path+Filename of each file that contains one of the 'lcc:Find' values.

Syntax: lcc:reportFilesMatchedFind *[tab]* [Path]
Example: lcc:reportFilesMatchedFind report-filesMatchedFind.txt
Example #2: lcc:reportFilesMatchedFind \\server\share$\folder\report-filesMatchedFind.txt

lcc:reportFilesMatchedFindEncoding *(optional)(one per Logic File)*

If provided, will create a Report file (text) at the value (path) supplied containing the Path+Filename of each file that contains one of the 'lcc:Find' values and what that files Encoding is, i.e. UTF8, Unicode, etc.

Syntax: lcc:reportFilesMatchedFindEncoding *[tab]* [Path]
Example: lcc:reportFilesMatchedFindEncoding report-filesMatchedFindEncoding.txt
Example #2: lcc:reportFilesMatchedFindEncoding \\server\share$\folder\report-filesMatchedFindEncoding.txt

lcc:reportFilesMatchedFindModifiedDate *(optional)(one per Logic File)*

If provided, will create a Report file (text) at the value (path) supplied containing the Path+Filename of each file that contains one of the 'lcc:Find' values and that file's last Modified Date.

Syntax: lcc:reportFilesMatchedFindModifiedDate *[tab]* [Path]
Example: lcc:reportFilesMatchedFindModifiedDate report-filesMatchedFindModifiedDate.txt
Example #2:
lcc:reportFilesMatchedFindModifiedDate \\server\share$\folder\report-filesMatchedFindModifiedDate.txt

lcc:reportFilesValid *(optional)(one per Logic File)*

If provided, will create a Report file (text) at the value (path) supplied containing the Path+Filename of each file that is valid (matches all filters, i.e. extension, name filter).

Syntax: lcc:reportFilesValidFind *[tab]* [Path]
Example: lcc:reportFilesValidFind report-filesValid.txt
Example #2: lcc:reportFilesValidFind \\server\share$\folder\report-filesValid.txt

lcc:reportFilesValidEncoding *(optional)(one per Logic File)*

If provided, will create a Report file (text) at the value (path) supplied containing the Path+Filename of each file that is valid (matches all filters, i.e. extension, name filter) and what that files Encoding is, i.e. UTF8, Unicode, etc.

Syntax: lcc:reportFilesValidFindEncoding *[tab]* [Path]
Example: lcc:reportFilesValidFindEncoding report-filesValidEncoding.txt
Example #2: lcc:reportFilesValidFindEncoding \\server\share$\folder\report-filesValidEncoding.txt

lcc:reportFilesValidModifiedDate *(optional)(one per Logic File)*

If provided, will create a Report file (text) at the value (path) supplied containing the Path+Filename of each file that is valid (matches all filters, i.e. extension, name filter) and that file's last Modified Date.

Syntax: lcc:reportFilesValidFindModifiedDate *[tab]* [Path]
Example: lcc:reportFilesValidFindModifiedDate report-filesValidModifiedDate.txt
Example #2: lcc:reportFilesValidFindModifiedDate \\server\share$\folder\report-filesValidModifiedDate.txt

lcc:reportFilesValidExtension *(optional)(one per Logic File)*

If provided, will create a Report file (text) at the value (path) supplied containing the Path+Filename of each file that has a valid 'lcc:fileExtension'.

Syntax: lcc:reportFilesValidExtension *[tab]* [Path]
Example: lcc:reportFilesValidExtension report-filesValidExtension.txt
Example #2: lcc:reportFilesValidExtension \\server\share$\folder\report-filesValidExtension.txt

lcc:reportFilesValidNameFilter *(optional)(one per Logic File)*

If provided, will create a Report file (text) at the value (path) supplied containing the Path+Filename of each file that has a valid 'lcc:fileNameFilter'.

Syntax: lcc:reportFilesValidNameFilter *[tab]* [Path]
Example: lcc:reportFilesValidNameFilter report-filesValidNameFilter.txt
Example #2: lcc:reportFilesValidNameFilter \\server\share$\folder\report-filesValidNameFilter.txt

lcc:reportDirectoriesValidNameFilter *(optional)(one per Logic File)*

If provided, will create a Report file (text) at the value (path) supplied containing the Path of each directory that has a valid 'lcc:directoryNameFilter'.

