SilverAge Software, Inc.

Text Workbench

User Guide

Version 6.x

CONTENTS

GENERAL INFORMATION

About Text Workbench

Text Workbench is an assistant simple to use tool to help you process text in a number of files with a single click.

Text Workbench supports the following file types.

- **ANSI text files**: SGML documents (HTML, XML, etc.), source code and any other plain text files.
- **Unicode files**: standard Unicode files;
- **UTF-8 files**: files with multibyte symbols according to the UTF-8 specification;
- **Microsoft® Office™** files: **Microsoft Word** documents (.doc, .dot, .rtf) and **Microsoft Excel** documents (.xls). This feature requires that these components of the Microsoft Office are installed on your computer.

Major advantages

- Works really fast, even with regular expressions! Faster on ANSI files, and also fast on Unicode and UTF-8 files.
- Seamless embedded text editor with syntax highlighting for HTML, CSS, JS, Perl and C++.
- Built-in **file rename** facility.
- Supports regular expressions with extended syntax.
- Supports various command line parameters.
- Comprehensive user interface.
- Clear diagnostic messages.
- Keeps your files safe as the process of replacement is undertaken in memory, not in files.
- Multiple user interface languages.

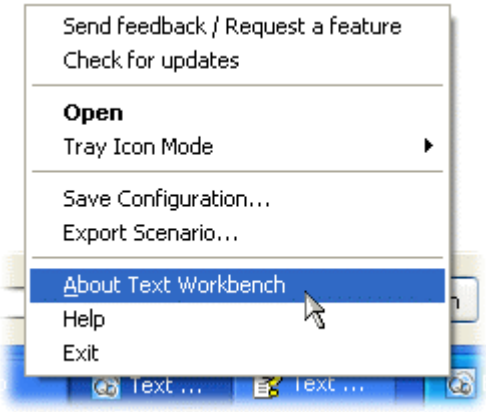
Common tasks

- Perform simple file search
- Search for text in files
- Search for text in files in multiple folders
- Search for text using regular expressions
- Find and replace text in files
- Find and replace text in Microsoft Word and Microsoft Excel files
- Search and replace text using regular expressions
- How to collect any text from multiple files in a single file
- Create a CSV file with e-mails collected from multiple files

- How to insert a file reference regarding directory hierarchy
- How to setup a start-off to documenting your source code
- How to use scenarios to automate work
- How to use command line

Checking for updates

You can easily check for product updates by right-clicking the application icon in the system tray and selecting command **Check for updates**. Please ensure that you are connected to the Internet before doing so.



GETTING STARTED WITH TEXT WORKBENCH

This is the short description of operating the **Text Workbench**. After you have read it, you can start working with the software. If you need more information on any particular feature, please read the corresponding topics. Many frequently used operations are described in the **Common Tasks** section.

1. Set the path to the **folder** containing the files to process. You can click Folders to show the folder tree in which you can easily select the required folder.
2. Check the **Recur subfolders** box if you want to change files in the recurred nested subfolders of the folder you have specified.
3. Set the one or more **file types** (masks).

   Examples:
   `*.txt`
   `*.txt;*.html`
   `prod*.html;prod*.asp`

4. If you need to use the regular expressions, check the **Regular expressions** box.
5. Type in the text to search for (**Find What** field).

   (Optional) If your text contains multiple lines, you can click button to open the **Multiline Editor** to enter the text.
6. Type in the text to replace the found text (**Replace with** field). You can leave this field empty to delete the found text.
7. *(Optional step)* **Additional search criteria** can be set if needed:

   Date Tab

   1. Specify whether change all files or those having the definite date only. You can set the range implicitly, or in days from the current day, or in months from the current month. The day range includes the first and the last day.

   Properties Tab

   2. Specify whether change all files or those having the definite size only. When setting the size, please note that when kilobytes are chosen as the measurement unit, this value is multiplied by 1024.
   3. Specify should the application process read-only files. If needed, you may choose to set this attribute back after the file is processed.

4. Click **Search** to just search for files, or **Replace** to replace. The **Search only** box forces the search mode even if **Replace** is clicked.

The search and/or replace process is now starting. You can always interrupt it by pressing **Stop**.

FILE NATURE DETERMINATION: HOW THE PREPROCESSOR WORKS

Text Workbench is enabled to process files of many different formats: ANSI text files, Unicode files, UTF-8 files, Microsoft Word and Microsoft Excel documents. Besides, it can even find and display image files of the most common formats (GIF, JPEG, PNG, BMP, TIFF, Windows metafiles and icons).

Such flexibility cannot be achieved without having to preliminary preprocess files. All files that Text Workbench software encounters while traversing the specified folder(s), are first assorted and then a set of statistical and deterministic algorithms are applied to them.

The sequence below illustrates the preprocessing steps that Text Workbench undertakes before starting the actual file search and/or replace operation.

1. The finder encounters a new file - say, named `file.ext`.
2. A set of tests is applied to the file to determine whether it can be processed.
    1. If the file matches at least one of the supplied file search masks, the system continues processing the file.
    2. If the file matches at least one of the supplied file exclude masks, the system skips the file.
    3. The system checks the file timestamp. If any date and time restriction is set on the Date Tab, and the current file *does not* match the date and time criteria, the programme skips the file.
    4. The file read-only attribute is verified. If the Properties Tab settings does not allow processing read-only files, and the current file is read-only, the system skips the file.
    5. If the file size restriction is set on the Properties Tab, the system matches the current file size against the file size restriction criteria. If the current file does not match, the system skips the file.
    6. If a user specifies to create file back-ups on the Properties Tab, and the actual replace operation is performed (as opposed to the simple search operation), the system checks the current file extension. If it is the same as the one specified to be used to create back-up files (on the Properties Tab), the system skips the file.
    7. If the current file resides in a folder with the name same as specified to be used for back-up folders on the Storage Folders Tab, and the actual replace (not just the search) operation is performed, the system skips the file.
8. If all above tests are passed, and the option **Replace text in file names, not in file content (rename files)** is *not* set, the actual preprocessing is performed.
    9. The system checks the file extension.

- If it is **.doc(x)**, **.dot(x)** or **.rtf**, the file is a **Microsoft Word** document.
- If the extension is **.xls(x)**, the file is a **Microsoft Excel** document.
- If the extension is **.bmp**, **.gif**, **.png**, **.ico**, **.jpg**, **.jpeg**, **.tif** or **.wmf**, the file is an image.
- If the extension pertains to file types which are binary a priori (**.exe**, **.dll**, **.vxd**, **.obj** etc.), the file is marked as binary.

10. The system suspects files to be non-ANSI a priori. This is why the system checks some first bytes of a file (in fact, the BOM - Unicode byte order mark - is verified):

    - If the first three bytes are EF, BB and BF, the file is a UTF-8 file.
    - If the first two bytes are FE and FF, the file is a Unicode file (Little Endian, which is the Windows standard).
    - If the first two bytes are FF and FE, the file is a Unicode Big Endian file.
    - If the first two bytes define file as of some other exotic Unicode format (UCS-32), the file nature is set as binary.

11. If the option **Analyse input and exclude binary and image files** is *not* set (Options Dialog - Processing), the system handles the current file as an ANSI file. The system determines the real file nature only if a user attempts to view the file.
12. Otherwise (if the option **Analyse input and exclude binary and image files** *is* set), the programme applies the statistical analysis to the file content. If any byte of the file is below 0x20 and is not any printable symbol except CR, LF and the tab, this byte is counted as the one that may indicate a binary file. If the percentage of suspect bytes is above the value specified in the Options Dialog - Processing, the file is marked as binary.

    On the other hand, if the deterministic analysis of the byte stream indicates that the found sequences (lead bits followed by encoded symbols) inhere in a **UTF-8** file, the current file nature is set to **UTF-8** even if the file does have **BOM** at the start.

13. If the option **Replace text in file names, not in file content (rename files)** *is* set, Text Workbench tries to rename the found files. If a user has clicked **Replace**, the actual renaming take place. If a user has clicked **Search**, Text Workbench suggests and displays new file names.

    If this option is unchecked, **Text Workbench** processes (searches or replaces) the file contents.

Chapter 1.
User Interface

Main Window Tabs

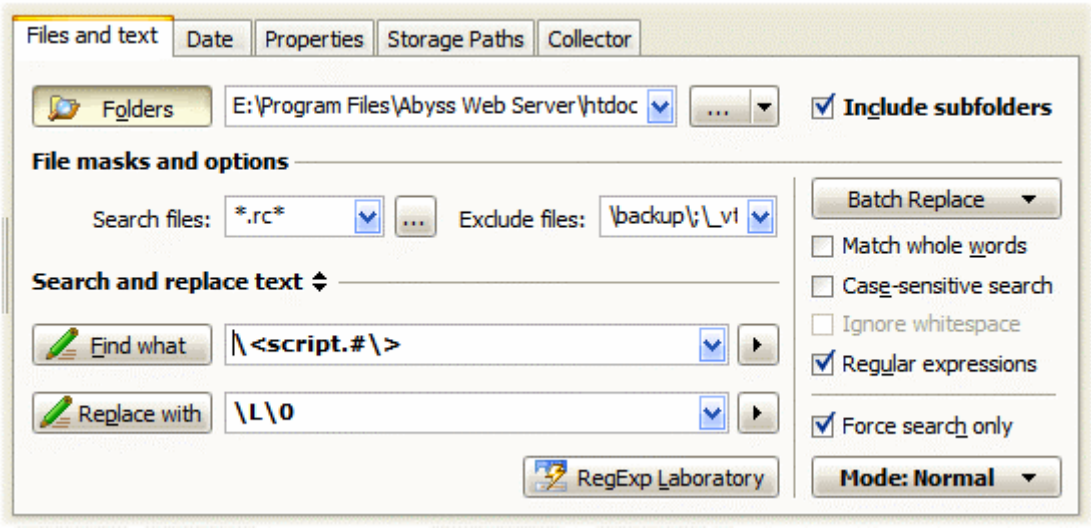
FILES AND TEXT TAB

The **Files and Text** tab is the main user interface screen with which you set the search and replace process parameters.

This page can have two visual contexts - common and advanced.

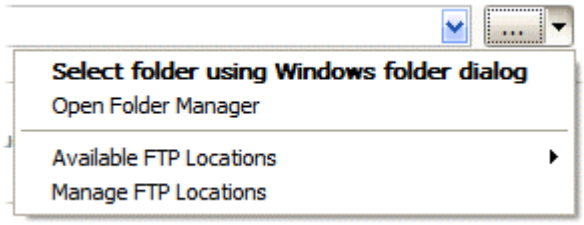
Common Context

The *common* visual context is active by default and is most commonly used. The illustration below shows how the tab looks in this mode.



The table below describes the tab fields.

Field	Description
Folders (*drop-down list and text field*)	Displays one or more folders that Text Workbench will scan for files. You can specify here multiple folders separated with the following symbols: semicolon (**;**) or **|**. The path that you specify here may contain not only canonical path. You can include system environment variables here. For example, the following path is possible: `%WINDIR%\TEMP\`
Folders (button)	Clicking this button shows or hides a folder tree in which you can select folders by checking the appropriate boxes. If you select more that one folder, these folders are all included in the ***Folders text field*** separated by semicolon.

Folder Functions button 	The left part of this split button ("**...**") opens a standard Windows folder selection dialog which you may encounter all over the place in other applications and Windows itself. The split arrow, when clicked, opens the following menu:  in which: ▪ **Select folder using Windows folder dialog** command speaks for itself and acts like the left part of this split button ("**...**"); ▪ **Open Folder Manager** opens the Folder Manager to define and select the named folder sets; ▪ **Manage FTP Locations** opens the FTP Locations dialog box in which you can define the named FTP accounts for your folders on remote FTP servers; ▪ **Available FTP Locations**: this submenu is available only if you have previously created one or more FTP account (location) using the **Manage FTP Locations** command.
Include subfolders	With this option checked, Text Workbench will scan all the nested subfolders of the specified search folder(s) recursively.
File masks and options	
Search files	One or more semicolon-separated masks to match against files in folder(s) to search. The two types of masks are allowed. Wildcard Mask Wildcard mask allows using asterisk to match one or more symbols. For example, the following mask set will match `htm`, `html` and `aps` files: `*.htm;*.html;*.asp` Regular Expression Mask A mask is considered a *Regular Expression Mask* if it contains one of the following symbols: **?** **|** **** **/** **<** **>**. A regular expression mask is just a regular expression that is matched against a file name. For example, the following mask will match all files names beginning with **get**, having a digit in the end and an extension **.html**: `get.+\d\.html`
File selection button 	If you want to find and replace text in only one file rather than in all files in a folder, you can click this button to select the desired file.

	Selecting a file sets the **Folders** field to the path to the selected file, and the **Search files** field to the file name.
Exclude files	One or more semicolon-separated masks of files that should not be processed. If a mask starts with backslash ("****"), the mask is used as a *path mask* and indicates folders that should be skipped. For example, is the mask is `\css`, folders like `c:\web\css\` or `c:\site\css_files\new\` will not be processed. **Note!** This field allows only **Wildcard Masks** (see above). The **Regular Expression Masks** are not recognized.
Search and replace text	
Find What button	Click this button to open the **Multiline Editor** dialog. This dialog allows entering text with multiple lines in normal mode, converting line breaks to `\r` and `\n` as appropriate.
Find What text field	This field is used to enter the text to search for. You can click the drop-down button to select the previously specified text. The recent text list can contain up to 15 lines. You can leave this field empty to only search for *all* files matching the supplied *mask* in the *search folder(s)*.
Replace With button	Same purpose as for **Find What** button, but applies to the replacement text.
Replace With text field	Same purpose as for **Find What** text field, but applies to the replacement text. **Note** Generally, it is not necessary to escape non-alphabetical symbols except backslash (****) in the **Replace With** field if replacing with the Regular Expressions, but doing so is *highly* recommended to avoid collisions.
Swap Text mini button	If clicked, text from the **Find What** field is placed in the **Replace With** field. Text from the **Replace With** field is placed in the **Find What** field.
Control Escape buttons	If you have the option **Regular Expressions** selected, shows a menu with regular expressions that you can select and insert. If the **Regular Expressions** option is not selected, shows a menu with common control escapes.
Regular Expression Laboratory button	Opens the Regular Expression Laboratory window, where you can test your regular expressions.
Options Group	

Batch Replace button	Clicking this button invokes a menu containing the **Scenario** options. If you select a scenario or open the Scenario Manager dialog and activate a scenario, the **Files and Text** tab switches to the **Advanced** mode (see below).
Match whole words	If checked, Text Workbench only finds instances of the **Find What** string that are matched in complete words. For example, a search for "*Plan*" will return matches like "*Plan*", "*Plan.*", "*Plan,*", "*Plan* " but not "*Planning*" or "*Planned*".
Case-sensitive search	Checking this button allows restrict your searches to those phrases that match the sought text exactly. If using Regular Expressions, you can control case sensitivity with the regular expression switches(`\L`, `\U`, `\c` and `\C`). See more information on Regular Expressions.
Ignore whitespace	If checked, Text Workbench will treat all the occurrences of space and tab characters as if there are none. For example, a search for "*Plan for today*" will return matches like "*Plan for today*", "*Plan fortoday*", "*Plan for today*", "*Planfortoday*". This option is effectively a shortcut to **Ignore blanks** and **Ignore tabs** options in the Options Dialog - Processing page of the application options sheet.
Regular expressions	Check this button to allow searches with Regular Expressions.
Force search only	This option overrides the behaviour of the **Replace** button. If checked, no replacements will occur even if the sought text is found. Use this options as a failsafe if the **Replace** button is of rare use for you.
Application mode button Mode: Normal	Shows a menu in which you can select the processing mode. Currently, there are two modes available: ▪ **Normal** mode: sets Text Workbench in the most commonly used text search and replace mode to find and/or replace text in files; ▪ **Rename** mode: brings the program to the **file rename mode** in which Text Workbench does not search the file contents. Instead, it searches and/or replaces text in the file names.

Notes on Rename mode

In **Rename** mode, the following rules apply.

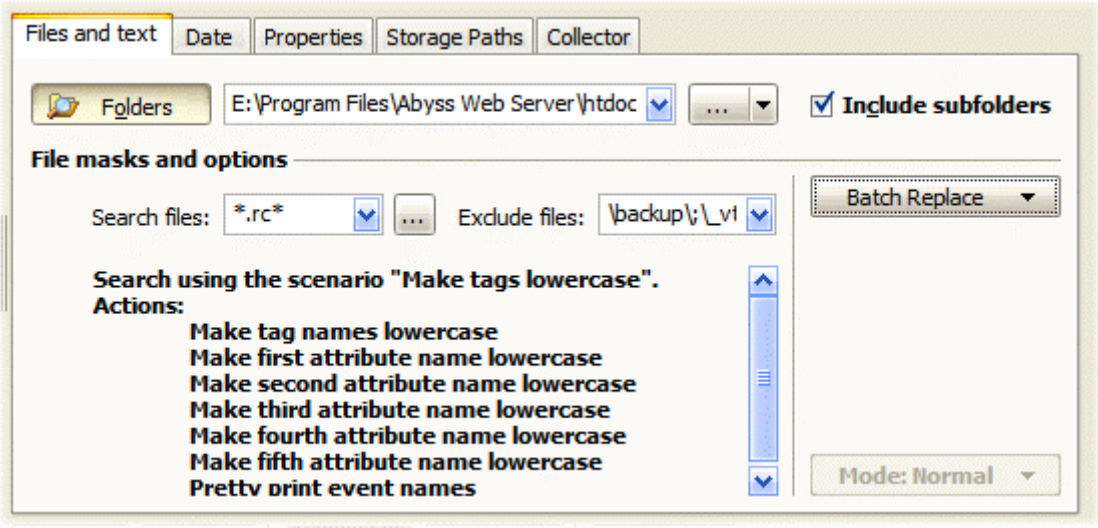
1. If you click **Search**, or if the **Search only** box is checked, the program searches file names and displays new possible names of the matching files.
2. If you click **Replace** and the **Search only** box is **not** checked, the program searches file names and renames the matching files.

> **Note!**
> Rename mode cannot be used with scenarios.

By default, the folder names are not affected. You can rename folders by running **Text Workbench** with the `/renamefolders` option on the command line. Folders can be renamed if only the **Recur subfolders** option is off.s

Advanced Context

The Advanced visual context is active in case any search Scenario is active. In this mode, all standard text options are disabled and the corresponding controls hidden. The scenario name is displayed instead:



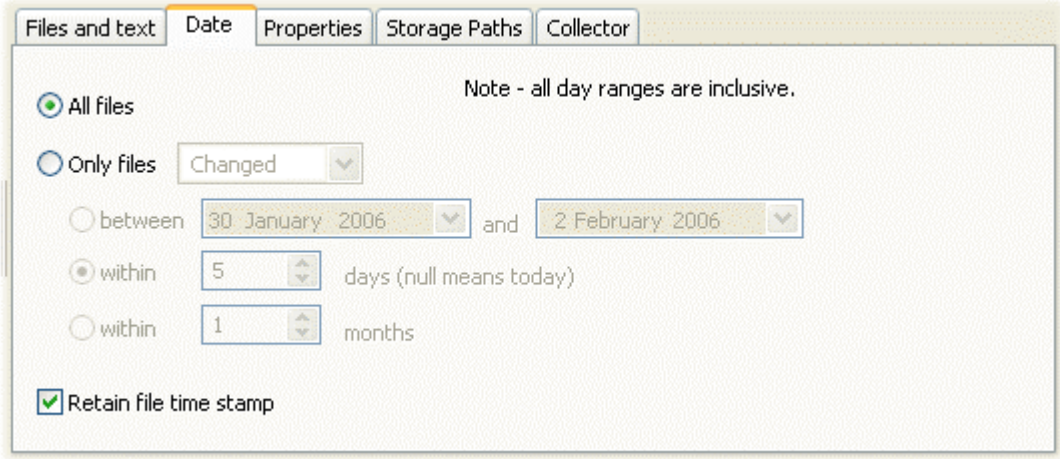
The only text control that remains on the screen is **Replace text in file names, not in file content (rename files)**. If you check this box, the active scenario is switched off, because the **Renamer** cannot be used with scenarios.

See Also

Scenario Editor Dialog

DATE TAB

This tab allows you to specify the date or range of dates that the matching files should have been created, opened or changed.

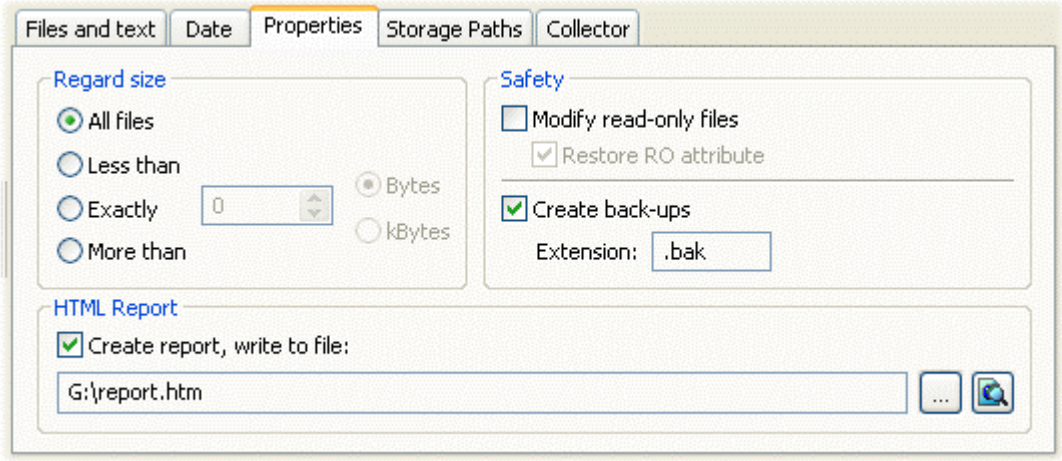


The table below describes the tab fields.

