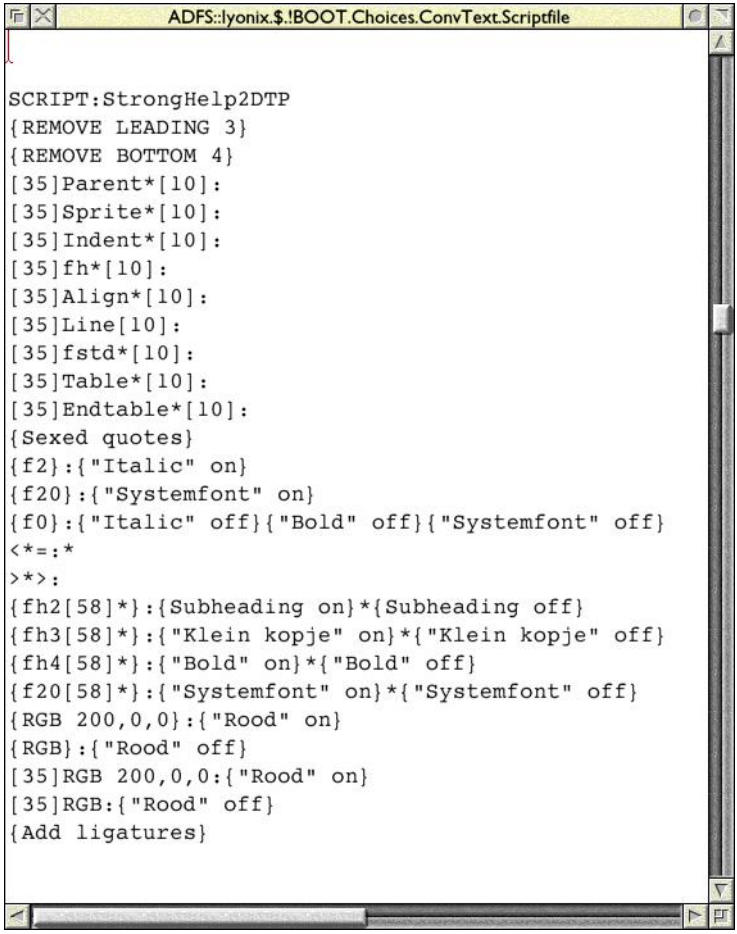
!ConvText

A guide to scripted Search & Replace
for RISC OS

Comments, questions, complaints?
paul@sprie.nl

Check for new versions at:
www.riscos.sprie.nl



The original OvationPro document has been created by first defining some styles and then by dragging the text files from the StrongHelp manual through the !ConvText script above. Apart from adding pictures, only a minor amount of fine tuning was necessary.

General

There are several programs around that let you convert text files in numerous ways. For example, they change *quotes* into so-called *smart quotes*, or they change *line feeds* into *carriage returns*, and so on. None of these, however, will probably cover all of your needs.

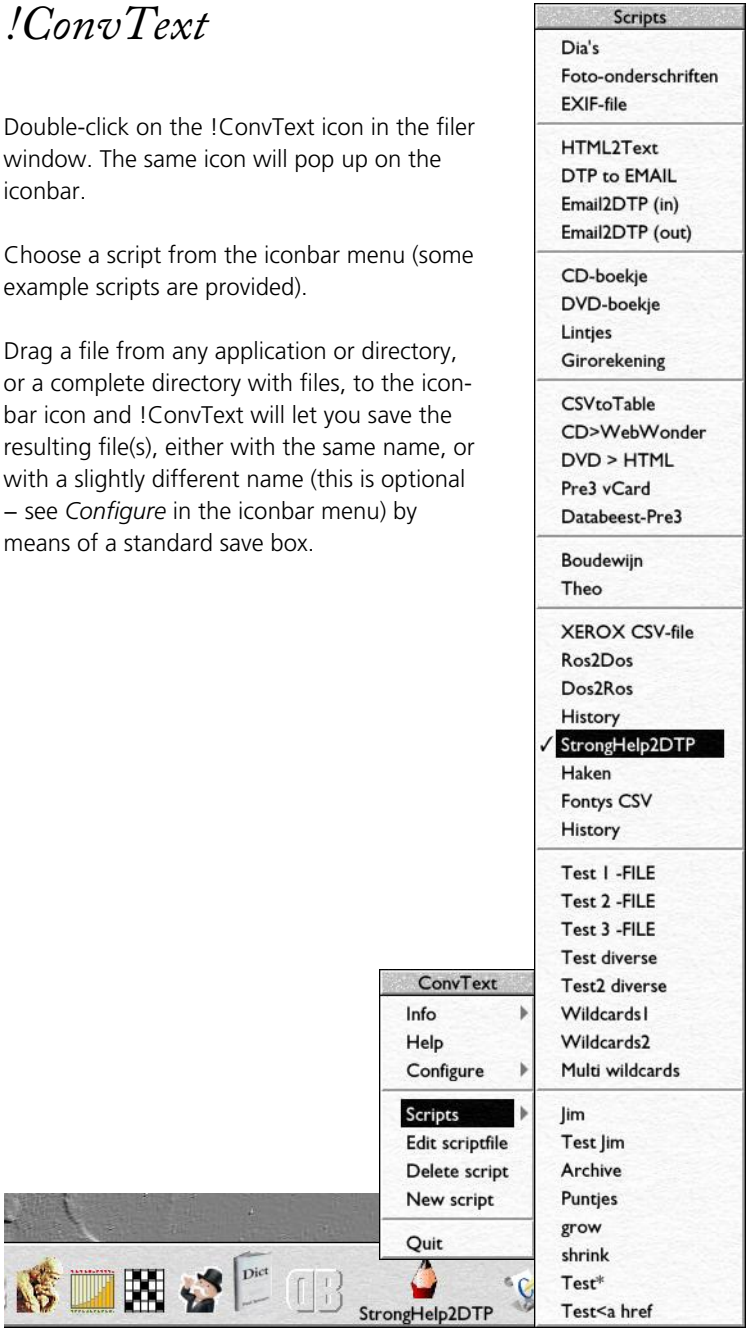
For example, when you select some text from a DTP file in order to send it by email, the mail program will probably prompt you for *Non-ISO-8859-1* characters, such as bullets, smart quotes, ligatures, dashes and so on. Replacing all of these involves a lot of (sometimes complex) find-and-replace actions, especially when this work has to be done on a regular base.