Syntax: lcc:reportDirectoriesValidNameFilter *[tab]* [Path]
Example: lcc:reportDirectoriesValidNameFilter report-directoriesValidNameFilter.txt
Example #2: lcc:reportDirectoriesValidNameFilter \\server\share$\folder\report-directoriesValidNameFilter.txt

lcc:reportFindMatched *(optional)(one per Logic File)*

If provided, will create a Report file (text) at the value (path) supplied containing the Path+Filename [tab] Source Line of each line in a file that matches a 'lcc:Find' value.

Syntax: lcc:reportFindMatched *[tab]* [Path]

Example: lcc:reportFindMatched report-findMatched.txt
Example #2: lcc:reportFindMatched \\server\share$\folder\report-findMatched.txt

Logic File – Examples

#1: Find Values, Without No Changes

```
lcc:debugLevel      14      Saving changes to file
lcc:debugLevel      15      Matches found per file
lcc:debugLevel      16      Program finished - stats
lcc:debugLevel      19      Show Each Match

lcc:logPath     lccFindReplaceLog

lcc:rootPath    F:\folder\folder2
lcc:rootPath    \\server\share$\folder

lcc:heartbeatDirectoriesTraversed  20
lcc:heartbeatFilesValid         1
lcc:heartbeatFilesTraversed 2
lcc:heartbeatFilesMatchedFind       1
lcc:heartbeatFilesMatchedFindPause          YES

lcc:reportFilesMatchedFind report-filesMatchedFind.txt
lcc:reportFilesValid  report-filesValid.txt
lcc:reportFilesValidExtension         report-filesValidExtension.txt
lcc:reportFilesValidNameFilter        report-filesValidNameFilter.txt
lcc:reportDirectoriesValidNameFilter        report-directoriesValidNameFilter.txt
lcc:reportFindsMatched       report-findsMatched.txt

lcc:subFolders          YES

lcc:fileNameFilter      logic
```

```
lcc:fileExtension       .txt
lcc:fileExtension       .htm


lcc:find        some neat value
lcc:find        another cool thing
```

#2 Find Values, Replace With Values, Save Changes, Make Backups With Current Date First

```
lcc:debugLevel      14      Saving changes to file
lcc:debugLevel      15      Matches found per file
lcc:debugLevel      16      Program finished - stats
lcc:debugLevel      19      Show Each Match

lcc:logPath     lccFindReplaceLog

lcc:rootPath    F:\folder\folder2
lcc:rootPath    \\server\share$\folder

lcc:heartbeatDirectoriesTraversed  20
lcc:heartbeatFilesValid         1
lcc:heartbeatFilesTraversed 2
lcc:heartbeatFilesMatchedFind       1
lcc:heartbeatFilesMatchedFindPause          YES

lcc:reportFilesMatchedFind report-filesMatchedFind.txt
lcc:reportFilesValid  report-filesValid.txt
lcc:reportFilesValidExtension         report-filesValidExtension.txt
lcc:reportFilesValidNameFilter        report-filesValidNameFilter.txt
lcc:reportDirectoriesValidNameFilter        report-directoriesValidNameFilter.txt
lcc:reportFindsMatched        report-findsMatched.txt

lcc:subFolders          YES
```

```
lcc:saveChanges       YES
lcc:backupExtension .[lcc:YYYYMMDD]-lccFRBak

lcc:fileNameFilter      logic

lcc:fileExtension       .txt
lcc:fileExtension       .htm


lcc:find          some neat value
lcc:replaceWith       some neater value

lcc:find          another cool thing
lcc:replaceWith       not so cool anymore
```