Field	Description
All files	Default selection. Denotes that all found files will match, disregarding time stamp.
Only files	Enables the drop-down list allowing to select the type of date to match: ▪ Changed ▪ Created ▪ Opened After you have selected the desired date type, you should select the date using the following control groups.
between dates	Select this option if you know the exact dates of file creation and want to find them. Choose dates from the drop-down calendars. The search includes files that match the starting and ending dates as well.
within days	Select this option if you want to find files changed, created or opened within the specified number of days. Set the number of days to 0 (null) to find files with today's file stamp.
within months	Select this option if you want to find files changed, created or opened within the specified number of months.
Retain time stamp	Tells to leave the times of the *modified* file as-is, without change.

PROPERTIES TAB

This page contains various options to control file matching and perform user-defined operations.



The tables below describe the tab fields.

Regard Size Group

Field	Description
All files	Default selection. Denotes that all found files will match, disregarding the size.
Less than	Denotes that files sized less than specified value will only match.
Exactly	Denotes that files with the specified size will only match.
More than	Denotes that files with the size more than specified value will only match.
Bytes or **kBytes**	Select the units of file size measure. To precisely set the size, select **Bytes**. **Note** Please keep in mind that *1 kByte* equals *1024 bytes*.

Safety Group

Field	Description
Modify read-only files	By default, this option is disabled. If checked, the TextWorkbench will attempt to modify files with the R/O attribute.
Restore read-only attribute	If the **Modify read-only files** option is on, denotes that the R/O attribute will be restored for such files after they are modified.
Create back-ups	If enabled, directs creating copies of files to be modified. By default, back-up copies are created in the same directory as the

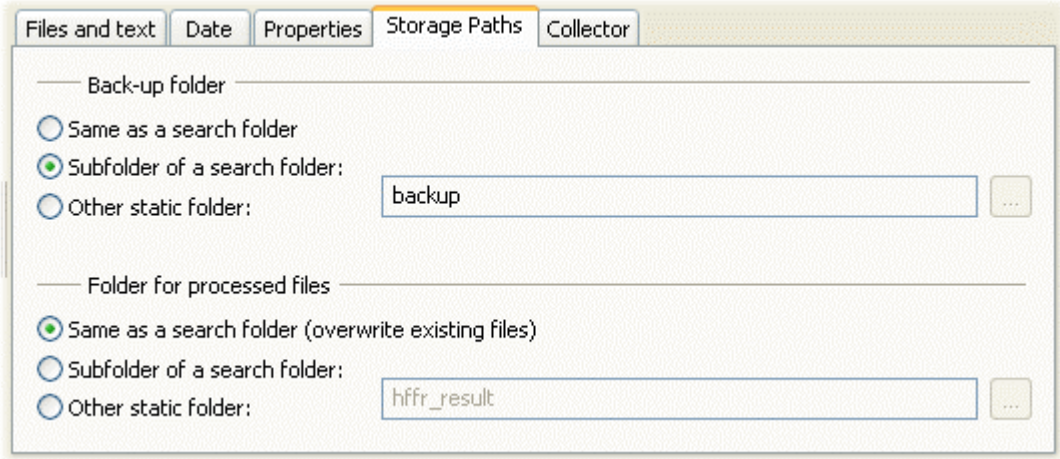
	modified file and have the `.bak` extension. To alter this behavior, open the Storage Paths Tab.
Extension	Default extension to append to back-up copies. Files with this extension are excluded from search.

HTML Report

Field	Description
Create report	If enabled, the HTML report will be created.
File path field	Specify path and name of the HTML report file here.
View report button	Click to open the report in the system default browser.

STORAGE FOLDERS TAB

Text Workbench is capable of storing the back-up copies and processed files in folders other than specified source folder(s). This tab allows to specify rules that the **TextWorkbench** will use when creating and copying files.



The tables below describe the tab fields.

Back-up Folder Group

Any of the following options is only applicable if the **Create back-ups** option is set on the Properties Tab. Otherwise, no backup copies are created.

Field	Description
Same as a search folder	Default selection. The following rules apply. ▪ Back-up files are stored in the same directory as the source files. ▪ Back-up files have the extension supplied on the Properties Tab. ▪ If the back-up file already exists, it will be overwritten. ▪ Files having the extension supplied on the Properties Tab are excluded from search.
Subfolder of a search folder	Denotes that the back-up files will be copied to a folder created in the search folder. You can set the folder name in the text field. The following rules apply. ▪ The back-up subfolder is created in the root search folder (or folders, if multiple folders are specified). ▪ The back-up subfolder structure mirrors the search folder structure recursively. ▪ Back-up files do not change the extension; they are simply copied. ▪ Back-up folders are excluded from the search.

Other static folder	Denotes that the back-up files will be copied to a supplied static folder. The folder path may be a fully qualified path:
	`c:\My files\Backup`
	or they may include environment variables:
	`%WINDIR%\TEMP`
	▪ Back-up files do not change the extension; they are simply copied.
	▪ Static back-up folders are excluded from the search.

Folder for Processed Files Group

These options control where the processed files are stored. Any of the following options is only applicable if the **Find What** field is not empty and the **Search only** option is *not* checked on the Files and Text Tab - that is, any replacement may occur.

Field	Description
Same as a search folder	Default selection. **Note** The existing files will be overwritten.
Subfolder of a search folder	Denotes that the processed files will be stored in a folder created in the search folder. You can set the folder name in the text field. The following rules apply. ▪ The target subfolder is created in the root search folder (or folders, if multiple folders are specified). ▪ The target subfolder structure mirrors the search folder structure recursively. ▪ Target folders are excluded from the search. This option cannot be used with FTP folders.

Other static folder	Denotes that the processed files will be created in a supplied static folder. The folder path may be a fully qualified path:
	`c:\My files\Backup`
	or they may include environment variables:
	`%WINDIR%\TEMP`
	▪ The target subfolder structure mirrors the search folder structure recursively.
	▪ Static target folders are excluded from the search.

Remarks

General rules for creating names for directories (and files) on the Windows platform include the following:

- Use any character in the current code page for a name, but do not use a path separator, a character in the range 0 through 31, or any character explicitly disallowed by the file system. A name can contain characters in the extended character set (128–255).
- Use the backslash (\), the forward slash (/), or both to separate components in a path. No other character is acceptable as a path separator.
- Use a period (.) as a directory component in a path to represent the current directory.
- Use two consecutive periods (..) as a directory component in a path to represent the parent of the current directory.
- Use a period (.) to separate the base file name from the extension in a directory name or file name.
- Do not use the following characters in directory names or file names, because they are reserved: **< > : " / \ | * :**
- Do not use device names, such as *aux*, *con*, *lpt1*, and *prn*, as file names or directory names.
- Do not assume case sensitivity. Consider names such as *OSCAR*, *Oscar*, and *oscar* to be the same.
- The following reserved words cannot be used as the name of a file: CON, PRN, AUX, CLOCK$, NUL, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, and LPT9. Also, reserved words followed by an extension—for example, NUL.tx7—are invalid file names.

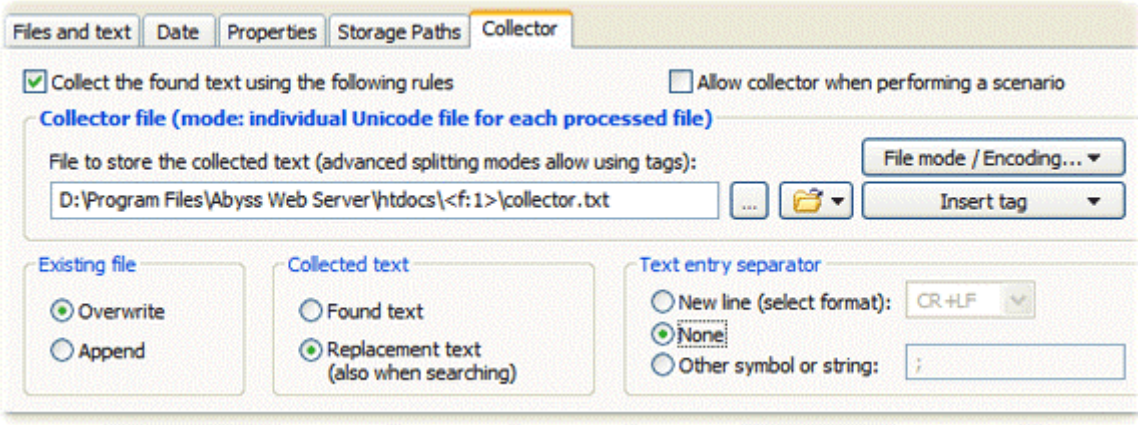
(Source: *Microsoft Developer Network* http://msdn.microsoft.com)

COLLECTOR TAB

Functions of this tab allow you to store the found text (or the text you want to replace the found text with) in an arbitrary file. Such collector function allows to:

- generate CSV files with e-mail addresses from a number of HTML pages;
- form a source code documentation draft by collecting the function prototypes and placing them in separate files;
- perform any other operations that require data gathering.

If you do not need special expert features, you can use the **Single File** collector mode. The **Collector** function is generally useful if searching with regular expressions, as it allows to find the irregular text portions.



Using The Collector

Follow the steps below to gather text from files and store it in a collector file. This sequence presumes the use of **Single File** mode.

1. Specify the search location, file mask(s), text to find (and optionally to replace - this mostly applies to regular expressions) as usually.
2. On the **Collector** tab, check the box **Collect...**
3. Specify the collector file path and name.
4. If you want to create a new collector file, select the **Overwrite** file mode. Otherwise, select the **Append** mode.
5. Set what kind of text you want to collect - **Found** text of **Replacement** text. Choosing the **Replacement text** option allows you to alter the found text in any way, even if you do not perform the actual replace.
6. Choose the kind of **separator** to insert between each added text entry.
7. Click the **Search** button (or **Replace** if you need to simultaneously perform the replace operation).

Tab Fields

The tables below describes the tab fields.

General Options

Field	Description
Collect the found text using the following rules	If checked, creates (or reuses) the collector file and stores the text in it. If unchecked, no collecting occurs.
Allow collector when performing a scenario	If this option is checked, the collector function is allowed if you are executing a scenario. Otherwise, the collector is not active if searching using a scenario.

Collector file

Field	Description
File to store the collected text	Path and name of the file to collect the text. If the file and/or path do not exist, they will be created. Path and file format depends on the current *splitting mode*. ▪ **Single file** mode does not allow special tags. You have to specify the file path and name in common format. ▪ **Individual file for each processed file** mode allows the use of tags `<F:n>`, which are used to insert tagged parts of the current *F*ile name if regular expressions are used as the file filter. If you do not use regular expressions in your file filter, you can use tag `<F:0>` to insert the whole file name. ▪ **Individual file for each match** mode allows the use of tags `<F:n>` just in the same way as for the previous mode, plus tags `<M:n>`. These tags insert the contents of the stored expressions (*M*atches) obtained when processing a file. You define stored expressions in the **Find What** field of the Files and Text tab. **Regular expressions** option *must* be turned on to use tags `<M:n>`. The modes **Individual file for each processed file** and **Individual file for each match** allow using *other special tags*. See the **Remarks** section below for the list of these tags. See **Remarks** below for detailed description of the collector file modes. The collector file is always created in the ANSI format, regardless of the processed file nature. Tags are case-insensitive. This means that `<F:n>` is equivalent to `<f:n>` and `<M:n>` is equivalent to `<m:n>`.
Browse for file button	Click this button to locate and select the file to store the collected text.

...	
Open file button	Clicking this button will display a menu with the following commands: ▪ **Open File** - opens the file using the associated application. ▪ **Open File Folder** - opens the Windows Explorer and locates the file in its right pane.
File mode / Encoding button	Click this button to set how the collector file will be created (creation mode and encoding). See **Remarks** below for detailed description of the collector file modes.
Insert tag button	Click this button to insert a tag of one of the two allowed types: <F:n> or <M:n>. This button is the shortcut to inserting these tags.

Existing file mode

Field	Description
Overwrite	Select this option to replace the data in the existing file with the new contents. In the file does not exist, it will be created.
Append	Select this option to append the collected data to the file contents. In the file does not exist, it will be created.

Collected text

Field	Description
Found text	If selected, the found text is stored in the collector file.
Replacement text (also when searching)	If selected, the text that you've specified in the **Replace With** field is stored in the collector file. If you use Regular Expressions, you can store the replacement text even if simply *searching* files for text. This allows you to alter the found text in any aspect.

Text entry separator

Field	Description
None	The text is collected as-is, no additional data is written to a file.
New line	Finalizes each write operation to a collector file with a CR+LF pair, CR or LF.
Other symbol or string	Allows to insert arbitrary text between each write. Useful for creating CSV files.

Remarks

The following is the additional information on the formatting rules that apply to

collector files.

Special tags

The following special tags can be used with **Individual file for each processed file** and **Individual file for each match** modes.

Tag	Description

<rnd:*N*>	Inserts a string of not more than *N* random numbers.
	
	For example, the path:
	`C:\web\Review_<rnd:5>.html`
	
	may be formatted as:
	`C:\web\Review_CE3A8.html`

<count:*N*>	Inserts a numerical value which is initially 1, and incremented each time a new file name is formatted. If the number of digits in the counter is less than the specified value *N*, the output value is padded on the left with zeros. The counter value is reset every time the **Search** or **Replace** button is clicked.
	
	For example, the path:
	`C:\web\<count:4>.catalog.item.html`
	
	will create the following files:
	`C:\web\0001.catalog.item.html`
	`C:\web\0002.catalog.item.html`
	`...`
	`C:\web\0009.catalog.item.html`
	`C:\web\0010.catalog.item.html`
	`...`

Collector file modes. Encoding

The collector file can be created in one of the 3 formats: **ANSI**, **Unicode** and **UTF-8**. You can choose the desired encoding by clicking the **File mode / Encoding** button and selecting it in the menu.

> **Note!**
> If you choose the ANSI encoding, and the programme encounters a Unicode file and finds text to be added to the collector file, the text will be converted to single-byte (ANSI) representation using the system code page.

There are three collector file modes, varying in both complexity and capability.

Single File

This is the simplest mode. The **single file** mode stores all the found data in a single file, path and name provided. You can use this mode, for example, to collect any information from your files. For example, you can collect e-mail addresses from multiple pages and store them in a single file.

Example

To store all e-mail addresses form your HTML pages, select the **Single File** mode, specify path to the collector file (e.g. `C:\myweb\emails.csv`), type the regular expression in the Files and Text tab to describe the e-mail pattern (e.g. `[\w\.\_\d]+\@[\w\.\_\d]+\.\w+`), set the **Collected text** option to **Found text**, **Text entry separator** to **New line** and click **Search**.

Individual file for each processed file

This mode creates a new collector file every time a new found file is started to process. This allows you to separate the found text and distribute it to individual files.

File name creation: "Individual file for each processed file" mode

If you need separate files for each processed file, you would have to define a special rule to form a collector file name and/or path. **Text Workbench** offers you a special *tag* to define parts of the collector file name and/or path. The tag has the following format: `<F:n>`. This tag inserts the contents of the stored expression used in the file filter expression, numbered by *n*. To insert the while name of the currently processed file, you can set *n* to 0.

Example

If you define the following file filter: `(get\w+)\.html` and specify the following collector file pattern:

```
C:\result\file_<F:1>\file.<F:0>.result.txt,
```

the collector will create a file

```
C:\result\file_getID\file.getID.html.result.txt
```

when **Text Workbench** starts to process file `getID.html`.

Individual file for each match

This is the most powerful though the most complicated mode. This mode creates a new collector file *every time a new occurrence of the matching text is found*. This allows you to create individual files for each found text fragment.

File name creation: "Individual file for each match" mode

To define a pattern for collector files, you can use tags `<F:n>` as with the **Individual file for each processed file** mode as well as tags `<M:n>`. This tag is equivalent to the regular expression operators `\n` used in replace pattern. This tag inserts the contents of the stored expression numbered by *n*. To insert the whole found text in the file path or name, you can set *n* to 0. You define stored expressions in the **Find What** field of the Files and Text tab. **Regular expressions** option *must* be turned on to use tags `<M:n>`.

Example

If you define the following file filter: `(\w+)\.cpp` and specify the following search expression in the Files and Text tab that would match a C++ class method definition:

```
{[\_\w\d]+}?\s*(C[\w\d]+)\:\:([\_\w\d]+)
```

and specify the following collector file pattern:

```
C:\myclassdoc\class<M:1>\<M:1>.<M.2>.<F:1>.txt
```

the collector will create a file

```
C:\myclassdoc\classCMyList\CMyList.AddTail.MyList.txt
```

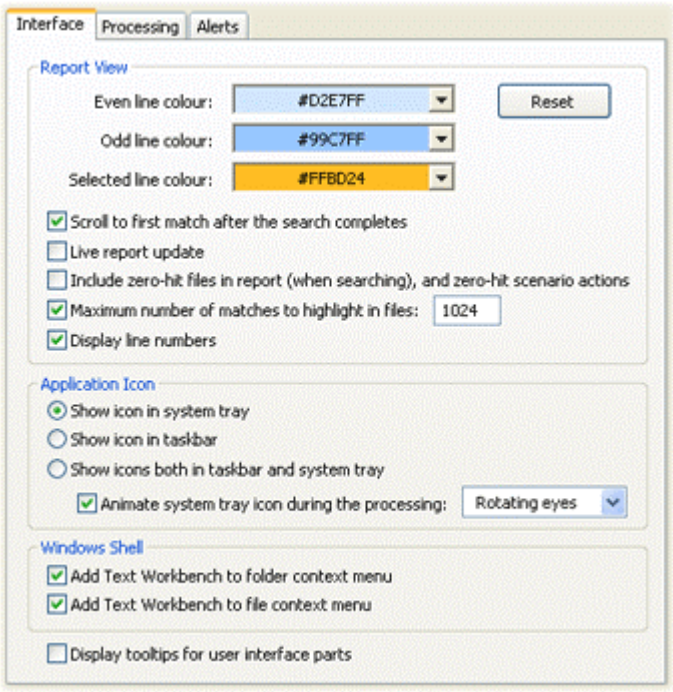
when **Text Workbench** processes file `MyList.html` and finds the method `CMyList::AddTail`.

The Options Dialog

THE OPTIONS DIALOG - INTERFACE

The **Interface** option page contains settings that define the display mode and style of **Text Workbench**.

You can click on any image area to scroll to the description.



Dialog Fields

The tables below describe the dialog fields.

Report View

This group of controls defines the appearance and behaviour of the report areas: the match tree and the file viewer.

Field	Description
Even line colour	Defines the colour of the first (zero-based) and the consecutive alternate matches in the file match report and the file viewer.
Odd line colour	Defines the colour of the second (zero-based) and the consecutive alternate matches in the file match report and the file viewer.
Selected line colour	Defines the colour of the selected match in the file match report and

	the file viewer.
Reset	Click to reset colours to their default values.
Scroll to first match after the search completes	If checked, the first file in the tree will be selected and its contents loaded in the viewer.
Live report update	If this option is unchecked, the search results are added to the file match tree and no scrolling to the last added item occurs. If this option is checked, the search results are added to the file match tree and the tree scrolls every time a new item is added. **Enabling this option slows down the processing speed.**
Include zero-hit files in report (when searching) and zero-hit scenario actions	Text Workbench versions prior to 3.1 did not include files not containing the target text in the report when performing the *search* (unlike the *Replace*). This option allows to include files not containing the sought text in the report. If a scenario is active and this option is enabled, scenario actions that do not produce any matches, are also included in the report. Otherwise, only actions with matches are included.
Maximum number of matches to highlight in files	Sometimes, especially when you process and then view very large files that may contain a lot of occurrences of the sought text, you can notice the process of match marking in the **Browser** view is extremely time consuming. This option lets you put restrictions on the maximum number of matches that can be marked in the **Browser** view. The most reasonable value lies over the range of 1000 matches.
Display line numbers	If checked, the browser text view will show line numbers.

Application Icon

Field	Description
Show icon in system tray	This option tells to place the application icon in the system tray (area that contains the system clocks). The application icon will not be listed on the Windows Taskbar when **Text Workbench** window is minimized, thus allowing to save space.
Show icon in taskbar	This option tells to list the application icon on the Windows Taskbar (the default behavior).
Show icons both in taskbar and system tray	This option combines the previous two modes: tells to place the application icon in the system tray and on the Windows Taskbar.
Animate system tray icon during the processing	If you select to display an icon in the tray, you can check this option to enable the icon animation during the process time. The animation had been optimized to be not resource-consuming, so you can use it as an indicator of any long-lasting operation. You can choose between the two animation types: ▪ rotating:

	▪ and blinking:

Windows Shell

Field	Description
Add Text Workbench to **folder** context menu	If checked, the *folder* context menus in Windows Explorer will display command **Process with Text Workbench**. This provides quicker access to Text Workbench functions from within Windows shell.
Add Text Workbench to **file** context menu	Checking this option includes the command **Process with Text Workbench** in the *file* context menus in Windows Explorer.