This is where !ConvText comes in. !ConvText is driven by extremely simple scripts that will do all the work in one go. Just choose the appropriate script, drag a file to the iconbar icon and !ConvText will convert that file according to the list of commands in the script. This is almost instantaneous for short files, though it may take some time with very large and complex files. But don't worry, !ConvText runs in the background, so the desktop always remains accessible.

Searching is very straightforward. Apart from the specially defined *'built-in commands'*, processing a search script can be compared to an equal number of find-and-replace actions in, say, !Edit or !Zap.

Running !ConvText

- Double-click on the !ConvText icon in the filer window. The same icon will pop up on the iconbar.
- Choose a script from the iconbar menu (some example scripts are provided).
- Drag a file from any application or directory, or a complete directory with files, to the iconbar icon and !ConvText will let you save the resulting file(s), either with the same name, or with a slightly different name (this is optional – see *Configure* in the iconbar menu) by means of a standard save box.



Creating a Script in !ConvText

This is, of course, the main thing to do. ConvText scripts are kept in a single file, called *Scriptfile*, which is stored in the default Boot:Choices-directory.

You can alter your scripts directly in this central Scriptfile (by choosing *Edit scriptfile* from the iconbar menu), or you can alter them individually in a separate window, which usually is more convenient. The latter is achieved by Adjust-clicking on the iconbar icon. The *Configure* window offers the choice.

The same principle is true for a new script: When you choose *New script* from the iconbar menu, !ConvTexts configurations settings will offer you the possibility to either enter the new script directly into the Scriptfile on a location of your choice, or enter the new script in a separate window. The latter will then be added to the Scriptfile automatically.

Whether you choose to edit the Scriptfile directly or, more conveniently, edit each individual script in its own separate window, you can always insert as many blank lines, and as many comments as you wish, for better legibility.

Syntax

A line may contain either a *title*, a *find-and-replace string*, or a *predefined command*.

Comments

Comments can be placed everywhere in the script. The comment separator is |.

In case of a search for the |-character, type the ASCII value instead: [124]

Take a look at the example script file to see how comments can be included.

Size of a script

!ConvText can handle 30 different scripts by default. Each script may contain up to 100 different lines. This will hopefully cover most people's needs. However, !ConvText can be configured to handle up to 90 scripts with 300 lines each, if necessary.

The next section will explain the creation of scripts in more detail.

Defining a title

- A new script always starts with its title – the one that will appear in the iconbar menu.
- The title is limited to 30 characters and must be preceded by `SCRIPT:` (all capitals – don't forget the colon).
- If *~line* is added to a script title, a line will be drawn after its appearance in the scripts submenu. For example: `SCRIPT:Ligatures~line`.

Find-and-replace strings

General

The syntax of any find-and-replace string is extremely simple:

- First type the *find string*, then type a *colon* (:), and finally type the *replace string*.
  Example:
  `•:-`
  will change every single bullet into a single hyphen.