#3 Find Values, Replace With Values, Save Changes, Skip Backups

```
lcc:debugLevel        14      Saving changes to file
lcc:debugLevel        15      Matches found per file
lcc:debugLevel        16      Program finished - stats
lcc:debugLevel        19      Show Each Match

lcc:logPath     lccFindReplaceLog

lcc:rootPath    F:\folder\folder2
lcc:rootPath    \\server\share$\folder

lcc:heartbeatDirectoriesTraversed  20
lcc:heartbeatFilesValid       1
lcc:heartbeatFilesTraversed 2
lcc:heartbeatFilesMatchedFind       1
lcc:heartbeatFilesMatchedFindPause        YES

lcc:reportFilesMatchedFind report-filesMatchedFind.txt
lcc:reportFilesValid  report-filesValid.txt
```

```
lcc:reportFilesValidExtension        report-filesValidExtension.txt
lcc:reportFilesValidNameFilter       report-filesValidNameFilter.txt
lcc:reportDirectoriesValidNameFilter       report-directoriesValidNameFilter.txt
lcc:reportFindsMatched       report-findsMatched.txt

lcc:subFolders         YES

lcc:saveChanges      YES
lcc:skipBackups      YES

lcc:fileNameFilter     logic

lcc:fileExtension      .txt
lcc:fileExtension      .htm


lcc:find         some neat value
lcc:replaceWith       some neater value

lcc:find         another cool thing
lcc:replaceWith       not so cool anymore
```

#4 Find Values and MultiLine Values, Replace With Values and Multiline Values, Save Changes, Skip Backups

```
lcc:debugLevel       14      Saving changes to file
lcc:debugLevel       15      Matches found per file
lcc:debugLevel       16      Program finished - stats
lcc:debugLevel       19      Show Each Match

lcc:logPath    lccFindReplaceLog

lcc:rootPath   F:\folder\folder2
lcc:rootPath   \\server\share$\folder
```

```
lcc:heartbeatDirectoriesTraversed  20
lcc:heartbeatFilesValid        1
lcc:heartbeatFilesTraversed 2
lcc:heartbeatFilesMatchedFind       1
lcc:heartbeatFilesMatchedFindPause         YES

lcc:reportFilesMatchedFind report-filesMatchedFind.txt
lcc:reportFilesValid  report-filesValid.txt
lcc:reportFilesValidExtension         report-filesValidExtension.txt
lcc:reportFilesValidNameFilter       report-filesValidNameFilter.txt
lcc:reportDirectoriesValidNameFilter        report-directoriesValidNameFilter.txt
lcc:reportFindsMatched       report-findsMatched.txt

lcc:subFolders         YES

lcc:saveChanges       YES
lcc:skipBackups       YES

lcc:fileNameFilter      logic

lcc:fileExtension       .txt
lcc:fileExtension       .htm


lcc:find          some neat value
lcc:replaceWith        some neater value

lcc:find          lcc:multiLineValueStart
some neat value line #1
some neat value line #2

some neat value line #3
lcc:find          lcc:multiLineValueEnd
lcc:replaceWith        some neater value

lcc:find          another cool thing
```

lcc:replaceWith lcc:multiLineStart
not so cool anymore line #1
not so cool anymore line #2

not so cool anymore line #3
lcc:replaceWith lcc:multiLineEnd

Definitions

BOM - Byte Order Mark, ref: https://en.wikipedia.org/wiki/Byte_order_mark

Modifications

NAME	DATE	MODIFICATION
David Mielcarek	9/21/2016	Created
David Mielcarek	9/22/2016	Added keys 'lcc:targetFilePrePend', 'lcc:addLineNumbers'
David Mielcarek	9/23/2016	Added keys 'lcc:reportFilesValidEncoding', 'lcc:reportFilesMatchedFindEncoding', 'lcc:reportFilesValidModifiedDate', 'lcc:reportFilesMatchedFindModifiedDate'
David Mielcarek	4/11/2017	Added key 'lcc:includeBOM'
David Mielcarek	20180606	Added key ' lcc:replaceLinesWithoutCRLF ' and Flags to lcc:replaceWith
David Mielcarek	20180627	Added key 'lcc:pathExclude'
David Mielcarek	20180801	Added command line parameters option. Added [lccFlag:nothing] to key 'lcc:replaceWith'
David Mielcarek	20181127	Added special character flags to lcc:find
David Mielcarek	20181128	Added '[lccFlag:nothing]' to key lcc:replaceWith

David Mielcarek	20190103	Added 'lcc:userInput' option on values. Added keys 'lcc:replaceWithUserInputPrepend', 'lcc:replaceWithUserInputPostpend'

End of document