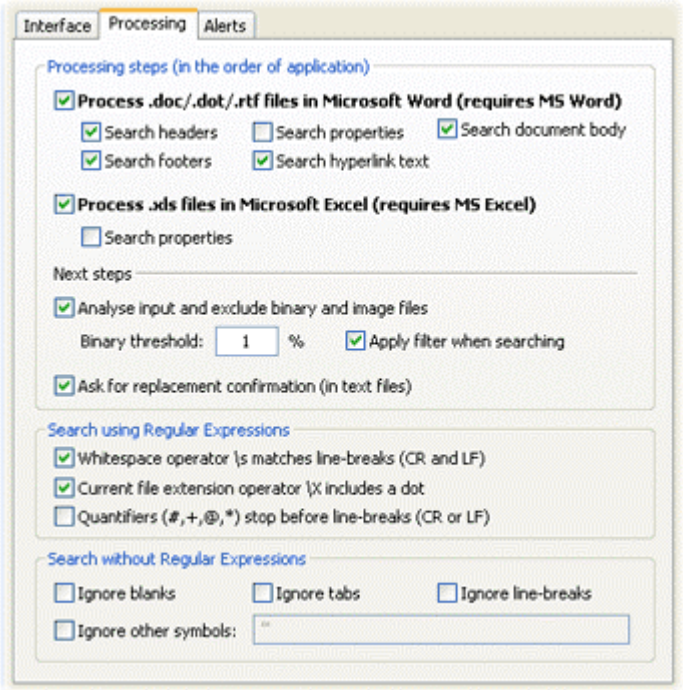
Other options

Field	Description
Display tooltips for user interface parts	If checked, *tooltips* (small text labels) will appear when you hover the mouse pointer over any control (button, field etc) of the programme window. Tooltips give brief description of these controls.

THE OPTIONS DIALOG - PROCESSING

The **Processing** option page contains settings that define how **Text Workbench** treats files and text.

You can click on any image area to scroll to the description.



Dialog Fields

The tables below describe the dialog fields.

Processing steps

Field	Description
Options in this group are listed just in the same order that they are applied to input files.	
Process .doc/.dot/.rtf files in Microsoft Word (requires MS Word)	If this option is checked, all **.doc**, **.dot** and **.rtf** files will be processed using the installed **Microsoft Word** engine. This allows to modify these files correctly, ensuring their safety and integrity. The following restrictions apply when processing **.doc**, **.dot** and **.rtf** files in Microsoft Word. ▪ Regular Expressions cannot be used. ▪ Microsoft Word does not report the true match count; the only information that can be obtained is that the sought text is found (or not found) in a file. ▪ **Collector** does neither apply to Word documents, nor it collects any text in such files.

	▪ As processing a Word file requires running a Microsoft Word instance, the processing speed is rather slow. If this option is unchecked, any **.doc** or **.dot** file will be reported as a binary file if the binary filter is switched on. **RTF** files may report that the text is found, but we *do not recommend* using simple text replace on **RTF** files to avoid data corruption.
Search headers	If checked, headers of a Word document are also processed, not only the main content.
Search footers	If checked, footers of a Word document are processed as well as the main content.
Search properties (*Word*)	If checked, Text Workbench searches (and replaces if needed) common document properties of a Word document.
Search hyperlink text	If checked, text of hyperlinks in a Word document are processed as well.
Process .xls files in Microsoft Excel (requires MS Excel)	If this option is checked, all **.xls** files will be processed using the installed **Microsoft Excel** engine. Checking this option is the only way to find and replace text in your **XLS** files. This allows to modify these files correctly, ensuring their safety and integrity. The following restrictions apply when processing **.xls** files in Microsoft Word. ▪ Regular Expressions cannot be used. ▪ Microsoft Excel does not report the true match count; the only information that can be obtained is that the sought text is found (or not found) in a file. ▪ **Collector** does neither apply to Excel documents, nor it collects any text in such files. ▪ As processing an Excel file requires running a Microsoft Excel instance, the processing speed is rather slow. If this option is unchecked, any **.xls** file will be reported as a binary file if the binary filter is switched on.
Search properties (*Excel*)	If checked, Text Workbench searches (and replaces if needed) common document properties of an Excel document.
Analyse input and exclude binary files	If this options is checked, each file that is about to be processed is checked for presence of non-printable bytes. The main intention of this option is to ensure the safety of binary files if the file filter mask is set to * (match all file names and extensions) . Normal text files do not contain non-printable characters. The only allowed non-printable symbols are blanks (`\x20`), tabs (`\x09`), carriage returns (`\x0D`) and line feeds (`\x0A`). If the file contains any other non-printable characters (with the code less than `\x20`), it is considered *suspect*. You can adjust the suspect value by using the *Binary threshold*

	parameter. Additionally, this option tells **Text Workbench** to analyse input when replacing and recognize **UTF-8** files without **BOM**'s (byte order mark). If you turn this option off, non-standard **UTF-8** files (those without **BOM**) will be processed as **ANSI** files without loss of information. **Enabling this option slows down the processing speed.**
Binary threshold	This parameter defines the maximum quota of non-printable characters allowed in the suspicious file. If the file exceeds this value, it is rejected and is not processed. For example, you can set the value to 1% to allow the incorrectly formatted text files to be processed. As researches show, this is the best value. A value of 0% rejects all suspicious files. A value of 100% is similar to unchecking the option *Analyse input and exclude binary files*.
Apply filter when searching	If checked, files will be checked for binary content when searching and replacing. This allows to find binary or invalid text files. If unchecked, files will be checked for binary or **UTF-8** content when replacing only. Uncheck this option if you want image files (pictures) to be included in the search results so you can view them.
Ask for replacement confirmation (in text files)	If this option is checked, the Replacement Confirmation Dialog will appear asking you to accept or decline the replacement of every occurrence of the sought text in each processed file. This option is only effective for text files. This does not apply to Microsoft Word or Excel documents.

Search using Regular Expressions

Field	Description
Whitespace operator \s matches line-breaks (CR and LF)	Normally, the regular expression operator `\s` matches line-breaks in addition to whitespace and tabs. This allows to find the irregular text blocks (that differ in formatting) easily. You may choose to turn this option off for some reason.
Current file extension operator \X includes a dot	The operator `\X` that is used in the *Replace* expressions inserts an extension of the file under process - for example, `.html`. Uncheck this option to not include the leading dot in the extension: `html`.
Quantifiers (#,+,@,*) stop before line-breaks	Normally, the regular expression quantifiers, if they control the any-symbol operator (**.**), match any symbols either as much as possible (greedy ***** and **+**), or as less as possible (non-greedy **@** and **#**). Checking

(CR or LF)	this box alters this behavior and disallows quantifiers to span line breaks even if patterns like **.+** are used to match blocks of text. This option affects the behavior of any-symbol operator only (**.**). This allows you to scan files line-by-line. If this option is on, the stand-alone operator .+ will find the whole non-empty line. If this option is off, the stand-alone operator .+ will match the whole file. **IMPORTANT!** Please note that, with this option **on**, stand-alone patterns that search for 0 or more of any symbols (**.@** or **.***) will cause infinite loops on line ends. Avoid using these operators stand-alone when this option is active. The only operators that can be orphaned (again, when this option is active) are **.+** and **.#**

Search without Regular Expressions

Field	Description
These options affect the search exactness. *Spanning* some or all of the formatting characters allows to find blocks of text similar to the target string but differentiating in formatting. See the **Remarks** below.	
Ignore blanks (\x20)	Skip all blanks (character code `\x20`) in both the processed text and the search string when searching. That is, number of blanks and their position in the text do not affect the match/no match result.
Ignore tabs (\x09)	Skip all blanks (character code `\x09`) in both the processed text and the search string when searching.
Ignore line-breaks (CR and LF)	Skip all carriage returns (`\x0D`) and line feeds (`\x0A`) in both the processed text and the search string when searching.
Ignore other symbols	Allows to specify and skip the user-defined symbols in both the processed text and the search string when searching.

Remarks

Spanning options

These options are extremely useful if you need to find some heterogeneous blocks of text and do not want to use Regular Expressions. For example, if you use a WYSIWYG HTML editor, you notice that it formats the code in a higgledy-piggledy fashion. The formatting is generally performed using blanks, tabs and line-breaks. Say, you need to find the following code:

```
<a href="http://www.mysite.com">
<img src="images/button.gif" alt="image" border="0"
width="128" height="82">
</a>
```

The WYSIWYG formatter might write it like this:

```
<a href="http://www.mysite.com"><img src="images/button.gif"
alt="image" border="0" width="128" height="82"></a>
```

or like this:

```
<a
href="http://www.mysite.com"><img
src="images/button.gif" alt="image" border="0" width="128"
height="82"></a>
```

or even like this:

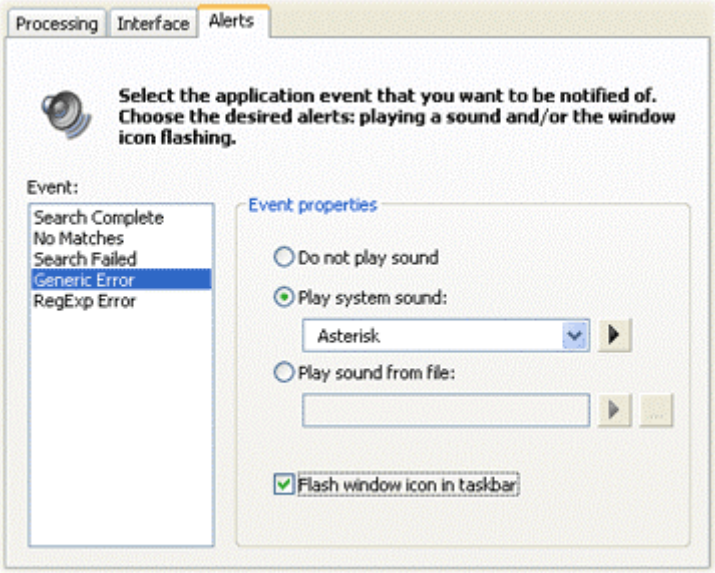
```
<a
href="http://www.mysite.com"><img
src="images/button.gif"
alt="image"
border="0"
width="128"
height="82"></a>
```

You can easily handle all cases using Regular Expressions, but if you do not want bother yourself or not familiar with them, you can use the spanning options. Simply check all the three boxes (to match the text in this example) and Text Workbench will find the string.

Note
Use of the spanning options results in search speed degradation.

THE OPTIONS DIALOG - ALERTS

The **Alerts** option page contains settings that allow you to configure how **Text Workbench** will notify you of the process completion and/or errors.



Dialog Fields

The tables below describe the dialog fields.

Events

This box contains a list of available application events. To assign any alert type to an event, select it in the list and configure it using the **Event properties** controls.

Event	Description
Search Complete	This event fires when the search (and/or replace) operation completes without any error and one or more matches was found.
No Matches	This event fires when the search (and/or replace) operation completes without any error and no matches was found.
Search Failed	This event fires if the operation fails due to any Microsoft Office error when processing the MS Office files, or a file cannot be accessed.
Generic Error	This event fires when the operation fails due to any system or preprocessing error.
RegExp Error	This event fires if the provided regular expression contains errors.

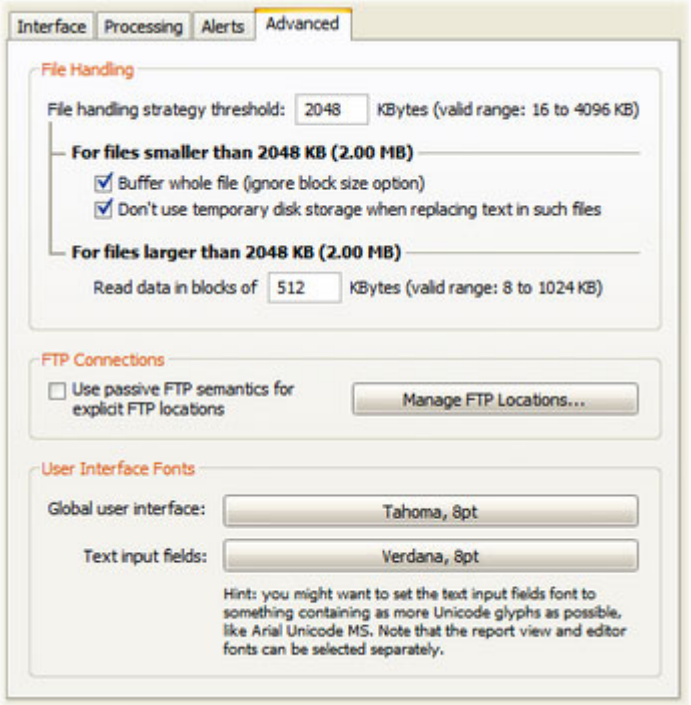
Event properties

Field	Description

Do not play sound	Does not emit any sound on event.
Play system sound	Plays a system-defined sound (configured in the **Control Panel**) on event.
Play sound from file	Plays a user-supplied sound (WAV file) on event.
Flash window icon in taskbar	Tells to flash the application icon in the Windows Taskbar if an event fires.

THE OPTIONS DIALOG - ADVANCED

This page contains settings for advanced control over **Text Workbench** behaviour.



Dialog Fields

The tables below describe the dialog fields.

File Handling

This group of controls provides a user with full control over the aspect of the application's file handling algorithm that can affect performance and memory consumption.

Field	Description
File handling strategy threshold	Specifies the maximum file size, in KB for which unbuffered processing is allowed. The files smaller than this value may be read into the system memory entirely. Files larger than this value will always be processed chunk by chunk without noticeable performance degradation, which enables Text Workbench to process files of unlimited size.
For files smaller than... (the threshold value)	
Buffer whole file	If checked, the file will be read into memory entirely if the file size is less than the threshold value. Otherwise, the file will be read in chunks of size specifies for large files.
Don't use temporary	This option takes effect only when replacing text in files. It is not used when performing search.

disk storage when replacing text in such files	If checked, Text Workbench allocates memory enough to hold the new file contents. After a file has been processed, the new data is written out to disk. If unchecked, Text Workbench creates a temporary file in the same folder as a source file and writes the new data to that file. When finished, the temporary file replaces the source file.
For files larger than... (the threshold value)	
Read data in blocks of...	Specifies the size, in KB, of the read buffer. The larger the buffer, the less disk read operations Text Workbench is required to perform. Unreasonably small buffer size may result in processing speed degradation when replacing text in huge files.

FTP Connections

This group of controls defines how the application connects to FTP servers.

Field	Description
Use passive FTP semantics for explicit FTP locations	This option sets the FTP passive mode for connections specified in free form in the **Folder** field, without using the FTP accounts (e.g. when setting the folder to something like *ftp://username:password@server.com/path*). In active mode the client connects from a random port N to the FTP server's command port 21. Then, the client sends the FTP command PORT N+1 to the server. The server will then connect back to the client's specified data port from its local data port 20. In passive mode the client initiates both connections to the server, solving the problem of firewalls filtering the incoming data port connection to the client from the server. Generally, the best mode can be derived experimentally.
Manage FTP Locations	Opens the FTP Locations dialog.

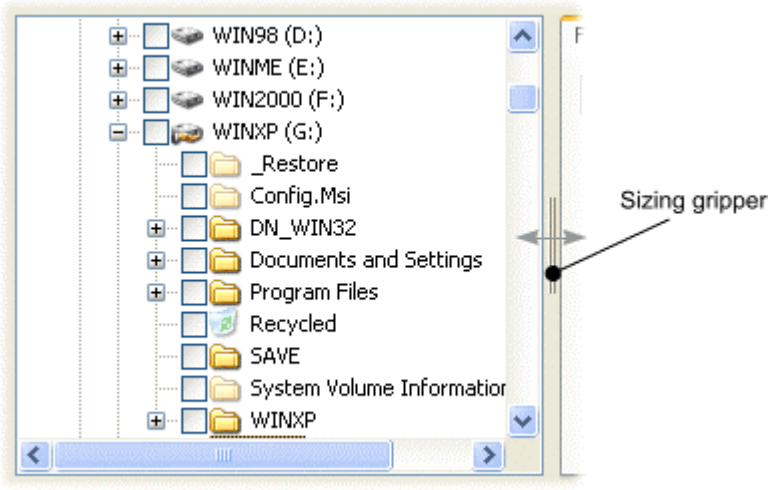
User Interface Fonts

This area serves to improve the user experience of Text Workbench.

Field	Description
Global user interface (font)	Sets the font that will be used throughout the Text Workbench windows. This may be useful for owners of laptops with small FullHD screens.
Text input fields (font)	Sets the font that will be used in the search and replace text entry fields. This includes **Find What**/**Replace With** text boxes, **Multiline Editor** window, **Regular Expression Laboratory** and other windows where a user inputs text that can be searched or inserted.

Shell Folder Tree

The **Shell Folder Tree** is shown when you click the **Folder** button on the Files and Text Tab.



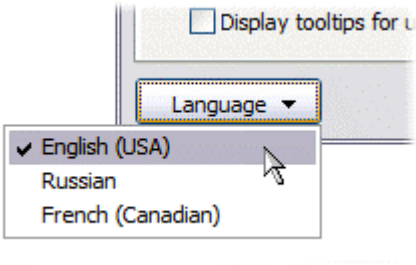
The tree displays both local and network folders and drives (but **not FTP folders**). Additionally, you will find check boxes beside each file system entity. Most of them can be checked to mark the folder or drive as the subject to search. If a check box cannot be marked, it means that it does not represent a physical storage (for example, **Network Neighborhood** folder.)

To hide the tree, click the **Folder** button again.

If you need to change the size of the folder tree, you can click on the *Sizing gripper* (see the figure above), and start dragging while holding the left mouse button.

Changing The Interface Language

Text Workbench interface can be presented in different languages. You can change the active language by opening the **Options** window (select **Options** on the toolbar), and then clicking the **Language** button:



Note that **Text Workbench** will apply the selected language next time you start the program.

If you cannot find your language in the menu, you can translate the interface into your native language. You are free to offer your conditions of co-operation. Please send your requests to support@silveragesoftware.net.

Report Area

The **Report Area** displays information on the find (and replace) process. The Report Area consists of the toolbar and the two main areas: the **File Match Tree** and the **Viewer**.

The **File Match Tree** contains information on files that has been processed and matches. Root elements of the tree displays basic information: relative file path, number of hits and the time spent for the operation.

- If any match(es) has been found in the file, they are displayed as the child elements of the file.
- Click the plus icon to expand the file element and view matches.
- Right-click on any file to invoke a context menu with options to open the file and explore the folder that contains this file.

The Toolbar

The Toolbar contains various useful commands. The command description is provided below.

Button	Description
Print	Click to print the current file tree report.
Browser	Shows or hides the text view area.
Prev	Click to browse to the previous match.
Next	Click to browse to the next match.
First	Click to go to the first file in the report.
Last	Click to go to the last file in the report.
Wrap	Toggles the word wrapping mode in the text view area.
Font	Click to select the font for the text view area.
Options	Click to display the Options dialog, which allows you to customize various useful options. The drop-down button displays a menu with the following commands: ▪ **Load Configuration** - opens the standard Windows file dialog, allowing you to select and apply the previously saved configuration file (`.hfcfg`) . ▪ **Save Configuration** - opens the Configuration Export Dialog, allowing you

	to save the current configuration and optionally send it via the e-mail.

Text Viewer

The **Text Viewer** loads the currently selected file and highlights the matching text so you can easily browse through it. To facilitate browsing, you can select a match in the tree and view it in the text viewer.