- Don't put any spaces before or after the colon, unless you have special reasons to exclude or include these spaces in the resulting file.

- If you want to *delete* a string: just leave the space after the colon empty.

- Find-and-replace strings are limited to only one per line.

ASCII numbers

- Rather than typing the actual string, you can also enter the relevant ASCII number. The number must be enclosed by brackets.
  For example:
  `[143]:[45]`
  is identical to
  `•:-`

- ASCII numbers are mainly used for characters such as tabs [9], line feeds [10] or carriage returns [13], that can't easily be typed, or for characters that otherwise aren't visible, such as spaces [32] or hard spaces [160].

Wildcards and special operators

A find-and-replace string may contain wildcards. Both the *single character* wildcard (#) and the *multi character* wildcard (*) are supported. You may use several asterisks and hashes in one string, with a limit of 10 each, making a total of 20 wildcards per string!

A special wildcard is `-EOF` which will delete everything from a given string up to the end of the file.

Converting the #-character, or the *-character, or the -EOF string

In case you want to convert the actual wildcard characters, rather than use them as wildcards, you should use their ASCII values instead:

`[35]` or `[42]` or `[45]EOF`

(If the dash, or any of the characters of the -EOF wildcard, is replaced by its ASCII value, it will no longer be recognised as a wildcard.)

Wildcardish operators

There are three special operators, intended for special cases:

- `-FIRST`
- `-LAST`
- `-FILE`

These operators should be added to the end of a string. Don't forget the leading dash!

- When `-FIRST` is added to a search string, ConvText will process the first occurrence of the found string only, while leaving every following match untouched.

- When `-LAST` is added to a string, ConvText will process the last occurrence of the found string only, while leaving all preceding matches untouched.

(Both the -FIRST and -LAST operators may be used in conjunction with wildcards.)

- When `-FILE` is added to either a search string or a replace string, ConvText will substitute that string by the contents of a given file. The appropriate string should therefore contain a complete filepath, for example

  `Search string:ADFS::Iyonix.$.Convtext.Test-FILE`

CAUTION: When the `-FILE` operator will be used in a search string, things will go wrong because of the two colons after ADFS. They should therefore be replaced by their ASCII values, typically:

`ADFS[58][58]Iyonix.$.Convtext.Test-FILE:replace string`.

Predefined commands

- !ConvText provides the possibility to write your own predefined commands. These are simply to be considered as a call to other scripts.
  Example:
  Suppose, you've written a script, called `Remove ligatures`, which converts every possible ligature to its two (or three) character equivalent. If you want to use this script in another script, then just type `{Remove ligatures}`, rather than copying the entire script into the new one. In this way, you can neatly create your own library of usefull routines.

- Remember that predefined commands always have to be enclosed by braces.

- If you don't want certain scripts to be visible in the menu (e.g. your 'library'-scripts), then put them at the bottom of the script file, right under the line `INVISIBLE SCRIPTS`. (See also the example script file, provided with this program.)

Built-in commands

- !ConvText already provides 19 predefined commands that are built-in (so, the corresponding scripts don't occur in the script file). These commands will hopefully cover some of the most frequently used conversion actions. At least ten of them do conversions that are difficult to implement by simple search-and-replace strings.

- These *built-in commands* are always typed as capitals and must also be enclosed by braces.

- Built-in commands and predefined commands are limited to only one per line.

List of built-in commands

{REMOVE SMART QUOTES}
This will convert all smart quotes into plain quotes.

{SMART QUOTES}
This will intelligently convert all plain quotes into smart quotes, although some additional find-and-replace strings may sometimes be needed (see the example script).

{REMOVE MANY SPACES}
This will convert multiple spaces into just one space.

{REMOVE MANY TABS}
This will convert multiple tabs into just one tab.

{REMOVE MANY NEW LINES}
This will convert multiple line feeds or carriage returns into just two (!) of them, in order to preserve a blank line between paragraphs. In case you want to remove that one as well, you should add the following line to your script: `[10][10]:[10]`

{JOIN BROKEN LINES}
This will join lines that are broken by line feeds, but will leave paragraphs unchanged. This is useful when exporting emails to wordprocessors, for example.

{FLIP DATE}
This will change dates from *dd-mm-yyyy* into *yyyy-mm-dd*.
For example `23-03-1955` will become `1955-03-23`.
(Mainly intended for sorting purposes.)

{SWAP CASE}
This will turn lower case characters to upper case and vice versa

{LOWER CASE}
This will turn all characters to lower case

{UPPER CASE}
This will turn all characters to upper case

{FIRST CAPS}
This will turn all characters to lower case, except for the first character of each word

{EXPOSURETIME}
This will turn the sometimes rather bizarre exposure times from EXIF information (digital cameras) into the more common 1/x format

{ENSURE LEADING BLANK LINE}
This will add a blank line to the top of the file, if there wasn't one already.
This command is extremely useful, when doing S&Rs on the start of new lines.
Without a leading blank line, the first line of every file would otherwise be ignored.