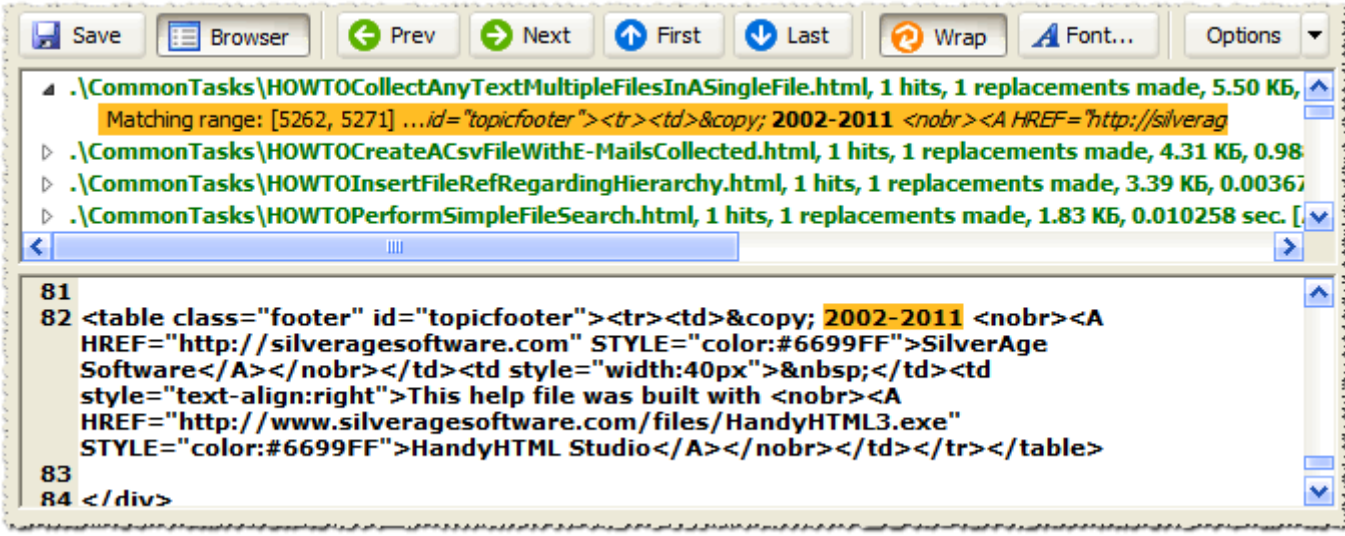
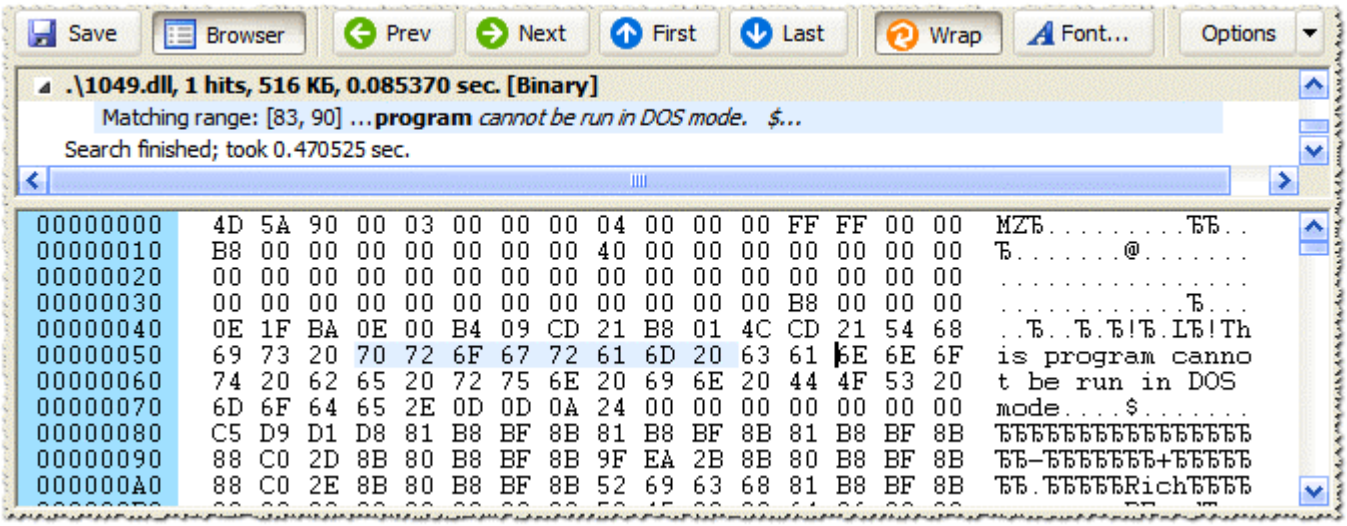


Image Viewer

If the file is an image (that is, it has an extension **bmp**, **gif**, **png**, **ico**, **jpg**, **jpeg**, **tif** or **wmf**), the Text View is replaced with the image view:

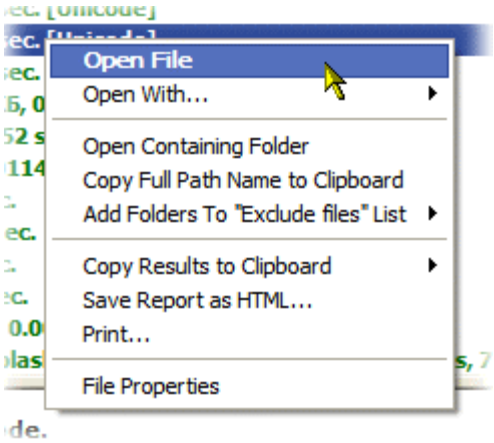
Binary Viewer

If the viewed file is considered binary (see Options for more information), the text view is replaced with hexadecimal binary view:


Save
Browser
Prev
Next
First
Last
Wrap
Font...
Options
.\1049.dll, 1 hits, 516 КБ, 0.085370 sec. [Binary]
Matching range: [83, 90] ...program cannot be run in DOS mode. $...
Search finished; took 0.470525 sec.
00000000 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZЂ.....ЂЂ..
00000010 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 Ђ.....@.....
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000030 00 00 00 00 00 00 00 00 00 00 00 00 B8 00 00 00 .....Ђ...
00000040 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ..Ђ..Ђ.Ђ!Ђ.LЂ!Th
00000050 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program canno
00000060 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS
00000070 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 mode....$.....
00000080 C5 D9 D1 D8 81 B8 BF 8B 81 B8 BF 8B 81 B8 BF 8B ЂЂЂЂЂЂЂЂЂЂЂЂЂЂЂЂ
00000090 88 C0 2D 8B 80 B8 BF 8B 9F EA 2B 8B 80 B8 BF 8B ЂЂ-ЂЂЂЂЂЂЂ+ЂЂЂЂЂ
000000A0 88 C0 2E 8B 80 B8 BF 8B 52 69 63 68 81 B8 BF 8B ЂЂ.ЂЂЂЂЂRichЂЂЂЂЂ

Context Menu

The context menu appears when you click a file with the right mouse button in the **File Match Tree**.



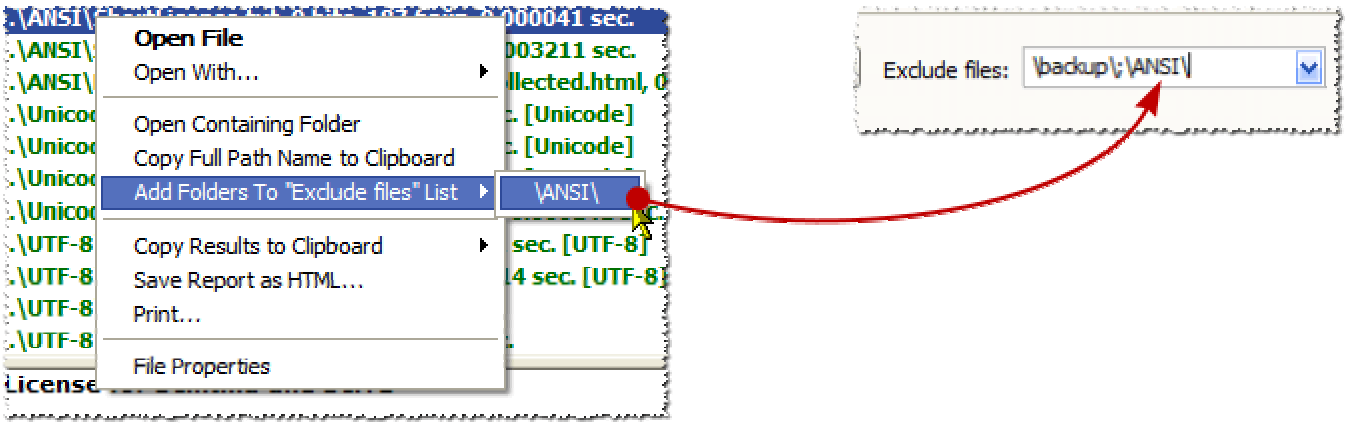
Context Menu Commands

Command	Description
Open File	Select this command to open the selected file in the associated application. **Text Workbench** uses Windows shell associations to open files. If there is no application associated with the selected file type, the standard dialog is displayed enabling to select the application in which the selected file is to be opened.
Open With	Offers a list of applications in which the selected file can be opened.
Open Containing Folder	Opens the Windows Explorer window and highlights the selected file in it.
Open Temporary Local Folder	Opens the Windows Explorer window and displays a local folder containing downloaded, processed and back-up files of an FTP server whose file is currently selected in the log view. This menu item is only displayed for remote FTP files.
Copy Full Path Name	Places the file path and name on the Clipboard.
Add Folders To "Exclude files" List	Opens a submenu containing the names of subfolders that form the path to a file for which the context menu is being shown. Selecting a subfolder from the submenu appends it to the contents of the "Exclude files" field. See the remarks below the table.
Copy Results to Clipboard	Opens a menu with commands to place the log view contents onto the Clipboard in the text format. ▪ **Messages and Files** - copies only the first level of the log view; that

	is, diagnostic messages and information on the found files. ▪ **Messages, Files and Matches (slow)** - copies the entire contents of the log view: diagnostic messages, file information and all the matches. If there are a lot of files and matches, this operation may take a long time to complete.
File Properties	Displays a dialog in which you can alter the file properties: date and time of creation / lass access / last write; and the file attributes. This menu item is disabled for remote FTP files.

Notes on adding folders to exclusion list

For example, if you invoke the context menu for a file that exist in /ANSI/ subfolder of the search folder (the one specified in the **Folders** text box), and select the folder, the following will occur:



Mode Switch Buttons and Progress Indicator

The bottom area of the application contains several sections, each serving different functions.

The mode switch buttons

The two buttons, **Browse** and **Edit**, enable you to switch between two major functions: *file search* and *file editing*.

The first mode (*file search*) is the standard mode in which you can search files and replace text in them. After you click the **Search** or **Replace** button, the programme starts the search session and displays results in the Report Area. After that, you can select any *text* file in the report tree and click **Edit**. This will set the application in the *file editing mode* and the full featured text editor is displayed instead of the **Report Area**. You can edit the selected file as you wish: insert and paste text, perform cut and copy operations, undo and redo actions. After you finish editing, you can click the **Save** button on the editor toolbar.

To revert back to the standard file search mode, click **Browse**.

The progress indicator

The progress indicator serves three different functions.

When processing files

Visually shows the relative amount of files processed and to be processed. In some circumstances, the indicator may show value less than 100% after the processing completes. This means that not all files in the search folder satisfy the search parameters.

When viewing files

The progress of marking matches in the **Browser** view (effective for large files; if the number of matches is few, the progress is not displayed as the process is very fast in such cases).

When downloading remote files

When you click on a remote file to view it, or if you click **Edit** to edit a remote file, the indicator displays the download progress. If you click **Save** in the text

editor when editing a remote file, the indicator displays the upload progress.

The session control buttons

The two buttons, **Search** and **Replace** are used to start the search (and/or replace) session. If the **Search only** box is checked, no replacement will occur.

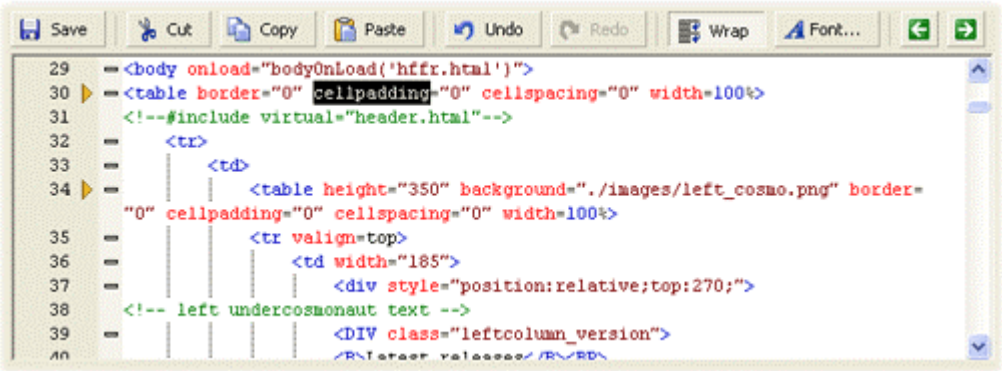
The **Help** button displays this help file.

Text Editor

The **Text Editor** becomes visible after you select any found text file and click the **Edit** button. The text editor allows you to edit any file without having to launch any external text editor.

The **Text Editor** supports all the most common text file formats: **ANSI**, **UTF-8**, **Unicode**.

If a file you have selected for editing contains entries of the sought text, the editor comes with bookmarks set against the matches. Bookmarks are indicated with ▶.



The text editor provide commands that are common to all text editors; these can be accessed via the editor toolbar.

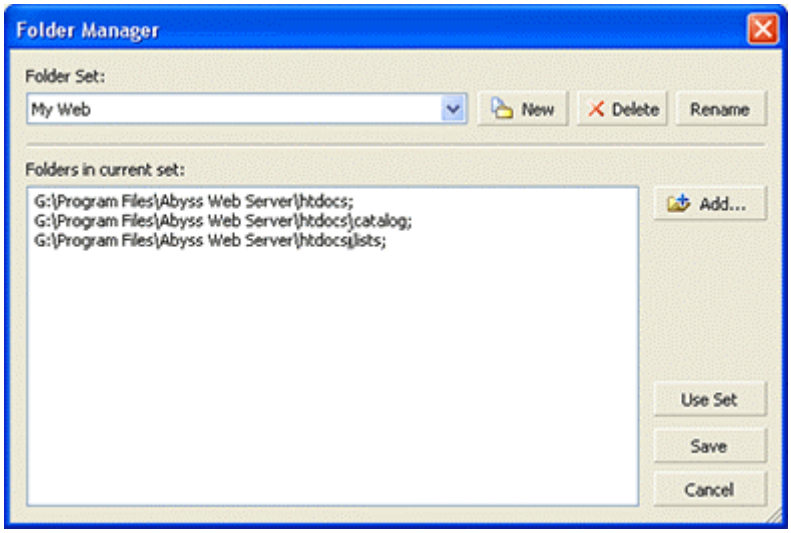
Toolbar Buttons

Button	Description
Save	Saves the active file. This button remains inactive until you make any modification to the active file. If you save a file that was opened from an FTP server, clicking this button uploads the file to the server.
Cut	Cuts the selection and places it on the Clipboard.
Copy	Copies the selection to the Clipboard.
Paste	Inserts the Clipboard contents.
Undo	Reverses the last command or deletes the last entry you typed.
Redo	Reverses the action of the Undo command.
Wrap	Toggles word-wrapping mode on or off.
Font	Changes font used by the editor to display text.

	If a file you have selected for editing contains entries of the sought text, this button navigates to the previous bookmark that has been set against the found match.
	Navigates to the next bookmark that has been set against the found match.

Folder Manager Dialog

The **Folder Manager** dialog is the convenient frond-end for specifying and storing sets of multiple folders in which the search process is to be performed.



To create a new folder set, click the **New** button. After the new set is created, you can add folders to it by clicking the **Add...** button.

The table below describes the dialog fields.

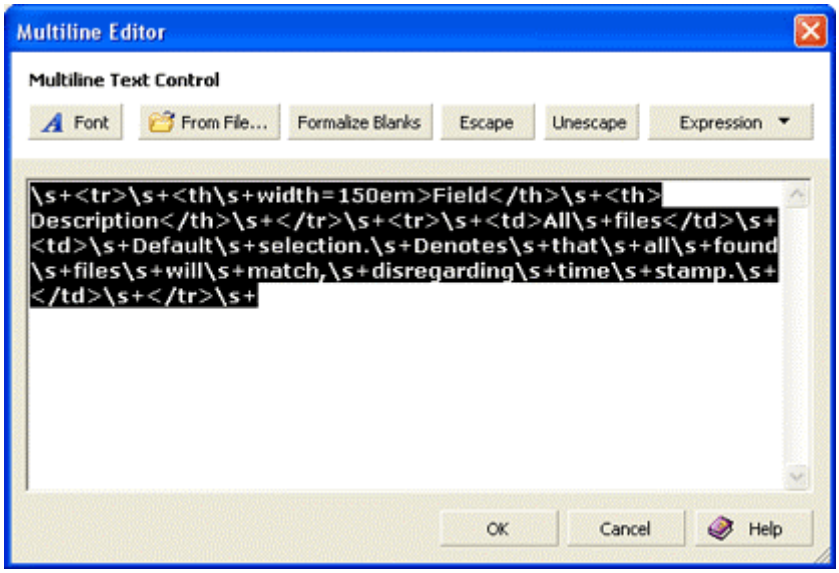
Field	**Description**
Managing Sets	
Folder Set	Displays a list of existing folder sets. You can select the set from the list by clicking on a drop-down arrow.
New	Click to create a new set.
Delete	Click to delete the selected set.
Rename	Click to rename the selected set.
Managing Folders in Set	
Folders in current set	Edit field displaying the folders in the current set. You can edit them manually here. Folders are separated by semicolon.
Add	Click to open the folder selection dialog. Select a folder and click **OK** to add it to the set.
Use Set	Click this button to save the set and apply it; that is, copy the folder paths to the **Search folders** field of the Files and Text Tab.
Save	Click this button to save changes and close the dialog.
Cancel	Click this button to close the dialog without saving changes.

Multiline Editor Dialog

When entering text in the Files and Text Tab, you can specify multiple lines of text by inserting control escapes `\r` and `\n`. This is rather inconvenient if there are a large number of text lines. Moreover, you cannot edit the multiline text as you usually do in text editors. This is why a **Multiline Editor** was created.

Multiline Editor allows you to supply multiple lines of text to search for or replace with. If the text in the Files and Text Tab contains line-breaking escapes (`\r` and `\n`), they will be converted to normal line breaks when the Editor is opened. When it closes, line breaks are converted back to the line-breaking escapes.

You can open this dialog box by clicking buttons Find what or Replace with.



The table below describes the Editor fields.

Field	Description
Font	Click to set the editor font most suitable for you.
From File	Click to load the text from any *plain-text* file in the editor. **Note** Size of the text is a limited value. If the text is a part of a scenario action, it should not exceed 8190 bytes (~ 8 kBytes). If the text is the contents of the **Find What** combobox of the **Files and Text** tab, it can be of any size, but only first 2048 bytes are kept between sessions - this is the Windows NT-based platform limitation on combo box item text retrieval commands.

<table>
<tr><td>Formalize Blanks</td><td>This button, if clicked, converts all spaces, tabs and line breaks of the selected text to the regular expression operator \s+ that would match the entire blank area. For example, the following text

<pre>
<tr>
<th width=150em>Field</th>
<th>Description</th>
</tr>
<tr>
<td>All files</td>
<td>
Default selection. Denotes that all found files will match, disregarding
time stamp.
</td>
</tr>
</pre>

will be converted into

<pre>
\s+<tr>\s+<th\s+width=150em>Field</th>\s+<th>Description
</th>\s+</tr>\s+<tr>\s+<td>All\s+files</td>\s+
<td>\s+Default\s+selection.\s+Denotes\s+that\s+all\s+found\s+files\s+will
\s+match,\s+disregarding\s+time\s+stamp.\s+</td>\s+</tr>\s+
</pre>

This ensures that all text blocks similar to this one but differing in formatting only will be found.

This button is only available if the **Regular Expressions** option is checked.</td></tr>
<tr><td>Escape</td><td>Click to insert the escape character (\) before any non-alphanumeric symbol in the selected text.

This button is only available if the **Regular Expressions** option is checked.</td></tr>
<tr><td>Unescape</td><td>Click to remove the the escape character (\) before any non-alphanumeric symbol in the selected text, if such escapes exist.

This button is only available if the **Regular Expressions** option is checked.</td></tr>
<tr><td>Expression</td><td>Click to display a menu with control escapes that you can select and insert in the text.</td></tr>
</table>

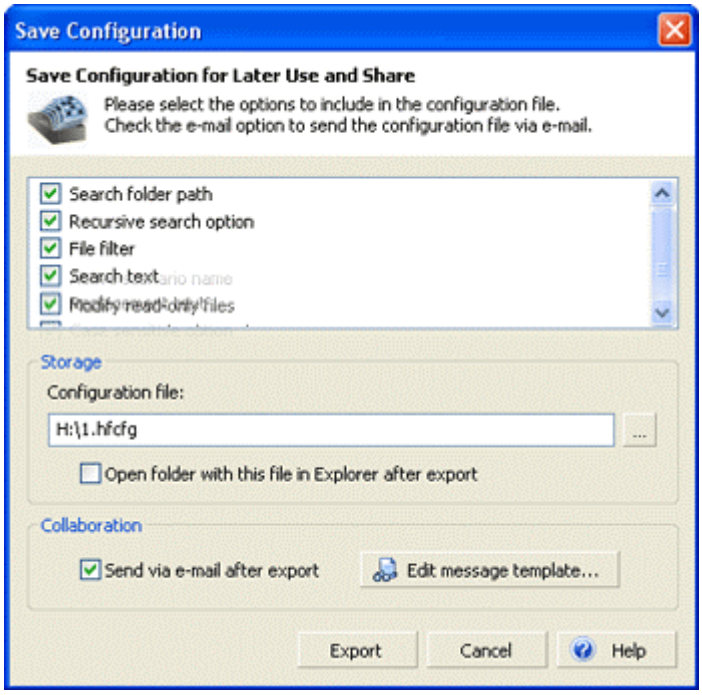
Configuration Export Dialog

A **configuration** is a collection of preferences representing the current application state. Configuration can include the search path, search and replacement text, etc.

The **Configuration Export Dialog** enables you to save Text Workbench configurations for later use. You can select options to include in the configuration file. This window, together with Scenario Export Dialog, enables you with the teamwork options.

When loading a configuration, if any of settings are not found in the configuration file, their current state remain unchanged.

You can use the exported configuration file as the command line option.



Exporting the Configuration

Follow the steps below to export the current configuration.

1. Click the drop-down arrow to the right of the Options button and select the **Save Configuration** menu item.
2. In the **Save Configuration** dialog, select the options you want to store in the configuration file by checking the corresponding boxes.
3. Specify the file in which you want to store the configuration. If you want to view the file folder in **Windows Explorer** after the export, check the **Open folder...** button.

4. If you want to send the exported configuration to your colleagues right after the export is done, check the **Send...** option. This will open the standard message creation window of your default e-mail client, with the configuration file attached.
5. Click the **Export** button.

Dialog Fields

The tables below describe the dialog fields.

Storage

Field	Description
Configuration file	A file that will contain the exported configuration. The default location for storing configuration files is the folder **My TextWorkbench Configurations** located in the **My Documents** folder. Nevertheless, you can save configuration files in any other folder.
Open folder with this file in Explorer after export	Check this option if you want to open the folder in **Windows Explorer** after the export.

Collaboration

Field	Description
Send via e-mail after export	Check this option if you want to send the exported configuration to your colleagues right after the export is done.
Edit message template... button	This option allows you to save time while preparing your message to send. Click this button to open an editor window, where you can provide your custom e-mail text that is preserved between sessions and will be used for creating new messages later. The message can contain macro `%%file%%` that is replaced with the actual configuration file name.

Replacement Confirmation Dialog

This dialog allows you to review each replacement in each processed file and accept or decline it. You have an option to accept or reject the current replacement; skip the currently processed file entirely; accept changes made to the current file and go to the next file; work silently during the present replacement session or abort the processing.