{REMOVE LEADING BLANK LINE}
This is the logical counterpoint of the former command. It will remove the leading blank line, if there is one.

{REMOVE LEADING X}
This could be useful in case of the need to remove a fixed number of lines from the top of the file.
X can be any number between 1 and 100.

{REMOVE BOTTOM X}
This could be useful in case of the need to remove a fixed number of lines from the bottom of the file.
X can be any number between 1 and 100.

{REMOVE TRAILING BLANK LINES}
This command will obviously delete every blank line at the bottom of the file.

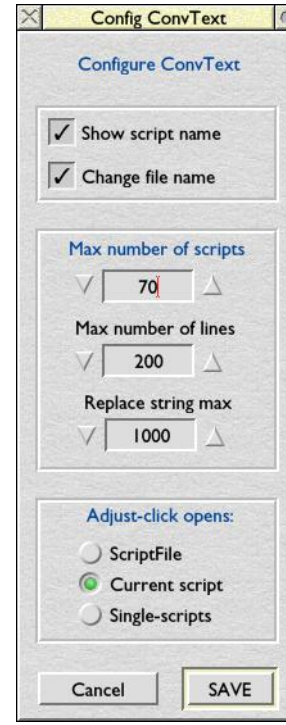
{AVERAGE WORD LENGTH}
Although this command is a built in command indeed, rather than finding and replacing something, it just counts the number of words in a text file and calculates the average word length. It won't change the file and you can't save it afterwards. It's just being informative. It should be used as a single line in a single script only.

{SETTYPE &xxx}
This command doesn't find and replace anything, but sets the file type of the file to be saven. The type should be represented by its corresponding hex number. For example, {SETTYPE &FFF} sets the file type to Text, even if the original file was, say, an OvationPro document. The command can be put anywhere in the script, although the last line is probably the most logic one.

Saving a Script

- Do not forget to save the script again after editing (just press F3).
- Do not save the file elsewhere, or change its name.
- There is no need to start the program again after a script change. !ConvText will read the script at every repeated menu click, and every time that a new text file is dragged to it.

Configuring !ConvText



The configuration window can be accessed from the iconbar menu ('Configure').

There are 6 configurable parameters:

Show script menu

This will toggle between different names under the iconbar icon, either 'Conv' or the name of the chosen script.

Change file name

This will toggle between the same file name or an automatically modified file name of the processed file.

Max number of scripts

The default value is 30. However, when needed, this can be increased up to 90.

Max number of lines

The default value is 100 per script. However, when needed, this can be increased up to 300.

Replace string max

The default size of a replace string that contains wildcards is 1,000 bytes. However, this can be increased in steps of 1,000 bytes, up to a maximum of 100,000 bytes.

Adjust-click opens:

By default, !ConvText opens the Scriptfile when adjust-clicking on the iconbar icon. However, a more convenient way, which is introduced in version 3.10, is opening an individual script in its own window, which avoids endless scrolling in a cluttered Scriptfile. This is done by choosing *Current script*.

Some users may use a solution by Gavin Wraith, which enables single script editing in a more complex way. In that case you should tick *Single-*

scripts, that will eventually open the corresponding directory, rather than the Scriptfile or the script.

CAUTION:

Increasing the maximum number of scripts and script lines will need a larger Wimpslot.

For example, increasing every value to its maximum, will need a wimpslot of at least 1600K. Don't forget to change that in the !Run file, if not already done.

SAVE and RESET

You may save the new options by pressing SAVE. However, when you changed the maximum number of scripts or the maximum number of lines, these changes will only have effect after a restart. !ConvText will prompt you accordingly and quit automatically.

You may also leave the Configuration window without pressing SAVE. Changes, other than the maximim number of lines or scripts, will be applied immediately, but when you start !ConvText again, these options will be reset to their original valuies.

Pressing Cancel will reset every change to its previously saved value.

The next section will present examples and explanations of actual scripts.

DTP to EMAIL Script

```
SCRIPT:DTP to EMAIL
{REMOVE MANY SPACES}
{REMOVE SMART QUOTES}
{REMOVE MANY NEW LINES}
{REMOVE MANY TABS}
[9]:[32]
[160]:[32]
•:-
·:.
[135]:...
-:-
—:-
–:-
fi:fi
fl:fl
[136]:EURO[32]
```

The first four lines are built-in commands, that are explained at the Built-in commands section.