You can enable or disable the replacement confirmations on the Options Dialog - Processing page.

Dialog Fields

The tables below describe the dialog fields.

Field	Description
Existing text	The text originally stored in file. The text to be replaced is highlighted with different background colour.
New text	The text that will replace the old one according to the settings on the Files and Text Tab.
Buttons	
Accept	Click this button to accept the current and all further replacements. You will not be prompted of these changes during this run; the operation will be performed

All	silently.
Replace	Click the **Replace** button to accept and confirm the current replacement.
Skip	Click this button to cancel the proposed replacement and continue searching.
Skip File	Click this button to reject changes made to the current file (if any) and go to the next file.
Next File	Click this button to accept changes made to the current file (if any) and go to the next file.
Abort	This button stops the processing.

Regular Expression Laboratory

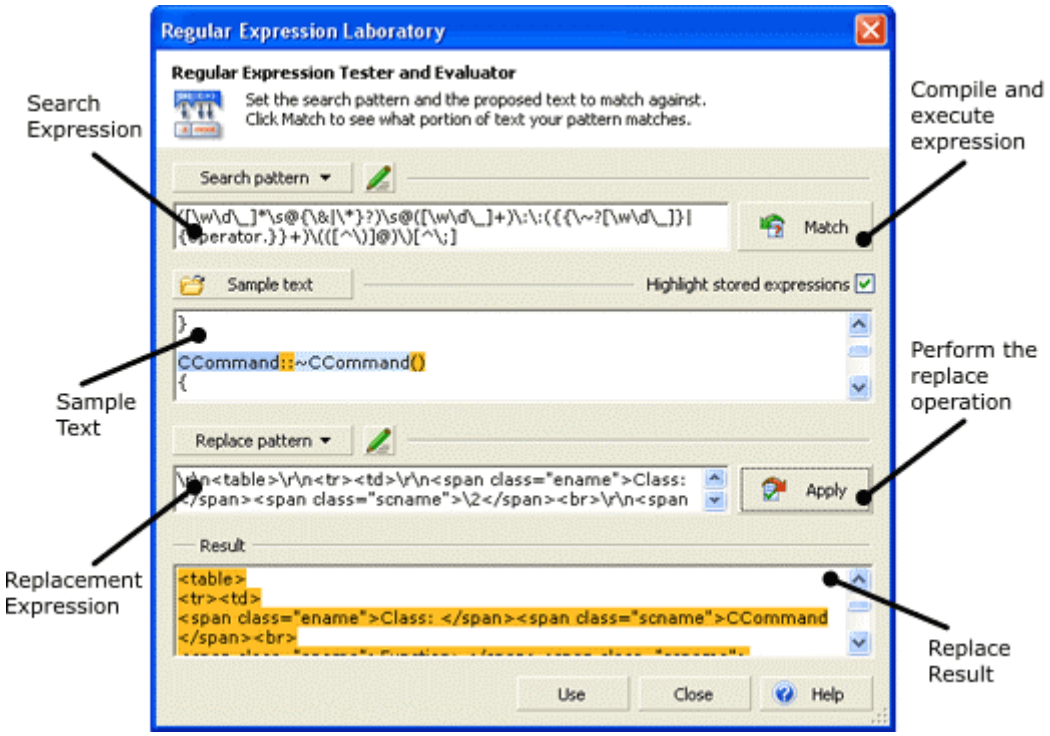
If you are not familiar with regular expressions and want to learn how to use them, or want to test your own expressions, the **Regular Expression Laboratory** will help you much. Its main intent is to visualise both the internals of the expression compilation process and how the prepared expression matches the text portions.

At the heart of the regular expression engine is a concept of a **finite-state automata**. You can treat this automata as a tiny virtual computer working by a program. The program that the automata runs is the precompiled binary form of an expression. In other words, the textual expression that you provide is first compiled into a binary form and then executed against a text.

This is why you get regular expression errors sometimes. Some applications may even crash if the regular expression faults are not handled properly. Using the **Regular Expression Laboratory**, you will easily get a robust working expression.

You can open the Laboratory by clicking button  in the Files and Text main window tab or Scenario Editor.

The following picture illustrates the **Regular Expression Laboratory** front-end.



How to Use the Laboratory

Using it is fairly simple. What is more, the Laboratory provides you with visual cues when you are mistaken with your expression. The below given sequence is usually the best practice.

1. Type your text to operate on in the **Sample text** field. As an alternative, you can load the text from file by clicking the *button* **Sample text**.
2. Type your expression in the **Search pattern** field. For example, if you want to match all HTML tags, you could type `\<[^\>]#\>`
3. Click the **Match** button. This will compile the expression and find the first occurrence of the matching text in the **Sample text** field. If the compilation fails, the erratic search expression symbol will blink with red.
4. Type the replacement text in the **Replace pattern** box. For example, if you want to convert the HTML tag to lowercase: `\L\0`
5. Click the **Apply** button. The result text box will contain the result of the replace operation.

Dialog Fields

The tables below describe the dialog fields.

Field	Description
Search pattern text box	Contains the search expression.
Search pattern button 	Shows a menu with regular expressions that you can select and insert.
Multiline editor button 	Click this button to open the **Multiline Editor** dialog. This dialog allows entering text with multiple lines in normal mode, converting line breaks to `\r` and `\n` as appropriate.
Match button	Compiles the provided expression and tries to match it against the sample text. Click this button again to find the next match in the sample text. You can press and hold this button to iterate through the text.
Sample text box	Contains the sample text that the expression will match against.
Sample text button	Allows to load the sample text from file.

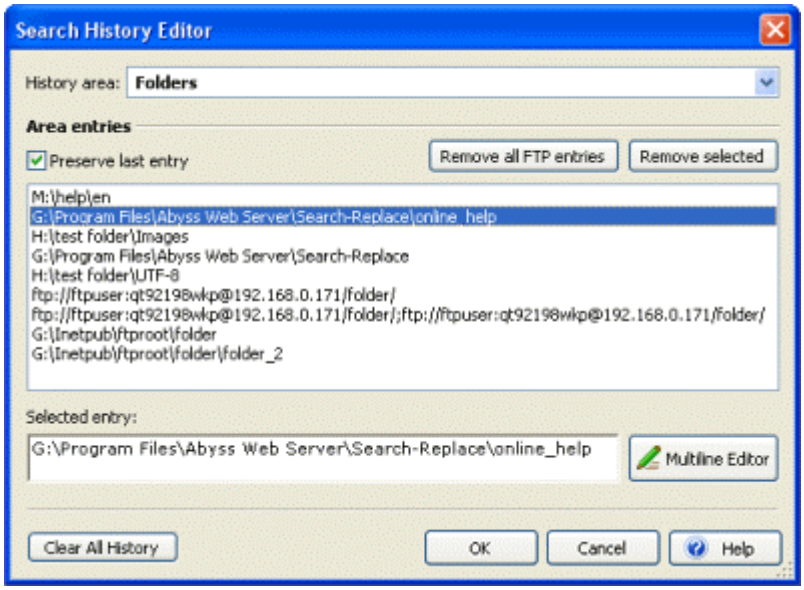
Sample text	
Highlight stored expressions	If checked, the matching stored expressions, i.e. those enclosed in parentheses ( ) will have different background colour in the sample text box.
Replace pattern text box	Contains the replacement expression.
Replace pattern button Replace pattern ▾	Shows a menu with regular expressions that you can select and insert.
Multiline editor button	Same purpose as for the **Multiline editor** button for the **Search pattern**.
Apply button	Given the replacement expression, performs the replace operation on the sample text.

Search History Editor

When you select a new search folder, or type a new text in the **Find What** or **Replace With** fields, and then click either **Search** or **Replace**, your new entry is stored in the corresponding list, so you can select and use them later.

The **Search History Editor** dialog is an auxiliary tool that you can use to delete or edit items stored in the following lists:

- Folders
- Search files (file masks)
- Exclude files
- Find what
- Replace with



Dialog Fields

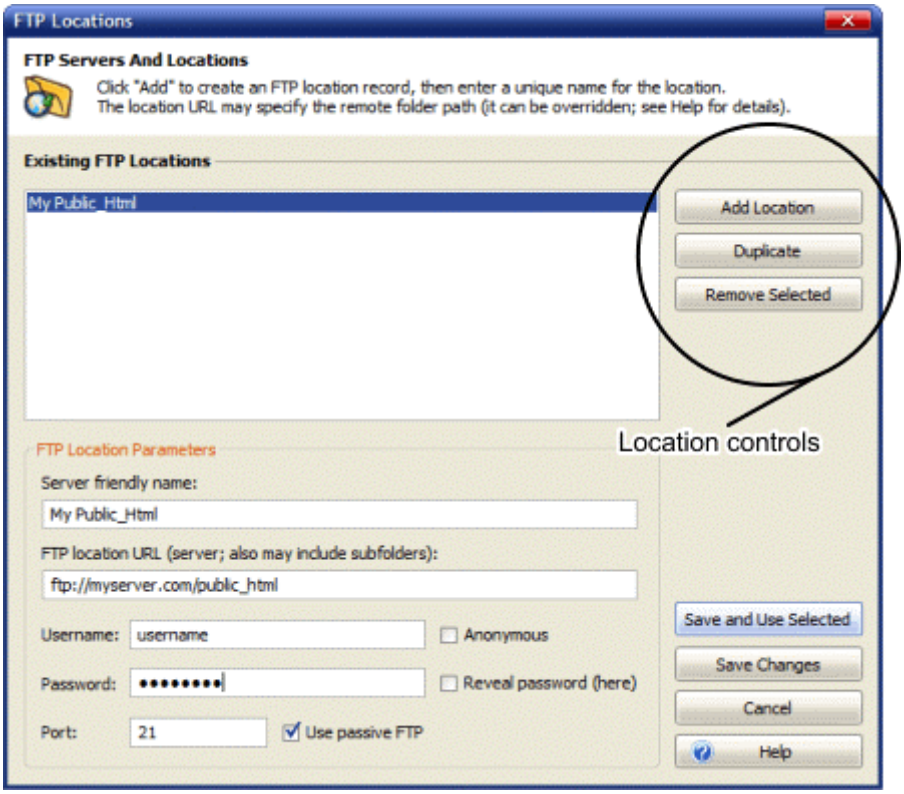
The tables below describe the dialog fields.

Field	Description
History area drop-down list	Select here the type of history to display.
Area entries	
Preserve last entry option	If this option is checked, clicking **OK** or **Clear All History** will not delete the text that is currently typed in the affected fields. If this option is not checked, current text will be deleted.

Remove all FTP entries button	This button is displayed only if **Folders** is selected in the history area drop-down list. Click it to erase FTP folders from all entries of the folder history. If an entry contains both local and FTP folders separated with semicolon, only FTP folders are erased. If an entry consists entirely of FTP folders, the whole entry is deleted.
Remove selected button	Removes the selected items. You can select and delete more than one item at once.
Selected entry	Displays the contents of the selected history item. You can edit it here.
Multiline editor button	Click to edit the selected history item in the multiline editor.
Global	
Clear All History button	Clicking this button erases history of all lists at once. The **Preserve last entry** option controls whether the currently typed text is also to be deleted.

FTP Locations

Here you can create and edit the FTP aliases, or short yet sensible identifiers for the remote FTP servers and folders. These aliases can thereafter be used to search and replace text on a remote FTP server. For example, you can create an alias "*My Personal FTP*" for your FTP server (and folders on that server), and then process files there by selecting the alias from the menu (see here) or by entering *ftp://<My Personal FTP>* in the **Folder** text field.



Dialog Fields

The tables below describe the dialog fields.

Existing FTP Locations

Field	Description
Locations list	A list of currently existing locations. To display the location properties, simply select it.
Add Location	Creates a new location.
Duplicate	Create a new location containing the same parameters as the selected one (copies the selected location).
Remove	Deletes the selected location.

Selected	

FTP Location Parameters

Field	Description
Server friendly name	Enter here the name that will uniquely identify the location and will be sensible to you. This name must be unique. This field refers to a "server name" rather than a location because the path can be overridden manually in the **Folders** text box (see the notes below).
FTP location URL	Enter here the location address, which can be: ▪ an unqualified or fully-qualified FTP server address (that is, *myserver.com* or *ftp://myserver.com*); ▪ an unqualified or fully-qualified FTP URL (the server address and the path, e.g. *ftp://myserver.com/folder1/folder2*). For tips on using the FTP paths, see the notes below.
Username	Specifies a valid username on the FTP server.
Anonymous	If checked, attempts to log in the server anonymously (*USER anonymous*).
Password	A corresponding password to log in the server.
Reveal password	If checked, shows the password characters. You may need this toggle to enter a complex and long password. This option has effect in this dialog box *only*; your password actually is not displayed anywhere else.
Port	Specifies the port to connect to the server.
Use passive FTP	Tells the server to enter "passive mode". In passive mode the client initiates both connections to the server, solving the problem of firewalls filtering the incoming data port connection to the client from the server. Generally, the best mode can be derived experimentally.

Common Controls

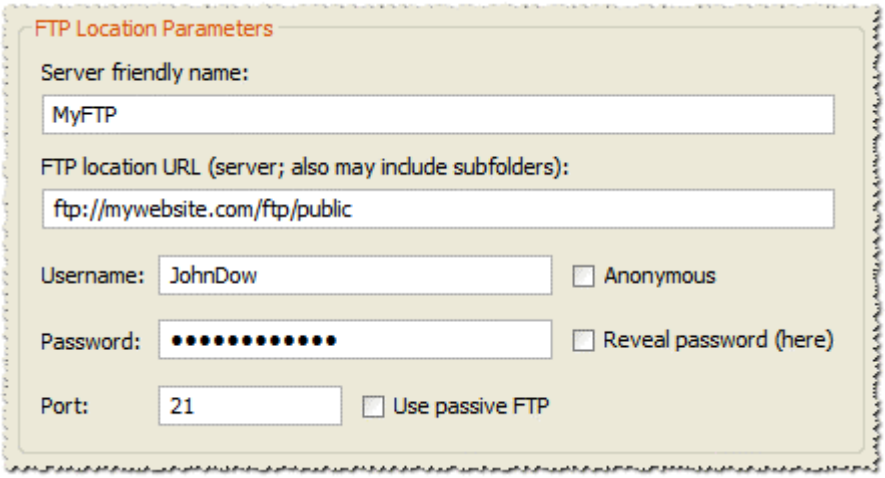
Control	Description
Save and Use Selected	Saves changes and puts the selected location in the format *ftp://<location_name>* to the **Folders** text box in the main application window.
Save Changes	Saves changes and closes the dialog.
Cancel	Closes the dialog discarding changes.
Help	Shows this help.

Notes on using FTP paths

The remote folder path which comes after the the server address may be

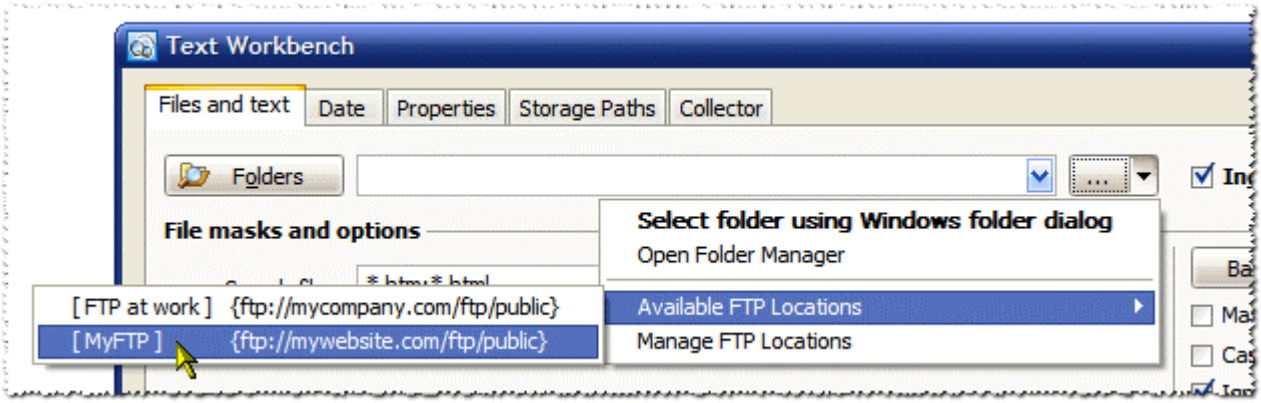
overridden in the **Folders** text box. This aspect of handling the FTP aliases has deliberately been made extremely flexible: a use case will worth a thousand words of explanation.

Assume you have previously created a location named "MyFTP" which refers to *ftp://mywebsite.com/ftp/public*:

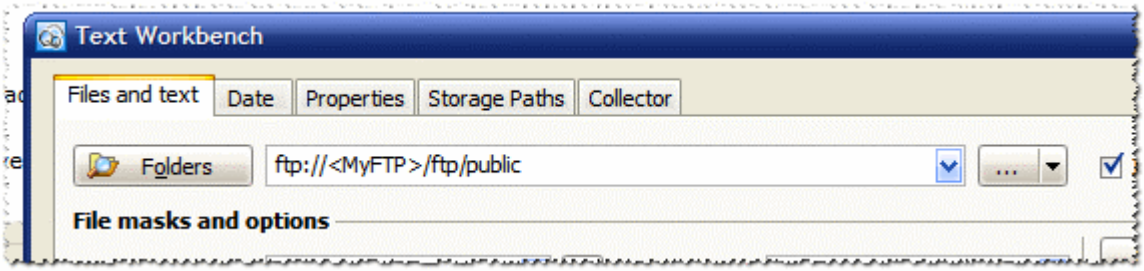


Here, *ftp://mywebsite.com/ftp/public* is the default location address.

Now, to use it you select it in the menu, just like this:



which inserts the alias-based path to the **MyFTP** server in the **Folders** text box (tip: hold **Ctrl** down to add the location to existing folders):



So, what will happen when we click **Search** or **Replace**?

Text Workbench will find the **MyFTP** location record and connect to the **MyFTP**'s server address (*mywebsite.com* as shown above) using the **MyFTP**'s username and password. Then, it will look for files in */ftp/public* on that server.

So, *ftp://<MyFTP>/ftp/public* will effectively become *ftp://mywebsite.com/ftp/public*.

Now let's go further. What will happen if we remove */public* from the path? It's simple: *ftp://<MyFTP>/ftp* will become *ftp://mywebsite.com/ftp*.

You can even change the path path completely. For example, the path *ftp://<MyFTP>/my_folder* will be mapped to *ftp://mywebsite.com/my_folder*. This is exactly what *overriding the path* means.

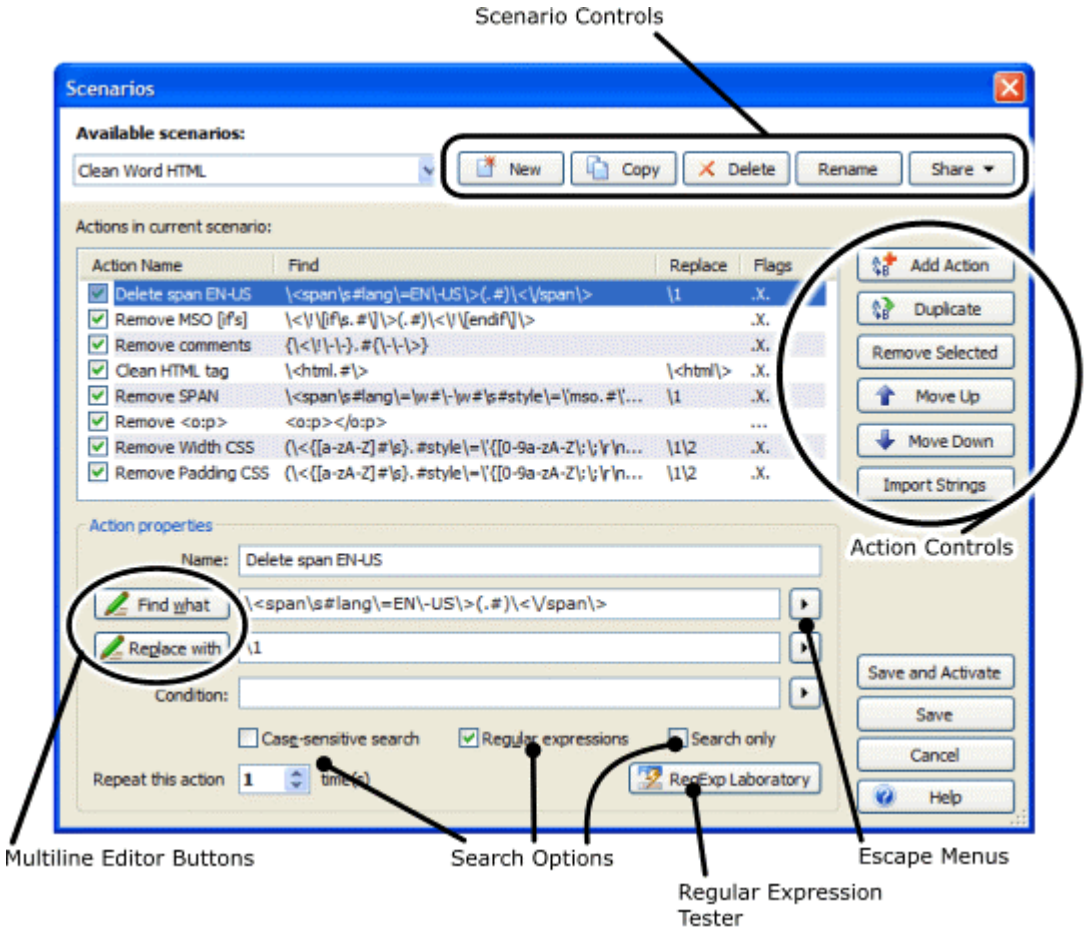
And finally, there is a thing which acts as a failsafe and a shortcut at the same time. If you delete the path in the **Folders** box, the full location URL you have specified for this location will be used instead. So, *ftp://<MyFTP>* will be generously mapped to *ftp://mywebsite.com/ftp/public*.