- The fifth line displays: [9]:[32]
  [9] is the ASCII value of the tab character. [32] is the ASCII value of the space character. In the previous line, multiple tabs were converted to just one, so then in this line, all single tabs are converted to a single space.
  The ASCII value of tab is used because there is no other way to type the tab (just like line feeds). The ASCII value of space is used, just to make that space visible in the script.

- The next line displays: [160]:[32]
  This converts hard spaces (ASCII value 160) into normal spaces. Again, ASCII values are used, rather than typing the actual character, just to make them visible in the script.

- The next two lines convert bullets to hyphens and decimal points to full stops. This is necessary because neither of them belong to the internati-

onal ISO standards. Other programs (or mail readers, or HTLM pages) may well display them quite differently.

- The same is true for the characters of the next four lines: [135] is the ASCII value for the triple dot, which will be replaced by three full stops. And the three special dashes will be converted to the normal one (which is a hyphen, actually).

- The next two lines will convert two ligatures to their two-characters equivalents.

- The last line will change the euro-sign (ASCII 136) into the word EURO, followed by a space.
  This ASCII value, however, is not standard. The euro-sign can also be found as ASCII 128. So, it would be quite possible that this line has to be changed to:

  ```
  [128]:EURO[32]
  ```

  or maybe to:

  ```
  [136]:[128]
  ```

  It will just depend of your needs.

EMAIL to DTP Script

```
SCRIPT:EMAIL to DTP
{REMOVE MANY SPACES}
{REMOVE MANY NEW LINES}
{JOIN BROKEN LINES}
{Sexed quotes}
[10]-[32]:[10]•[32]
...:[135]
fi:fi
fl:fl
[32]-[32]:[32][153][32]
[10][32]:[10]
EURO[32]:[136]
>[32]:
```

This script obviously tries to reverse the actions of the previous script, albeit with some alterations.

- The first two lines are self-explanatory.
  The third one, however, may need some comment.
  Often, text that is exported from other programs such as mail readers, will put line feeds at the end of every line, even in the middle of a sentence. Recovering the original paragraphs needs some precise find-and-replace work. However, the built-in command {JOIN BROKEN LINES} will do this automatically.
  N.B. If there isn't a blank line between two paragraphs before you use this script, the two paragraphs will be joined into one.

- The next line {Sexed quotes} is a call to another script, called *Sexed quotes.* You will find this script somewhere at the bottom of the script-file. Since it is placed under the line INVISIBLE SCRIPTS, it won't be visible in the menu. Here it is:

```
{SMART QUOTES}
'm :'m
'n :'n
'r :'r
's :'s
't :'t
```

{SMART QUOTES} is another built-in command, which converts plain quotes into smart quotes, something that requires a lot of work when done manually. It will intelligently put left quotes at the start of a string, and will put right quotes at the end of it, or at any point within the string.

There are, however, some exceptions. In Dutch, for example, some abbreviations exist, resulting in single characters, preceded by a right quote, like 'n or 's. The Smart Quotes command will have put left quotes here. So, some additional commands are needed to put them right again – this is done in the next five lines.

- Now take a look at the next command: This will convert all dashes at the start of a line, followed by a space, into a bullet. Since these dashes only occur at the start of a line, they are presumed to be bullets, originally. Any other dash will remain untouched.

- The next three lines are probably self-explanatory. They will change some character combinations back into their original ligatures.

- The next line displays

  ```
  [32]-[32]:[32][153][32].
  ```

  A dash that is enclosed between two spaces is presumed to be an *n-dash* (ASCII 153) – used in constructions like the one in this sentence.

- The next line ([10][32]:[10]) will remove a space at the start of a new line. This space could be the result of a previous task, that converted tabs to spaces, for example.

- The next line will convert the word EURO, followed by a space, into the original euro sign.
  Once again, ASCII value 136 might have to be changed into 128. It de-

pends on which program the resulting file will be dragged into, or on the font that will be chosen to display the resulting file.

- The last line will just remove so-called Quotation marks that are used in emails. The string after the colon is therefore empty. So, remember that *deleting* a string is just a matter of typing nothing at all after the colon.

Wildcard Examples

Example 1
Let's take a look at the following text:

```
1. Example one
2. Example two
3. Example three
```
etc.

Suppose, we want to change that into:

```
• Example one
• Example two
• Example three
```
etc.

This could be done by means of the following script:

```
[9]#.[32]:[9]•[32]
```

(assuming that every line starts with a tab – [9])

!ConvText will now process every string that has a single character enclosed between a tab and a full stop + space.

If the example text were to go beyond 9 (which means that from then on two digits would have to be changed into one bullet), then we would need a second line in our script, such as:

```
[9]##.[32]:[9]•[32]
```

And a third line would be needed if the example text went beyond 99:

```
[9]###.[32]:[9]•[32]
```

and so on.