Chapter 2.
Scenarios, scripts and batch replace jobs

SCENARIO EDITOR DIALOG

Very often the search and replace operation requires that you replace different multiple strings in a file or files. You can do it by specifying the search and replace text consequently. This is a monotonous annoying routine. **Text Workbench** offers a neat solution to this problem - the **Scenarios**.

A **Scenario** is a set of search and replace text and conditions applied consequently to a file or files. Such a set of text and conditions is referred to as an **Action**. You can create a scenario using the **Scenarios** dialog.



Creating Scenarios

Follow the steps below to create and use a scenario.

1. Click the **New** button to create a scenario. In the **Name** dialog box, provide a name for the scenario.
2. Click the **Add Action** button to create a new action.
3. Specify the sensible name for the action. This is essential if you want to edit your scenario some time later.
4. Specify text and options for the action.
5. Repeat steps 2 - 5 to create more actions.

6. Click either **Save and Activate** or **Save** button. If you click **Save and Activate**, the new scenario will be activated.

Dialog Fields

The tables below describe the dialog fields.

Available Scenarios Group

Field	Description
Scenario List	A drop-down list containing the existing scenarios. Select the scenario by clicking on a drop-down arrow.
New	Click to create a new scenario.
Copy	Click to create a new scenario containing the same set of commands as the selected scenario; that is, *copy* the selected scenario.
Delete	Click to delete the selected scenario.
Rename	Click to assign a new name to a scenario.
Share	This feature explores the teamwork functions. Clicking this button will display a menu with the following commands: ▪ **Import Scenarios** - opens the standard Windows file dialog, allowing you to select and import the previously exported scenario file (`.hfsc`), or create a new scenario from a text file. ▪ **Export Scenarios** - opens the Scenario Export Dialog, allowing you to select, export and optionally send the exported scenarios via the e-mail.

Scenario Actions Group

Field	Description
Action List	A list of actions in the current scenario. To display the action properties, simply select it. The action information includes its name, text to find, text to replace with and the options abbreviation (**c** - case, **X** - regular expressions, **s** - search only). Beside each action is a checkbox which can be used to enable or disable a respective action.
Add Action	Click to add a new action to the scenario.
Duplicate	Click to create a new action and copy the currently selected action into the created one.
Remove Selected	Click to remove the selected action from the selected scenario.
Move Up	Click to move the selected action upwards. Press and hold the button down to start continuously moving the action up.

Move Down	Click to move the selected action downwards. Press and hold the button down to start continuously moving the action down.
Import Strings	Opens a file selection dialog where you can select a **.txt** or **.csv** file whose content (search and replace strings) is to be added to the current scenario. After you selected a file, the Search and Replace Strings Import Dialog is displayed where you can specify the import rules.

Action Properties Group

Field	Description
Name	Name of the selected action.
Find What button	Click this button to open the **Multiline Editor** dialog. This dialog allows entering text with multiple lines in normal mode, converting line breaks to `\r` and `\n` as appropriate.
Find What text field	This field is used to enter the text to search for.
Replace With button	Same purpose as for **Find What** button, but applies to the replacement text.
Replace With text field	Same purpose as for **Find What** text field, but applies to the replacement text. **Note** Generally, it is not necessary to escape non-alphabetical symbols except backslash (****) in the **Replace With** field if replacing with the Regular Expressions, but doing so is *highly* recommended to avoid collisions.
Condition text field / button	Specifies the condition, which, if evaluates to **true**, lets a given action to be applied to a current file. A condition start with the dollar sign followed by the condition name and parameters supplied in brackets. The parameters must not include round brackets - **(** and **)**. You can use *one of* the following conditions. **Condition** / **Syntax** / **Description** Files matching mask / $FILE(*mask*) / Specifies that the current action will be applied only to files matching the specified mask. You can use wildcards (*****) in the mask. Regular expressions are not allowed here. Files NOT matching mask / $NOTFILE(*mask*) / Specifies that the current action will be applied only to files that *do not* match the specified mask. You can use wildcards (*****) in the mask. Regular expressions are not allowed here. The button opens a menu containing these conditions.
Control Escape	If you have the option **Regular Expressions** selected, shows a menu with regular expressions that you can select and insert.

buttons	If the **Regular Expressions** option is not selected, shows a menu with common control escapes.
Regular Expression Laboratory button	Opens the Regular Expression Laboratory window, where you can test your regular expressions.
Case-sensitive search	Checking this button allows restrict your searches to those phrases that match the sought text exactly. If using Regular Expressions, you can control case sensitivity with the regular expression switches. See more information on Regular Expressions.
Regular expressions	Check this button to allow searches with Regular Expressions.
Search only	If you check this button, no replacements will occur even if the sought text is found. Useful for simply locating files with the required text.
Repeat this action	Specify here how many times the selected action will iterate. Allows to replace the repetitive blocks of text. The default repetition count is 1.

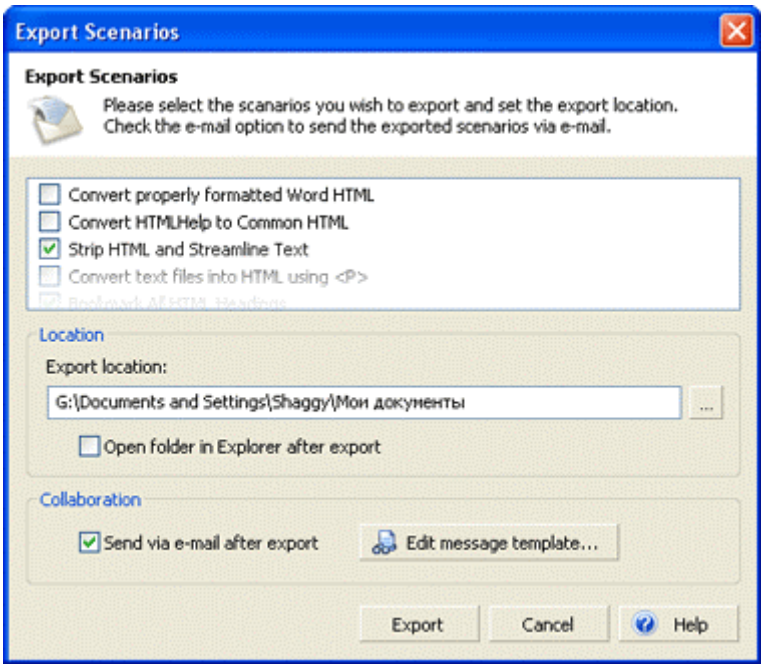
Common Controls

Control	Description
Save and Activate	Saves changes and sets the current scenario active in the main application window.
Save	Saves changes and closes the dialog.
Cancel	Closes the dialog discarding changes.
Help	Shows this help.

SCENARIO EXPORT DIALOG

The **Scenario Export Dialog** offers the user interface for exporting Text Workbench scenarios, which enables you with the teamwork options.

Each of the scenario is exported in an individual file in the user-supplied location.



Exporting Scenarios

Follow the steps below to export a scenario (or a set of scenarios).

1. In the Scenario Editor Dialog, click the **Share** button and select the **Export Scenarios** menu item.
2. In the **Scenario Export Dialog**, select the scenarios you want to export by checking the corresponding scenario boxes.
3. Specify the folder in which you want to store the exported scenarios. If you want to view the folder in **Windows Explorer** after the export, check the **Open folder...** button.
4. If you want to send the exported scenarios to your colleagues right after the export is done, check the **Send...** option. This will open the standard message creation window of your default e-mail client, with the exported files attached.
5. Click the **Export** button.

Dialog Fields

The tables below describe the dialog fields.

Location

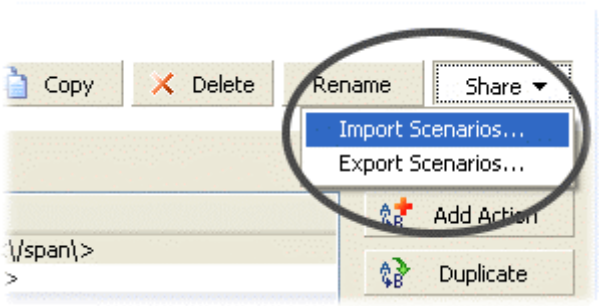
Field	Description
Export location	A directory that will contain the exported scenarios.
Open folder in Explorer after export	Check this option if you want to open the folder in **Windows Explorer** after the export.

Collaboration

Field	Description
Send via e-mail after export	Check this option if you want to send the exported scenarios to your colleagues right after the export is done.
Edit message template... button	This option allows you to save time while preparing your message to send. Click this button to open an editor window, where you can provide your custom e-mail text that is preserved between sessions and will be used for creating new messages later. The message can contain macro `%%files%%` that is replaced with the actual file names of the exported scenarios.

IMPORTING AND EXPORTING SCENARIOS AND SCRIPTS

You can easily share scenarios with your colleagues and team members. To export and import your scenarios, click the **Share** button in the **Scenario** editor:



Export

To export one or more of the scenarios, select **Export** in the menu. This will open the Scenario Editor Dialog in which you can choose scenarios to export and the destination location.

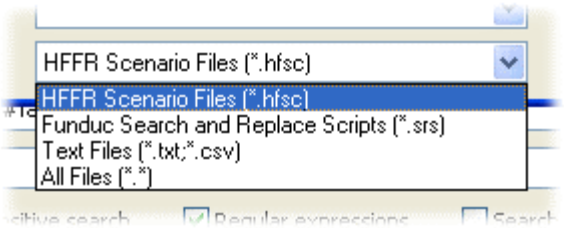
Import

To import a scenario, select **Import** in the menu. This will bring the standard Windows file dialog in which you can select a scenario that you want to import.

The **Text Workbench** can import the following batch replacement jobs:

- Text Workbench scenario files in the own proprietary format (**.hfsc** files);
- Funduc Software's Search And Replace scripts (**.srs** files);
- Common text files with search and replace strings delimited with user-defined separator (**.txt**) or CSV (comma-separated) files (**.csv**).

To select the desired format of imported files, use the file type selector:



To add a selected file to the scenario library, simply click **Open**. If you select to import a **txt** or **csv** file, the Search and Replace Strings Import Dialog is displayed where you have to set the import rules.

SEARCH AND REPLACE STRINGS IMPORT DIALOG

This dialog enables you to import text from an existing text file in the scenario. The file containing search and replace text is expected to contain pairs of search and replace text strings, separated with a user-defined symbol.

This dialog is displayed if you:

- click **Share** in the Scenario Editor Dialog, select **Import...** and then select a **.txt** or **.csv** file;
- click **Import Strings** in the Scenario Editor and select a **.txt** or **.csv** file.

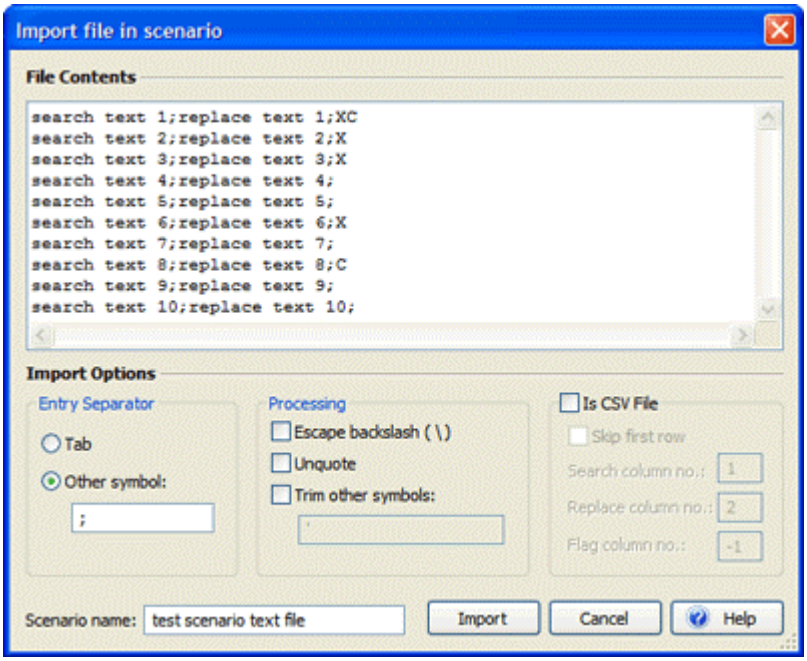
File Format

The programme can import strings from a plain text file, or from a CSV file. Depending on the file type, the import algorithm performs differently.

> **Note!**
> In fact, virtually any text file can be processed as a CSV file.

Text files

Text file mode is active when the option **IsCSVFile** is **off**. The figure below shows the import dialog in the text mode.



A file is scanned line by line. Each time a new text line is obtained, it is split in sections at positions where *separator* symbols are found. Each text line can contain up to 3 separators, thus it can be split in 3 sections. For example:

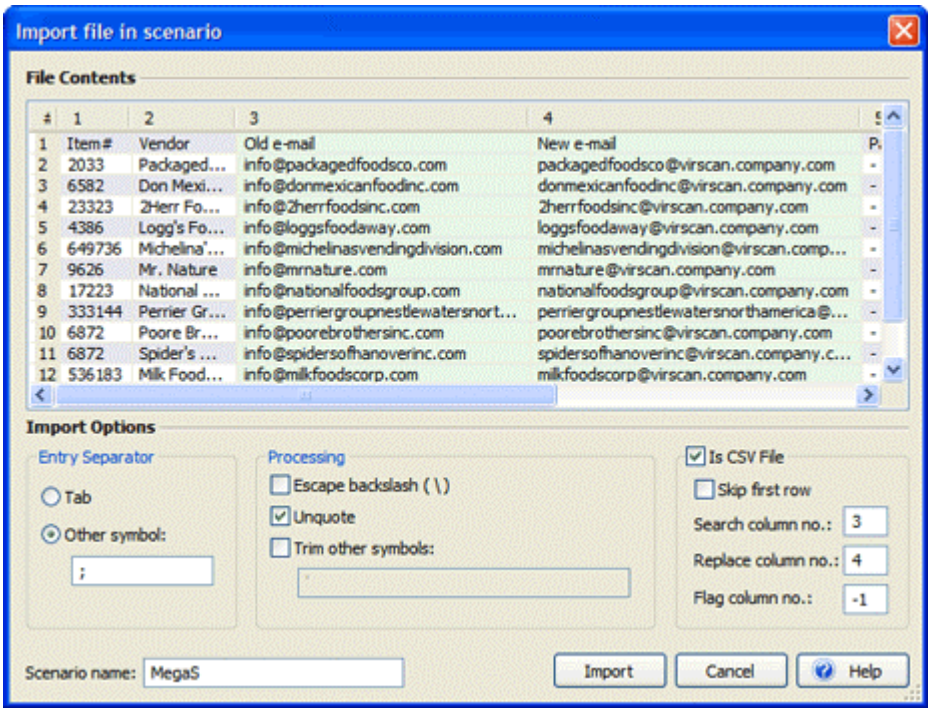
```
search text 1;replace text 1[;XC]
search text 2;replace text 2[;X]
search text 3;replace text 3[;X]
search text 4;replace text 4
search text 8;replace text 8[;C]

[...] - optional
```

The following table illustrates the sections.

Section 1	Section 2	Section 3
Search string.	Replace string.	Flags (optional). If present, specifies which flags of this action are to be switched on. The following flags can be used: ▪ **X**,**x** - use regular expressions. ▪ **C**,**c** - case sensitive search.

CSV files

CSV file mode is active when the option **IsCSVFile** is **on**:



With CSV files, you have to specify numbers of columns in which each of the three parameters reside: search string, replace string and flags (optional). Consider the following CSV file:

```
Order No;Part No;Vendor;Old Address;New Address;Contact Person;Misc
order;part;vendor;old address;new address;person;C
order;part;vendor;old address;new address;person;
order;part;vendor;old address;new address;person;X
```

Say we want to create a scenario to replace all old addresses with the new ones, and also regard possible flags that are in the column "Misc". We can see the following disposition in the file:

- old address column is **4**;
- new address column is **5**;
- flag colums is **7**. Flags have the same meaning as with text files.

Type these numbers in the corresponding fields, and - for this sample - set the separator to "**;**" and enable the **Skip first row** option.

Dialog Fields

The table below describes the dialog fields.

Field	**Description**
File contents	Displays the contents of the selected file. Examining the file structure allows you to properly set the import parameters.
Entry Separator *These options also affect CSV file processing.*	
Tab *	If selected, specifies the search and replace strings are separated with tab symbols in the selected file.
Other symbol *	Select this option to provide any separator symbol other than a tab.
Processing *These options also affect CSV file processing.*	
Escape backslash *	If selected, all the backslash symbols in the search and replace strings will be escaped. That is, a standalone symbol **** will become ****.
Unquote *	Check this option to remove any leading and trailing double quotes (**"**) from both search and replace strings.
Trim other symbols *	If selected, the import routine removes all the specified symbols from the start and the end of both search and replace strings.
CSV options	
Is CSV file	If this option is selected, the import routine treats the file as a CSV (comma-separated) file and splits it in numbered columns. In the CSV mode, the text view changes to the table view. For CSV files, you can temporarily turn this option off to view the file structure (separator etc.). **Note!** Only valid and properly formatted CSV files are guaranteed to be processed correctly.
Skip first row	If checked, the first row of a CSV file will not be added to a scenario.
Search column no. *	Specifies ordinal of a column containing the search strings. Columns start from 1.
Replace column no. *	Specifies ordinal of a column containing the replacement strings. Columns start from 1.

Flag column no. *	Specifies ordinal of a column containing the action flags. Columns start from 1. This field is optional. Set it to 0 or -1 or leave empty to not use flags.
Scenario name	Specifies the name of a new scenario created from the supplied file. These fields are only displayed if you import a new scenario. If you are adding strings to an existing scenario, the fields are not displayed.

* - changing these options in the CSV mode incurs the update of the table view after an action with a 1.5 second delay.

Chapter 3.
Regular Expressions

REGULAR EXPRESSIONS

This topic contains information on regular expression control switches and operators, as well as important notes on using regular expressions.

IMPORTANT!
Non-alphanumeric symbols in search expressions **must** be escaped with a backslash. For example, if you need to find a symbol # using regular expressions, you must type \#. Otherwise, symbols are treated as regular expression operators.

Note:
Literal switches are case-sensitive! For example, `\s` matches whitespace, while `\S` matches non-whitespace.

Repeat qualifiers (`@`, `#`, `*`, `+`) can be applied (for examlpe: `.+`) to match text segments as large as 2 Mb or less.

Note:
Greedy means matching this condition as much as possible, disregarding further coinciding or alike conditions. See more detailed description of the greedy and non-greedy modes in topic Using Regular Expressions.

Expressions for searching

Expression	Description.
^	Match the beginning of file. **Text Workbench** searches the file as if it is a single line. This allows to search for the line-breaking characters and text blocks, manipulate and replace them without any hassle. To insert any text in the beginning of a file, type: **^.** in the **Find What** field and **\0*text_to_insert*** in the **Replace With** field.
$	Match the end of file.
.	Match any character.
[ ]	Match characters in set. Specify set (e.g. [aghet2]) or range (e.g. [A-Z]).
[^]	Match characters not in set (e.g. [^A-Z]).
?	Match previous pattern 0 or 1 times (greedy).
|	Match previous or next pattern.
@	Match previous pattern 0 or more times (non-greedy).
#	Match previous pattern 1 or more times (non-greedy).
*	Match previous pattern 0 or more times (greedy).
+	Match previous pattern 1 or more times (greedy).

`{ }`	Group characters to form one pattern.
`( )`	Group and remember for further referencing and use.
`\`	Quote next character (only of not a-z; e.g. "\>" designates symbol ">").
`<`	Match beginning of a word.
`>`	Match end of a word.
`\t`	Match 0x09 (tab).
`\e`	Match escape (^E).
`\s`	Match whitespace (tab, space, CR, LF). More...
`\S`	Match non-whitespace.
`\w`	Match word character.
`\W`	Match non-word character.
`\d`	Match digit character.
`\D`	Match non-digit character.
`\U`	Match uppercase.
`\L`	Match lowercase.
`\C`	Match case sensitively from here on.
`\c`	Match ignoring the case from here on.
\x*NN* (ANSI) or \x*NNNN* (Unicode)	▪ **ANSI files**: Single byte character with hexadecimal value of NN, where N=[0-9A-F]. ▪ **UTF-8 and Unicode files**: Double byte character with hexadecimal value of NNNN, where N=[0-9A-F]. ▪ **Regular Expression Laboratory**: use Unicode notation (\x*NNNN*).
\d*NNN* (ANSI only)	Character with decimal value of NNN, where N=[0-9].
\o*NN* (ANSI only)	Character with octal value of NNN, where N=[0-7].

Expressions for replacing

Expression	**Description**
`\0`	Place whole found text.
`\#`	Place contents of the stored group numbered by the "#" [groups are defined by braces **(...)** in the search expression]; #=[1-9].
`\t`	Place tab character.
`\u`	Make next character uppercase.

`\l`	Make next character lowercase.
`\U`	Force further output uppercase.
`\L`	Force further output lowercase.
`\E`	Turn off case transformation (after the use of \U or \L)
`\f`	Insert current file name and extension (`file.ext`).
`\F`	Insert current file name only, without extension (`file`).
`\X`	Insert current file extension (`.ext`). More...
`\R`	Insert a random 8-digit number. For example: `267124D2`.