Example 2 – wildcard substitution

Let's take a look at a text that contains several lines of the following type (see the accompaning test text Wildcards):

```
Example one costs 10 pounds.
```

Suppose, you want to change such lines into:

```
Example one costs £10.
```

This could be done by means of the following script:

```
costs ## pounds.:costs £##.
```

!ConvText will now process every found text that matches 'costs *(two digits)* pounds', and will automatically substitute the digits that has been found into the replace string.

For any other amount, though, (that is, any amount that uses less or more than 2 digits), a new script line has to be written. The example script Wildcards1 shows them all.

Example 3 – multi character wildcard

In the previous two examples, strings were searched with single character wildcards. For every different occurence, a different script line was needed. The same effect, however, can be achieved more easily with the multi character wild card. Let's go back to the first example. The first three lines of that script could be replaced by just one:

```
[9]*.[32]:[9]•[32]
```

This will change any string between [9] and .[32] into a single bullet.

The same is true for the second example. Those lines could be replaced by just this one:

```
costs * pounds.:costs £*.
```

You may want to try the test text Wildcard, which is included in the archive, to play with the test scripts in the script file.

Warning!
Although using the *-wildcard seems to be a much easier approach, there is, however, a reason why the #-wildcard method might prove more valuable. Because of the simple fact that the *-wildcard will find *any* text that matches the given string, it may well find a match that has a several Kbytes section included! These unpredictable results are more likely to happen as the size of the file that has to be processed grows. Therefore, the *-wildcard should be used with care.

Example 4 – multi character wildcard substitution
Wildcards can be used in much more complex find&replace operations. Consider the following (not necessarily very meaningful) text:

1. Customer John Steed, London, bought article b017, for the amount of 35.00 pound.
2. Customer Mary Shelly, Edinburgh, hired article b121, for the amount of 17.15 pound.
3. Customer Paul Sprangers, Tilburg, stole article b007, for the amount of 0 pound.

etc

Suppose, you want to change all of these lines according to the following concept:

1. Customer (name): John Steed, (city): London, happily bought article C017, for the amount of £35.00

This can be achieved by one single script line, typically:

```
Customer *, *, * article b###, for the amount of
  *pound.:Customer (name): *, (city): *, happily *
  article C###, for f*
```

Note that all 4 occurrences of * and all 3 of # in the find string, neatly represent their matches in the replace string!
The example is quite silly, of course, but hopefully it will demonstrate the power of wildcards in !ConvText.

Example 5 – -EOF

A much more limited use is offered by the third wildcard: -EOF

Suppose, you want to delete the 'sig' of a number of e-mails, that look like this:

Kind regards,
Paul Sprangers

(Bulk of sig information)

Here, the -EOF wildcard will come to rescue. A possible script would be:

```
Kind regards,[10]Paul Sprangers-EOF:
```

This will search for the string 'Kind regards, *(new line)* Paul Sprangers' and will delete everything that follows. This wildcard will deny anything typed after it, including everything which is typed in the replace string.

(The -EOF wildcard is actually quite similar to a * at the end of a search string. In previous versions of !ConvText however, the * was bound to the dreaded 256 bytes limit, wich could be too short for this special purpose. Although a * would do the job in the current version, -EOF is still maintained for backwards compatibility.)

Limitations & Notes

- A find&replace string may not exceed the 256 bytes limit, colon included.

- However, when a *find* string contains one or more *-wildcards, these wildcards may represent an unlimited amount of bytes. For example, the following F&R string:
  `*:`
  will delete any match, typically the entire file, no matter how large it is.

- The maximum size of a *replace* string that contains one or more *-wildcards is limited to 1,000 bytes by default. However, this can be changed in the Configuration window in steps of 1,000 bytes, up to a maximum of 100,000 – although the minimum of 1,000 bytes is probably big enough for the vast majority of tasks.

- Currently, wildcards are limited to 10 each per string – either in the find string or in the replace string. Wildcards in the replace string mirror the matches from their counterparts in the find string (see the wildcard examples for further explanation).

- Currently, !ConvText allows up to 90 different script to be created, containing up to 300 lines each. However, configuring ConvText to use the maximum number of scripts and lines will need a greater WimpSlot. In order to achieve that, open the !ConvText directory by double clicking whilst holding down Shift. Find the !Run file and change its first line to:
  `WimpSlot -min 1600K -max 1600K`

- !ConvText is compiled with the ABCcompiler and therefore needs the ABClib module (usually found in the System directory – look at 310 modules).

- Although !ConvText is developped on an Iyonix running RISC OS 5, it will work on any RISC OS computer from 3.10 onwards, thanks to quite some clever code by Harriet Bazley.