`\S`	Insert a random 8-digit number that remains con**s**tant within a single replace expression.
	
	For example, the expression
	`id="\S" onClick="openPageId('\S')"`
	might insert
	`id="267124D2" onClick="openPageId('267124D2')"`

`\Zhe`	Converts the next symbol to numeric HTML entity.

`\ZhE`	Turns on the **ASCII - to - HTML Entity** conversion. Makes all further output converted to numeric HTML entities. For example, the operator
	`\ZhEMyString`
	will insert
	`MyString`

`\Zhx`	Turns off the **ASCII - to - HTML Entity** conversion previously initialised by `\ZhE`.
`\Zi:"`*`file_path_name`*`"`	Inserts contents of the specified file. *`file_path_name`* must be a fully qualified absolute path name. For example, the command `\Zi:"c:\MyWeb\style.css"` will insert contents of the file **c:\MyWeb\style.css**.
`\P[h]:"`*`file_path_name`*`"`	Executes the path name resolver. Very often your files (for example, HTML pages) are stored in different folders and use reference to one file in some other folder (for example, CSS file). This operator allows to insert a path name of the referenced file relative to a processed file, given a fully-qualified path and name of the referenced file. `\P` operator uses backslash as the path part separator. `\Ph` operator (h stands for *HTML mode*) uses forward slash. If the given path name cannot be converted to a relative path, the provided path name is inserted. See **Remarks** for example.

`\Pc[h]:"directory_path"`	Similar to operator `\P`. Executes the path name resolver on the currently processed file. Inserts path to the current file relative to the provided directory. For example, if the current file is `c:\dir1\dir2\dir3\file.ext` and the *`directory_path`* is `c:\dir1\dir2` , the operator `\Pc:"c:\dir1\dir2"` will insert `.\Dir3\file.ext` `\Pc` operator uses backslash as the path part separator. `\Pch` operator (h stands for *HTML mode*) uses forward slash. If conversion fails, the current file path name is inserted.
`\Pd[h]:"ref_dir_path"`	Executes the reverse path name resolver on the directory of the currently processed file. Inserts path to the provided directory relative to the directory of the current file. For example, if the current file directory is `c:\dir1\dir2\dir3` and the *`ref_dir_path`* is `c:\dir1` , the operator `\Pd:"c:\dir1"` will insert `..\..` `\Pd` operator uses backslash as the path part separator. `\Pdh` operator (h stands for *HTML mode*) uses forward slash. If conversion fails, the current file path is inserted. In all cases, *no* trailing slash is appended.
`\Ppf`	Inserts the name of the current file parent folder. For example, if the current file is `c:\dir1\dir2\dir3\file.ext`, the operator `\Ppf` will insert `dir3` No leading or trailing slashes are added.

`\Ppp`
 	
or
 	
`\Ppp[first,length]`
	Inserts the string which is the path to the current file.
	
	Syntax 1
	
	The first syntax inserts the whole path.
	
	For example, if the current file is `c:\dir1\dir2\dir3\file.ext`, the operator `\Ppp` will insert `c:\dir1\dir2\dir3`
	
	No trailing slash is appended.
	
	Syntax 2
	
	The second syntax inserts only a part of the path given the zero-based index of the first symbol, and the section length.

<table>
<tr><td></td><td>The table below illustrates the result of different calls for the current file <code>c:\dir1\dir2\dir3\file.ext</code>:

<table><tr><th>Operator</th><th>Inserts</th></tr><tr><td>\Ppp[0,7]</td><td>c:\dir1</td></tr><tr><td>\Ppp[3,9]</td><td>dir1\dir2</td></tr><tr><td>\Ppp[3,]</td><td>dir1\dir2\dir3</td></tr></table>
To insert symbols [and] right after \Ppp, use the following syntax: \Ppp\[(or \Ppp\]).</td></tr>
<tr><td>\xNN</td><td>Insert character with hexadecimal value of NN, where N=[0-9A-F].</td></tr>
<tr><td>\dNNN</td><td>Insert character with decimal value of NNN, where N=[0-9].</td></tr>
<tr><td>\oNN</td><td>Insert character with octal value of NNN, where N=[0-7].</td></tr>
</table>

Remarks

Example of using the path name resolve operator

For example, you are processing files and folders in C:\MyWeb\HTML\Catalog. You want to insert a reference to a CSS file style.css stored in the folder C:\MyWeb\HTML. You could write the following replace expression:

```
<LINK REL=stylesheet HREF=\"\Ph:"C:\\MyWeb\\HTML\\style.css"\"
TYPE="text/css">
```

This will insert the following text in HTML files in C:\MyWeb\HTML\Catalog:

```
<LINK REL=stylesheet HREF="../style.css" TYPE="text/css">
```

and the following text in HTML files in C:\MyWeb\HTML\Catalog\dir1\dir2:

```
<LINK REL=stylesheet HREF="../../../style.css" TYPE="text/css">
```

Note!
Root parts of path names of both a referenced and processed file must be the same. It means that they must have a common drive letter.

USING REGULAR EXPRESSIONS

What are Regular Expressions?

Regular expressions are a way to search for substrings ("matches") in strings. This is done by searching with "patterns" through the string.

You probably know the '*' and '?' charachters used in the `dir` command on the DOS command line. The '*' character means "zero or more arbitrary characters" and the '?' means "one arbitrary character".

When using a pattern like "`text?.*`", it will find files like

- `textf.txt`
- `text1.asp`
- `text9.html`

But it will not find files like

- `text.txt`
- `text.asp`
- `text.html`

This is exactly the way regular expressions work. While the '*' and '?' are a very limited subset of patterns, regular expressions supply a much broader spectrum of describing patterns.

Example usages could be:

- remove all occurrences of a specific tag from an html file;
- check whether an e-mail address is well-formed;
- replace value of some tag attribute with a different one;
- many more other tasks.

Any operator or set of operators represent a *pattern*.

You will probably need to match some patterns containing symbols that may differ and vary in some way. For example, you want to find words starting with `tom` and having four characters in length. The operator that matches any character is dot (**.**). Thus, the following pattern would match all these words: `tom.`

This example will also find text like `tom.`, `tom>`, `tom!`, etc.

To prevent the pattern `tom.` from matching not meaningful phrases, we should narrow the search criteria to only alphabetic symbols. This can be done using character sets. A set is specified with square brackets. Sets may include individual symbols and ranges. For example, the following set will match any one symbol of a, t, z and 8: `[atz8]`. And this set will match all lowercase letters:

`[a-z]`.

Thus, to limit the previous example to meaningful phrases, we could write a pattern: `tom[a-z]`.

Sometimes you need to find all symbols except some. Writing a large set including all possible symbols is ineffective. So we better use a negation operator in a set: `^`. For example, the following set will match any one symbol except @: `[^\@]`. Please note that the symbol @ is escaped as it is not alphanumeric.

Regular expressions would be of no use unless they might match any text of any length. To achieve this, repetition qualifiers were introduced, which allows matching nearly any text.

In the previous example, a pattern `tom[a-z]` would successfully find any word of four symbols in length except `tom` itself. To force the pattern to match `tom`, we should instruct it to do so. The qualifier `?` tells to match the preceding pattern 0 or 1 times. The following pattern will match `tom` as well: `tom[a-z]?`

Before we proceed with the other repetition qualifiers, we should understand one important thing about repetition modes.

Imagine a text that contains some occurrences of a character. For example, `one, two, three, four.` This text has 3 entries of a comma. Now we want to instruct the regular expression engine to "**match all characters but stop before a comma**".

A greedy mode will match *all* characters and stop before the **last** comma:

one, two, three, four.

A non-greedy mode will match *all* characters and stop before the **first** comma:

one, two, three, four.

Let us extend the previous example by introducing a new condition: match all text starting from `tom` but ending with full-stop. So we need to:

1. match `tom`;
2. match any character;
3. repeat the preceding condition 0 or more times until the first occurrence of the next match (4) is found;
4. match a full-stop (a dot).

The following table shows the corresponding operators:

Part	**Operator**	**Comment**
Match `tom`	tom	A simple text

Match any character	.	A dot-operator
Repeat the preceding condition until the first occurrence of the next match is found	@	Repeat qualifier: *Match previous pattern 0 or more times (non-greedy).*
Match a full-stop (a dot)	\.	A dot. Escape is added to instruct to process the dot as a common symbol, not operator.

Thus, the pattern would look like:

```
tom.@\.
```

Say you need to find one of the words: *macrocoding* and *macrocode*. There are several ways to do that. For example, we can split each word into **macrocod**+**ing** and **macrocod**+**e**. Now, we will need a pattern that would:

- match `macrocod`;
- match either `ing` or `e`.

When we say "or", we say "or". When a regular expression says "or", it says "|". Armed with this knowledge, we write: `macrocoding|e`.

Looks rather meaningless, doesn't it? What would this expression do: match `macrocoding or e` or match `macrocodin and g or e` ? That's why a *pattern operator* had been developed.

A *pattern operator* concatenates several stand-alone symbols or patterns to form one pattern. For example, a single symbol `e` is a pattern. The first symbol (`i`) in the "`ing`" is a stand-alone pattern. To form a single pattern from "`ing`", we should enclose it in braces:

```
{ing}
```

Now, `ing` is a single pattern.

This allows us to write the following pattern:

```
{macrocod{ing}|{e}}
```

This is a correct well-formed single pattern.

In terms of semantics, expression and pattern operators are the same. The difference is that the text that matches the expression is stored and can be referenced further, for example, when replacing.

For example, we could alter the previous example to make an expression out of the ending `{ing}|{e}` by enclosing it in the round braces:

```
{macrocod({ing}|{e})}
```

Now we can reference the ending with the operator `\1`. 1 stands for the number of the expression. We can write the replace pattern that would insert a plus sign between `macrocod` and the ending:

```
macrocod+\1
```

EXAMPLES OF REGULAR EXPRESSIONS

Below are given some examples of using regular expressions in various situations.

Find any HTML colour constant.

Find pattern	`\#{[0-9A-Fa-f]}+`

Replace all HTML colour constants with the new one (#AFEEEE).

Find pattern	`\#{[0-9A-Fa-f]}+`
Replace pattern	`#AFEEEE`

Replace all occurences of full stop before the lowercase letter with comma, truncating separating spaces to only one (useful for scanned documents).

Find pattern	`{\.\s#}(\L[a-z])`
Replace pattern	`\, \1`

Strip HTML tags.

Find pattern	`{\<}{\/?}{.#}{\>}`
Replace pattern	`empty`

Change the value of all "background-color:" CSS properties everywhere.

Find pattern	`(background\-color\:){\#{[0-9A-Fa-f]}+}`
Replace pattern	`\1#FFE4E1`

Replace <P> tags with <DIV> tags, preserving tag attributes.

Find pattern	`\<P\s(.#)\>(.#)\<\/P\>`
Replace pattern	`<DIV \1>\2</DIV>`

Colorize HTML tags for use inside HTML page: symbols with blue, tag with red, convert < and > to entities.

Find pattern	`\<({\/}?)(.#)\>`
Replace pattern	`<span style="color:blue"><\1</span><span style="color:red">\2</span><span style="color:blue">></span>`

Find host addresses.

The pattern would match the following:

- http://www.host.dom/
- http://host.dom/
- ftp://203.131.69.75/
- www.host.dom/

Find pattern	`({ht}|{f}tp\:\/\/)?{[0-9]#\.[0-9]#\.[0-9]#\.[0-9]#}|{{[^\.\/]#\.}?{[^\.\/]#\.[^\.\/]#}}\/`
Replace pattern	

Find the whole paragraph; store the paragraph contents.

Find pattern	`(.#){\r\n}|{{\r}|{\n}}`
Replace pattern	

REGULAR EXPRESSION LABORATORY

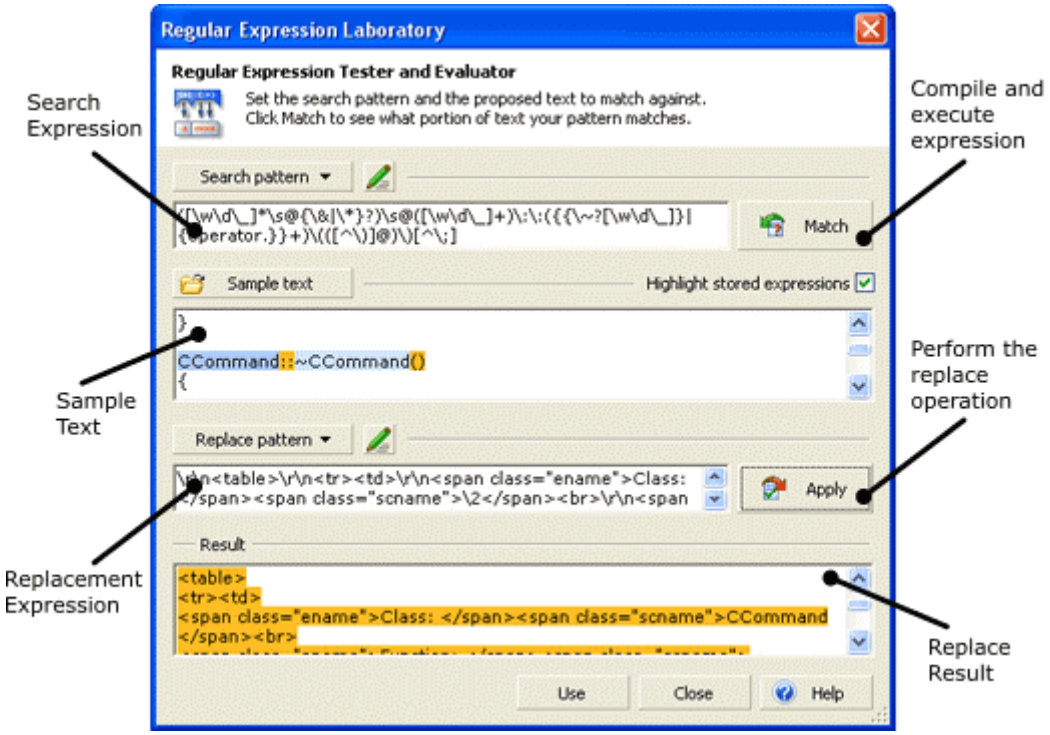
If you are not familiar with regular expressions and want to learn how to use them, or want to test your own expressions, the **Regular Expression Laboratory** will help you much. Its main intent is to visualise both the internals of the expression compilation process and how the prepared expression matches the text portions.

At the heart of the regular expression engine is a concept of a **finite-state automata**. You can treat this automata as a tiny virtual computer working by a program. The program that the automata runs is the precompiled binary form of an expression. In other words, the textual expression that you provide is first compiled into a binary form and then executed against a text.

This is why you get regular expression errors sometimes. Some applications may even crash if the regular expression faults are not handled properly. Using the **Regular Expression Laboratory**, you will easily get a robust working expression.

You can open the Laboratory by clicking button  in the Files and Text main window tab or Scenario Editor.

The following picture illustrates the **Regular Expression Laboratory** front-end.



How to Use the Laboratory

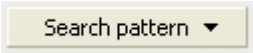
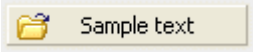
Using it is fairly simple. What is more, the Laboratory provides you with visual cues when you are mistaken with your expression. The below given sequence is

usually the best practice.

1. Type your text to operate on in the **Sample text** field. As an alternative, you can load the text from file by clicking the *button* **Sample text**.
2. Type your expression in the **Search pattern** field. For example, if you want to match all HTML tags, you could type `\<[^\>]#\>`
3. Click the **Match** button. This will compile the expression and find the first occurrence of the matching text in the **Sample text** field. If the compilation fails, the erratic search expression symbol will blink with red.
4. Type the replacement text in the **Replace pattern** box. For example, if you want to convert the HTML tag to lowercase: `\L\0`
5. Click the **Apply** button. The result text box will contain the result of the replace operation.

Dialog Fields

The tables below describe the dialog fields.

Field	Description
Search pattern text box	Contains the search expression.
Search pattern button 	Shows a menu with regular expressions that you can select and insert.
Multiline editor button 	Click this button to open the **Multiline Editor** dialog. This dialog allows entering text with multiple lines in normal mode, converting line breaks to `\r` and `\n` as appropriate.
Match button	Compiles the provided expression and tries to match it against the sample text. Click this button again to find the next match in the sample text. You can press and hold this button to iterate through the text.
Sample text box	Contains the sample text that the expression will match against.
Sample text button 	Allows to load the sample text from file.
Highlight stored expressions	If checked, the matching stored expressions, i.e. those enclosed in parentheses ( ) will have different background colour in the sample text box.

Replace pattern text box	Contains the replacement expression.
Replace pattern button  	Shows a menu with regular expressions that you can select and insert.
Multiline editor button 	Same purpose as for the **Multiline editor** button for the **Search pattern**.
Apply button	Given the replacement expression, performs the replace operation on the sample text.

Chapter 4.
Working via FTP

FTP: GENERAL INFORMATION

Introduction

Text Workbench keeps abreast of the times and respects your possible needs. Nowadays, we are quite used to the fact that the information we use is not stored at the local computer or home (corporate) network only, but remotely as well, at servers on the Internet.

Text Workbench (hereinafter called **TW**) can process remote files via the FTP protocol. This means that you can search for files that satisfy the specified criteria, find and replace text in files, rename remote files. You can even edit the found files in the seamless text editor, and all absolutely transparently - just as if you were processing files on your computer.

Requirements

Processing remote files requires that you have a **broadband permanent** Internet connection. This is not because **TW** is very fastidious about being modern. Being connected to the Internet continuously guarantees that you keep your files on an FTP server safe and sound. When processing (replacing text in) remote files, **TW**:

- downloads a file to a temporary local storage;
- examines it;
- if you select to replace some text in the file, performs the required operations;
- uploads the file (if it was modified).

The last operation (uploading) is the weakest point in the whole process: if uploading gets interrupted due to lost of connectivity (e.g. telephone connection breaks), the remote file becomes corrupted (cut off). If you have a dial-up connection, the described scenario is rare but possible. If you have a permanent connection via a dedicated line or using the DSL technology, this can almost never happen.

However, **TW** can still work via dial-up modem connections. It will ask you to connect to the Internet if you are not connected, and will perform the prescribed operations. Besides information integrity, the processing (downloading and uploading) speed is the major issue in this case. If you use a dial-up connection, we advise you to download required files to your computer first, process them with **TW**, and then upload them to the server. You can download and upload files using your favourite FTP client.

Limitations

The **Modify read-only files** option has no effect. **TW** cannot change file permissions.

Another possible issue is, when renaming remote files (using the **Application mode** button), your FTP server may reject suggested file names finding them

invalid. This depends on the remote server type, operating system and software installed on the server.

When processing, if an FTP server returns extended error codes (those wrapped in textual description), and the whole extended FTP error description cannot be displayed in a single report line properly, **TW** reports that the file cannot be found.

Relative target locations are not allowed with FTP folders (see the Storage Folders Tab topic for more information on storage modes).

Specifying the FTP folder path. Syntax and security considerations

There are two ways to specify the FTP server and path.

Way 1: Quick and dirty (an explicit URL)

Type the FTP location URL in the **Folders** text box. Specify a user name and a password that are used to connect to a server, *in the open form*. The whole remote path (URL) that you have to provide in the **Folders** field has the following format:

```
ftp://username:password@server[/folder/folder/....]
```

For example:

```
ftp://jeffrey:qt98267@jeffreyandco.com/wwwroot
```

This form is canonical and used widely (e.g. in the Internet Explorer and Windows Explorer).

If you want to access the server anonymously, simply omit the user name, password and separators. For example:

```
ftp://ftp.jeffreyandco.com
```

The search report displays server names without a user name or password.

Way 2: Do as good people do

However simple and quick, the quick method has a number of obvious disadvantages: it is insecure and not comprehensive. The best way is to create FTP accounts using the FTP Locations dialog and then specify the FTP paths by selecting one from the menu.

Temporary storage and back-up copies

When processing remote files, **TW** creates a special folder on your computer, where it stores the downloaded files and creates back-up copies of modified files.

The temporary storage folder root path is formed as follows:

```
<Windows temp directory>\HFFR.TW.FTP.Local\
```

After you have performed **at least one** search on a FTP server, you can open this folder in the Windows Explorer by typing the following path in the address field and pressing **Enter**:

```
%temp%\HFFR.TW.FTP.Local
```

In this folder, **TW** creates folders for each server (*server-specific folders*):

```
<Windows temp directory>\HFFR.TW.FTP.Local\server name\
```

TW downloads files from a server and creates their back-up copies in a server-specific folder.

THE ROBUST WAY: USING FTP LOCATIONS

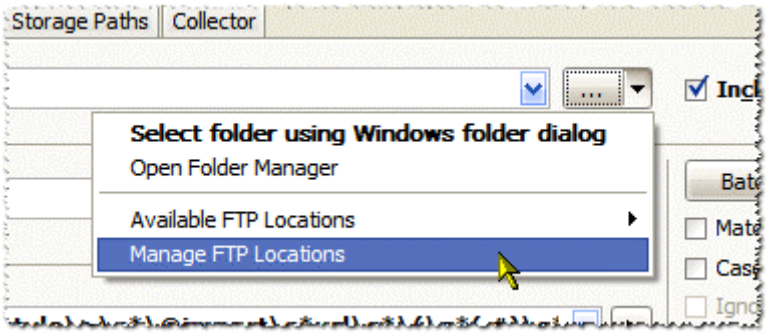
The best way to process remote files on an FTP server is to create FTP accounts using the FTP Locations dialog and then specify a required FTP path by selecting one from the menu.