Thanks to

- Paul Beverley of Archive, who not only sent loads of suggestions, but also corrected (and completed) the original help file – a laborious task!
- Eddie Lord, for converting the help file to the StrongHelp format
- Harriet Bazley, for making the program work under RISC OS versions prior to 5.00, and for implementing direct file dragging from and to other applications - actually for a major rewrite of the entire interface!

License

About the author
Paul Sprangers can be contacted at:

E-Mail: paul@sprie.nl
URL: www.riscos.sprie.nl

Version history

2.01

- Support for exposure times in EXIF files added. {EXPOSURETIME}
- recalculates the sometimes bizarre exposure times to a more readable format. However, be sure that no other fractions occur in the file, because they will be converted too.
  Text using a slash, such as Pict/JPG, will be left untouched, though.
- In former versions, trailing linefeeds were always automatically removed, whether intended or not. This function is now disabled.

2.02

- {ENSURE LEADING BLANK LINE} added.
  This adds a leading blank line to the file, which makes certain replace actions, such as searching for the start of every new line, more easily to do for the whole file, including the first line.
- {REMOVE LEADING BLANK LINE} is the logical counterpoint of the former command.
- {REMOVE LEADING x LINES} is a further new possibility:
  not surprisingly, it removes the first x lines of a file (with a maximum of 100).
- {REMOVE BOTTOM x LINES} does the same as the former, but then for the bottom lines, obviously.
- New wildcard added: -EOF, which will delete everything from a given string up to the end of the file. See the accompanying StrongHelp file for an example.
- Bug fix concerning ASCII zero characters.
- Little bug fix concerning trailing character at in-memory transfer.
- {REMOVE TRAILING BLANK LINES} added, which use is probably obvious.

2.03

- New wildcard added: -FIRST will only process the first occurrence of a given string. Syntax: search string-FIRST:replace string
- New wildcard added: -LAST will only process the last occurrence of a given string. Syntax: search string-LAST:replace string
- New operater added: -FILE will substitute the string in which the operator is found, by the contents of a given file. Syntax: search string:filepath -FILE Or: filepath-FILE:replace string

2.04

- New configuration window, accessible from the iconbar menu. Options extended to a configurable number of scripts (30-90) and a configurable number of lines per script (100-300).
  CAUTION: Increasing the maximums may need a larger wimp slot.
  You should change that in the !Run file. Reminder: 90 scripts of 300 lines each needs a wimp slot of 1,600K
- Website button added to Info window.
- Slightly modified Stronghelp manual.
- 2.04d: bug fix in read script routine.
- 2.04f: bug fix in interpreting [xxx] strings (as opposed to ASCII values, such as [65].
- Processing directories now also works when the first entry in a directory is another directory.
- Iconbar will be properly updateted after choosing a new script
- Adding ~line to a script name will cause a line to drawn after its appearance in the scripts submenu. For example: SCRIPT:Ligature~line
- Iconbar text will now remove the suffix ~line from a script title.

- If the last two characters of a file are [10][32], ConvText will remove the trailing space before processing, since in very rare circumstances this space will cause corruption (mentioned by Jim Nagel).
- An attempt is made to enable file saving, even after an error warning, in order to save as much of the processed text.

3.00
- Entire new core.
- Much better wildcard implementation.
- Asterisks (*) in search strings may now represent an unlimited amount of bytes.
- Up to 10 asterisks (*) and 10 hashes (#) may be used per string
- also in replace strings! Wildcards in the latter will obediently paste their corresponding matches from the search string into the replace string.
- The length of a replace string with wildcards is limited to 1000 bytes by default. However, this can be changed in the Configuration window in steps of 1000 bytes, up to a maximum of 100,000.
- The -FIRST and -LAST operators may now be used in conjunction with wildcards too.
- The -EOF wildcard is kept for backwards compatibility, although a * would now achieve the same.
- The original backslash, used for searching for * and \# is now abandoned. Use their ASCII values instead: [42] and [35].
- A more accurate count of changes during conversion.
- Much better memory management and garbage control.
- Dynamical memory slot.
- In spite of all the improvements, !Convtext is now about 35% faster!
- Optimised {REMOVE MANY}-routine
- Much better multi-tasking, at a small speed penalty (still 35% faster)

3.01
- Tiny, little bonus added: {AVERAGE WORD LENGTH} It's a built in command, but rather than finding and replacing something, it counts the number of words in a text file and calculates the average word length. Not important, but possibly funny.
  It should be used as a single line in a single script. It won't let you save the resulting file, even though it removes subsequent spaces for a proper calculation. It's just being informative.
- Much more accurate and much faster {AVERAGE WORD LENGTH}.
  Contrary to the previous version, all sorts of characters such as **,./$%*()** and many more, are excluded from the calculation, while the calculation is now nearly instantaneous.
  Currently, the word count and therefore the calculated average word length is much more accurate than e.g. in !Zap or !OvationPro!

3.02
- Nasty bug (reported by D. Rooke) removed. Actually, it was more than a bug. What was considered to be an elegant rewrite of the core, appeared to be a complete mess! The core assumed a string after a wildcard to be found when only the first character matched!
  This is now sorted out I hope.

3.03
- New built-in command: {SETTYPE &XXX} This will change the filetype according to the entered hexadecimal number. The command can be put everywhere in the script, although the last line is probably the most logic one.

3.04
- Second attempt to fix the mess that was introduced with the rewrite of the wildcard section. Single wildcard occurrences seem to work now.
  Multi wildcard occurrences apparently need some extra work, though.
- Select-clicking on the iconbar icon now opens the Script Menu, rather than a reminder that you have to use the iconbar menu for that.
- Adjust-clicking on the iconbar icon now opens the Scriptfile. In a future version I'd like to make this more or less configurable.

- The wildcard section finally seems to work now!
  After debugging for hours, making heavy use of Martin Avisons priceless !Reporter, only a small number of adjustments were actually needed. No doubt, however, that soon new problems will be reported.

3.05

- Configuration window is extended. When choosing Edit Scripts (or Adjust-clicking on the iconbar icon) one can choose between either the Scriptfile indeed (default), or the Single-scripts directory that is introduced by Gavin Wraith, after an idea of Jim Nagel.
- Little counter bug removed, that made the program stop searching just before the last byte in the file.
- Although a click on the iconbar invokes the Scripts menu indeed, it appeared that choosing anything was not possible.
  This is now fixed.
- Little bug concerning the end -block-counter removed.
- If a * or a # is used more times in the replace string than in the search string, then the new * or the new # will have the contents of the previous ones. For example: when feeding the word 'test' to
  t#st:w#st, r#st
  the result will be: west, rest.
- Little bug concerning tex-counter removed.
- PROCfind is now better aligned with PROCfind_wild carded.

3.06 -- UNRELEASED --

- Scripts-directory added, which makes individual editing possible.
- If ConvText finds no Scripts directory, it wil automatically create one, based upon the Scriptfile. All individual scripts will receive an incremental prefix, while characters that are not allowed in file names will be replaced. The user will be notified of every change.
  Unlike Gavin Wraiths approach (Single-scripts), ConvText reads its information directly from the individual files. There's no need for manual regeneration of the Scriptfile after each change.
- This idea is now abandoned, in favour of a better one (See 3.10).

3.10

- ConvText finally supports individual script editing. When you choose the corresponding option in the Configure window ("Adjust-clicking opens: Current script"), an adjust click on the iconbar icon will then open the current script in its own window.
- If something goes wrong (in spite of a seemingly foolproof routine), there will always be a copy of the previous Scriptfile in the <Choices>-directory. Look for the file named: Script-old.
- The 'current script' approach is not very compatible with the 'single-scripts' solution of Gavin Wraith, since modifications will be written directly to the Scriptfile.
- The iconbar menu now shows 'Edit Scriptfile' rather than 'Edit script'. Editing individual scripts can only be achieved by adjust- clicking on the iconbar icon. The Scriptfile, however, remains accessible from the menu, for example for creating a new script.
- Configure now works more style guide compliant. Changes will be applied immediately, but they will only be permanent after a click on SAVE
- Little typo corrected, that only occurred when there are no invisible scripts.

3.11 -- UNRELEASED --

- !ConvText offers the choice between a Scriptfile that contains all the scripts (original approach), or a Scripts directory, that contains the individual scripts. Changes made to the Scripts directory are automatically mirrored to the Scriptfile. When there's no Scripts directory (or when you delete it), !ConvText will create one immediately, based upon the scripts as found in the Scriptfile.
- A new script can be created from the iconbar menu. A text file will be opened with the line 'SCRIPT:' already there.
- Scripts can be deleted from the iconbar menu. The only

need for opening the Scriptfile or the Scripts directory is for editing the invisible scripts.
- Since the Scripts directory doesn't offer any real advantage over the Scriptfile, the whole idea is abandoned – with pain in the heart…

3.12
- 'New script' and 'Delete script' entries are added to the iconbar menu. Unless you want to edit so-called Invisible Scripts, there's no longer any need for opening the Scriptfile.
- Iconbar text will now be properly updated after creating, or deleting a script.
- Various optimalisations in the interface code.
- !ConvText now really quits when it needs to, in order to make newly chosen values active.
- The scripts menu, when invoked with a click on the iconbar icon, is now placed properly.