You will need to have the following data at hand to create an FTP location record:

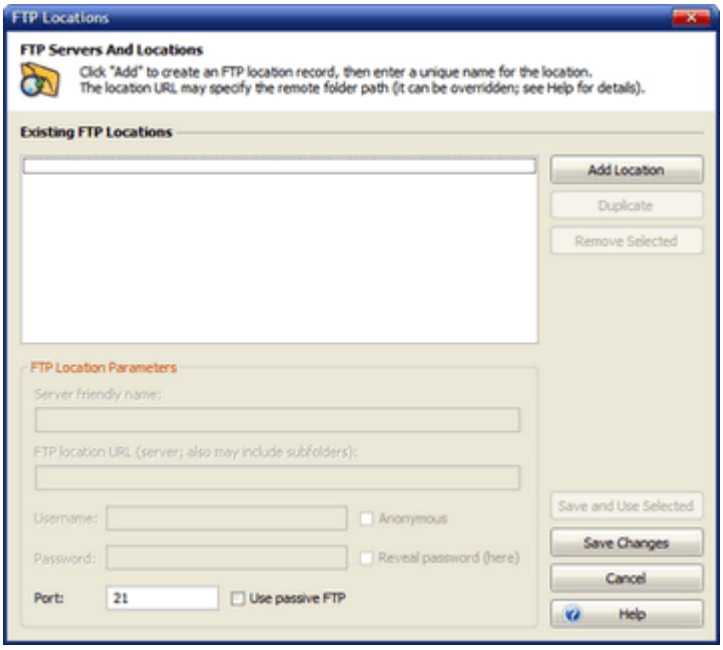
- the remote server address (always required);
- the username and the password with which you want to access the remote server;
- the path to the folder in which you want to look for files.

So, let's get started.

1. Click on the drop-down arrow of the **Folder Functions** button and select **Manage FTP Locations** in the menu:



This will bring up the FTP Locations dialog box:



It's still lonely and deserted - let's breathe new life into it!

2. Click **Add Location** button. Now there is a new location with the default name. Change the name to something meaningful that will help you recognize this FTP location anytime later... even when you grow old. Perhaps that's too much, but you've got the idea.
3. Now specify the FTP server data.

   First, enter the server address in the **FTP location URL** field. It may or may not include the *ftp://* prefix - it does not matter. The server address may include the full path to a remote folder. In other words, your location address may look something like this:

```
mywork.com
ftp://mywork.com
ftp://mywork.com/public/john
mywork.com/public/john
```

   Using paths in server URL's is described in detail in Notes on using FTP paths.

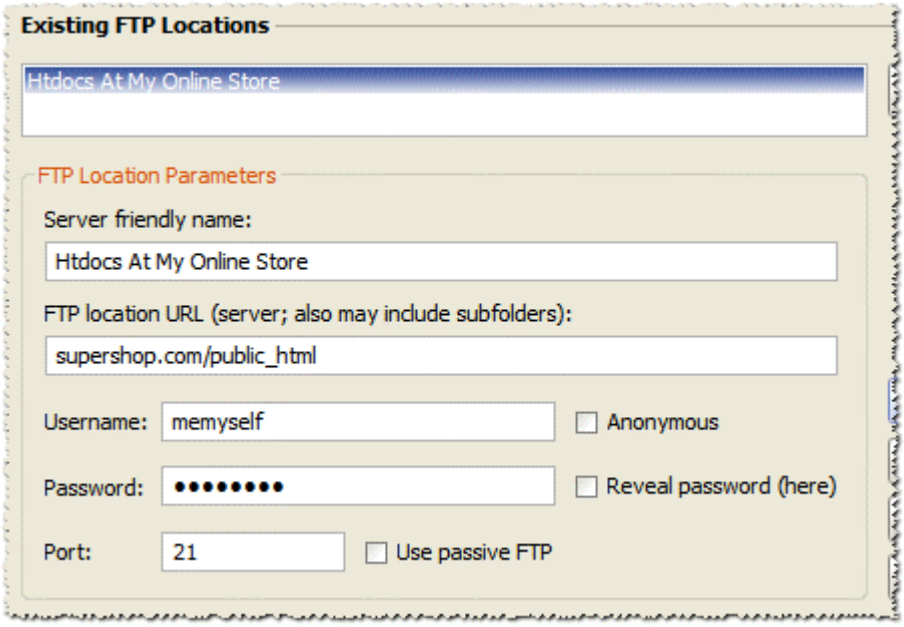
4. Then, type in your username and password.

   If your password is good and strong, it is obviously long. To avoid error, you may want to use the **Reveal password** option: it will show the password characters - in this dialog box only!

   If you're accessing a public server requiring no authentication, tick the **Anonymous** option.

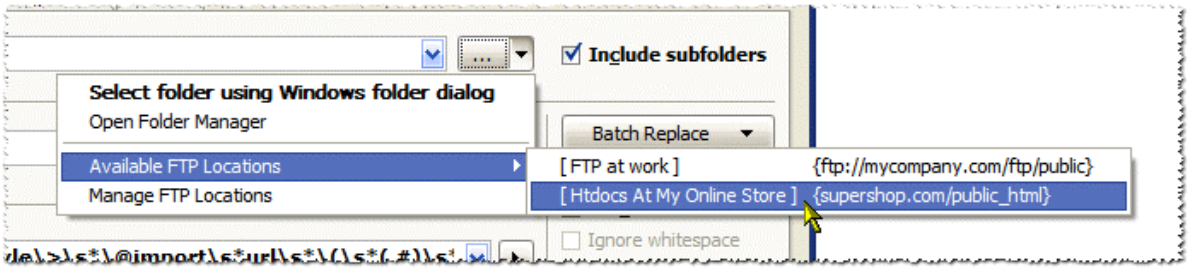
   Other options (port and passive FTP) are described here.

5. Your first FTP location is now ready to use. The parameters sheet may look something like this:

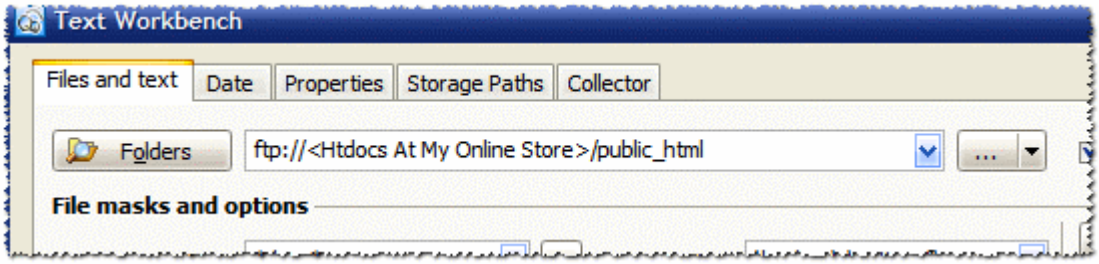


Now, to use it, you have two options. The first is fairly obvious: click **Save and Use Selected**. We'll exercise another way though, because you'll make much more use of it in future.

6. Click **Save Changes**. Now click on the drop-down arrow of the **Folder Functions** button again and open the **Available FTP Locations** submenu. Select your new location in it:



...and take a look at the **Folders** text box: it now contains the FTP location path:



Essentially, that's all you have to do to manage and use the FTP locations. Now you can just click **Search** or **Replace**.

Don't forget to read the topic on using the FTP Locations dialog box: it contains many useful tips!

USING FTP THE QUICK AND DIRTY WAY: EXPLICIT URL'S

Even if not using FTP Locations to define and use FTP paths, you have to specify almost the same information in the **Folders** text box:

- the remote server name;
- the user name and the password with which you want to access the remote server;
- the path to the folder in which you want to look for files.

The aggregate of the above entities forms an URL (*uniform resource locator*) - in our case, this is an FTP URL. We use FTP URL's to tell **Text Workbench** where to search and process remote files.

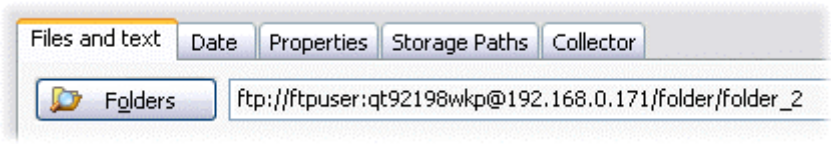
Text Workbench accepts the common canonical URL format, which looks as follows:

```
ftp://username:password@server[/folder/folder/....]
```

In this URL:

- `ftp://` is the protocol name (FTP);
- *`username:password`* are the user's user name and password separated with semicolon. If you access your FTP server anonymously, ignore the whole section *username:password@*;
- `server` is the remote server name, either a domain name (e.g. **server.com**) or an IP address (e.g. **127.0.0.1**);
- `folder` parts form the path to the search directory on the server. If you omit the path, the root directory of the server will be searched.

The full FTP URL is to be entered in the **Folders** field of the **Text and Files** tab, just like names of local folders:



Specify other search criteria. You will find many examples of different tasks in the **Common Tasks** and **Knowledge Base** sections of this help system.

In fact, this is all that you need to do to start processing files on a remote FTP server. After you click **Search** or **Replace**, **Text Workbench** starts processing remote files according to the specified rules and options.

Upon completion, you can click any found file to view it. The file will be downloaded from the server, thus ensuring the data is up-to-date. You can click **Edit** to load the file in the text editor. If you modify the file and click **Save** in the editor, the file is uploaded back to the server.

Chapter 5.
Command Line

COMMAND LINE

Text Workbench is equipped with a set of command line parameters which enable you to perform the find/replace tasks from within any batch file and/or external-tool-aware application.

All switches and their values are case-sensitive.

Please note that switches have no default values. The values that had been set during the last attended runtime session are considered default.

Switch	Description
File and Path Control	
`/folder:"`*`quoted_path`*`"`	The folder in which the operation will be performed. Quote the string if it has long names.
`/recurse:on|off`	Recurse subfolders.
`/type:"`*`quoted_type`*`"`	Type(s) (filters) of the files to search for.
`/romod:yes|no`	Modify read only files.
`/rorest:yes|no`	Restore read-only attribute.
`/retaintime:yes|no`	Tells to leave the times of the *modified* file as-is, without change.

`/bakmode:{d|s|p}:"`*`folder_or_path`*`"`	Back-up creation mode and path to the back-up folder.
	
	Modes (required)
	
	d - default;
	s - subfolder of the search folder;
	p - static path.
	
	Folder or path (required)
	
	If mode is d, folder name is ignored.
	If mode is s, name of the subfolder.
	If mode is p, path to the back-up folder; the path may include environment variables (*%variable_name%*).
	
	Command line path for `/bakmode` may not contain spaces. To overcome this limitation, declare any environment variable and specify it here with p mode.
	
	Examples
	
	`/bakmode:s:"backup"`
	`/bakmode:p:"%BackupFolder%"`

	These modes fully reflect the Storage Folders Tab options for the **Back-up Folder**.

`/tgtmode:{d|s|p}:"`*`folder_or_path`*`"`	Target (processed) file storage mode and path to the folder to store the processed files.
	
	Modes (required)
	
	d - default;
	s - subfolder of the search folder;
	p - static path.
	
	Folder or path (required)
	
	If mode is d, folder name is ignored.
	If mode is s, name of the subfolder.
	If mode is p, full path to the target folder; the path may include environment variables (*%variable_name%*).
	
	Command line path for `/tgtmode` may not contain spaces. To overcome this limitation, declare any environment variable and specify it here with p mode.
	
	Example
	
	`/tgtmode:s:"result"`
	`/tgtmode:p:"%TargetFolder%"`
	
	These modes fully reflect the Storage Folders Tab options for the **Folder for Processed Files**.

Text Control	
`/xwhat:"`*`quoted_string`*`"`	Find What. Quote the string if it contains space(s). To use the quotation marks inside this parameter, use double quotation mark for each instance. For example, the text `The "Good Ones"` must be represented as `/xwhat:"The ""Good Ones"""`.
`/xwith:"`*`quoted_string`*`"`	Replace With. Quote the string if it contains space(s). To use the quotation marks inside this parameter, use double quotation mark for each instance. For example, the text `The "Good Ones"` must be represented as `/xwith:"The ""Good Ones"""`.
`/case:on|off`	Case sensitivity.

`/rx:on|off`	Specifies to use (*on*) or not use (*off*) the Regular Expressions.
`/replace:yes|no`	If *yes*, specifies that the found text will be replaced.
`/scn:"`*`scenario_name`*`"`	Specifies the name of a scenario to execute. **Note** If the scenario name is correct, it will be activated upon start thus all other text control switches will have no effect if specified.
Report Control	
`/list:"`*`quoted_path_name`*`"`	Generate HTML listing.
Runtime Control	
`/cfg:"`*`configuration_file`*`"`	Name of the configuration file. **Note** This option erases and cancels *all* other options if they are specified on the command line. All options are presumed to be stored in the configuration file.
`/run`	Start automatically when launched.
`/close`	Close right after automatic run.
`/hidden`	Do not show the application window when running. This switch is only applicable if `/run` and `/close` switches are both specified.

CONFIGURATION FILE FOR USE IN COMMAND LINE

Instead of supplying a long tail of options on the command line, you can create and use the configuration file. The configuration file is a text file containing command-line options for use with **Text Workbench**.

Each option must reside within a single line of text. Options can start from the new line or can be merged into a single line. In the latter case, options must be separated with a space or tab.

The simplest way to create a configuration file is using the Configuration Export Dialog.

Example of the configuration file

```
File: c:\MyWeb\Projects\replace_span.hcfg

/folder:"C:\MyWebFiles\Projects" /recurse:off
/type:"*.htm*"
/xwhat:"\<span.#\>(.#)\<\/span\>"
/xwith:"\1"
/case:off
/rx:on
/replace:yes
/run
```

Running Text Workbench with a configuration file

```
hffr.exe /cfg:"c:\MyWeb\Projects\replace_span.hcfg"
```

Chapter 6.
Windows Scripting

OVERVIEW

The Microsoft Windows scripting technology offers a great advantage of creating script files and running them by specifying the script file name as a command line parameter of the **wscript.exe** or **cscript.exe** script host engines. This approach is more powerful and efficient than simply running programmes with parameters on the command line, since it provides greater flexibility. For example, you can process files by creating a Text Workbench object, and then save the results in the log file of your choice.

How to Start

The first thing is to create a Text Workbench instance. A Text Workbench automation object has a name "**TextWorkbench.Application**". In JScript, you can use the following call:

```
var tw = new ActiveXObject("TextWorkbench.Application");
```

Or, if you prefer VBScript:

```
Dim tw
Set tw = CreateObject("TextWorkbench.Application")
```

Now, all you have to do is set properties of the created object, and call Search or Replace to start processing.

Example

The following example in JScript shows how to perform replace operation described in the how-to topic How to Search And Replace Text Using Regular Expressions. The only difference is that the whole operation is automated, and we then create a file containing the detailed report (log file with matches), and open it in Notepad.

```
var tw = new ActiveXObject("TextWorkbench.Application");

// Set the processing folder
tw.sFolders = "C:\\MyWebFiles\\Projects";

// Specify files we want
tw.sFileTypeMask = "*.html";

// We want to process child folders as well
tw.bRecurSubfolders = true;

// We use regular expressions!
tw.bRegExp = true;

// Text to find: "table" tags with the "align" attributes
// whose value is "center"
tw.sFind = "\\<table(.#)align\\=center(.#)\\>";
// Replace "center" with "left" in table tags
tw.sReplace = "\\<table\\1align=left\\2\\>";

// Do not show Text Workbench window
tw.bHidden = true;

// Close Text Workbench when the tw variable
// goes out of scope
// (e.g. when a script or function,
// in which tw is defined, finishes)
tw.bCloseOnDelete = true;

// Other options
tw.sExcludeMask = "";
tw.bCase = false;
tw.bRenamerMode = false;

// Perform replacement
tw.Replace();

// Obtain the detailed report
var s = tw.sFullReport;

// Prepare the file system object
var fso = new ActiveXObject("Scripting.FileSystemObject");
// Create a file
var rfile =
fso.CreateTextFile("c:\\MyWebFiles\\twReport.txt",
true, false);
// Write the report to the file
rfile.Write(s);
rfile.Close();

// Open the log file in Notepad
// (or other application associated with .txt files)
var sh = new ActiveXObject("WScript.Shell");
var sCmd = "cmd /c start c:\\MyWebFiles\\twReport.txt";
var oExec = sh.Run(sCmd, 0);
```

PROPERTIES AND METHODS

The **TextWorkbench.Application** object implements the following methods and properties.

Methods

Method	Description
Search	Performs the search using criteria specified by passing desired values to the **TextWorkbench.Application** object properties.
Replace	Performs the replace operation.
SetBackupMode	Sets how and where the back-up copies are to be created. By default, back-up copies are created in the "**backup**" subfolder of a search folder.
SetTargetMode	Sets how and where the resulting (processed) files are to be created. By default, the processed files replace the existing files in their location.
EnableCollector	Enables the Collector function and specifies its parameters.

Properties

Property	Type	Description
sFolders	string	Specifies one or more semicolon separated folders whose files are to be processed. **This property has no default value.**
sFileTypeMask	string	One or more semicolon-separated masks of files that are to be processed. The default value is *, which matches all files.
sExcludeMask	string	One or more semicolon-separated masks of files that should not be processed. The default value is an empty string, which means that all files that satisfy **sFileTypeMask**, will match.
sScenarioName	string	Specifies the name of a scenario to execute. If the scenario name is correct, it will be activated upon start thus all other text control properties will have no effect. The default value is an empty string.
bRecurSubfolders	boolean	If **true**, specifies that all nested subfolders are to be searched. The default value is **false**.
bCase	boolean	Toggles case sensitivity on (**true**) or off (**false**).

		The default value is **false**.
bRegExp	boolean	Toggles the use of regular expressions. Pass **true** to activate regular expressions, or **false** to deactivate. The default value is **false**.
bRenamerMode	boolean	Passing **true** sets Text Workbench instance in the file renamer mode. The default value is **false**.
sFind	string	Specifies the text to find. The default value is an empty string.
sReplace	string	Specifies the text to replace the found one. The default value is an empty string.
bBackups	boolean	If **true**, back-up copies will be created. The method **SetBackupMode** can be used to control where and how back-ups are created. The default value is **true**.
sBackupExt	string	Specifies the string (extension) that is to be appended to the back-up copies, if they are created in the original directory. The default value is **.bak**.
bCloseOnDelete	boolean	If **true**, Text Workbench window will be closed when a **TextWorkbench.Application** object is deleted (for example, when a script finishes). If **false**, the window will not close. If *m_bHidden* is **true**, an attempt to set *bCloseOnDelete* to **false** will have no effect. The default value is **true**.
sShortReport	string	After the processing, contains the short report. A *short report* consists of the top-level entries of the log view (i.e. messages and files without matches). **This is a read-only property. This property is only valid after the processing.**
sFullReport	string	After the processing, contains the full report. A *full report* contains all entries of the log view (i.e. messages, files and matches). **This is a read-only property. This property is only valid after the processing.**

sHTMLReport	string	After the processing, contains the report in the HTML format with links to files. **This is a read-only property. This property is only valid after the processing.**

METHODS

Search

Starts the search operation on the *twObj*.

```
twObj.Search();
```

Arguments

No arguments.

Example

The following example illustrates the use of the **Search** method.

```
try
{
// Create the object
var tw = new ActiveXObject("TextWorkbench.Application");

// Assign parameters
tw.sFolders = "g:\\web";
tw.sFileTypeMask = "*.html";
tw.sExcludeMask = "";
tw.bRecurSubfolders = true;
tw.bCase = false;
tw.bRegExp = false;
tw.bRenamerMode = false;
tw.sFind = "class=header1";

// Don't close the programme so we can view results
tw.bCloseOnDelete = false;

// Go!
tw.Search();
}
catch(e)
{
// do some ex. handling stuff
}
```

See Also

Replace

Replace

Starts the replace operation on the *twObj*.

```
twObj.Replace();
```

Arguments

No arguments.

Example

The following example illustrates the use of the **Replace** method.

```
try
{
// Create the object
var tw = new ActiveXObject("TextWorkbench.Application");

// Assign parameters
tw.sFolders = "g:\\web";
tw.sFileTypeMask = "*.html";
tw.sExcludeMask = "";
tw.bRecurSubfolders = true;
tw.bCase = false;
tw.bRegExp = false;
tw.bRenamerMode = false;
tw.sFind = "class=header1";
tw.sReplace = "class=header2";

// Close the programme when the script no longer needs it
tw.bCloseOnDelete = true;

// Go!
tw.Replace();
}
catch(e)
{
// do some ex. handling stuff
}
```

See Also

Search

SetBackupMode

Sets the back-up creation mode and path to the back-up folder, on the *twObj*.

```
twObj.SetBackupMode(lMode, sBackupFolder);
```

Arguments

lMode

Specifies the desired back-up mode. The following values are possible.

- **0** - same as a search folder. Back-up files are stored in the same directory as the source files; an extension passed to the **sBackupExt** property is appended to the name of a back-up copy. If the back-up file already exists, it will be overwritten. Files having the extension passed to the **sBackupExt** property are excluded from the search.
- **1** - subfolder of a search folder. Back-up files will be copied to a folder **sBackupFolder** created in the search folder. The back-up subfolder structure mirrors the search folder structure recursively. Back-up files do not change the extension; they are simply copied. Back-up folders are excluded from the search.

- **2** - static folder. Back-up files will be copied to a supplied static folder. The folder path may be a fully qualified path, or may include environment variables, e.g. **%WINDIR%\TEMP**. Back-up files do not change the extension; they are simply copied. Static back-up folders are excluded from the search.

sBackupFolder

For mode **1**, the name of the subfolder in which the back-up copies are created. For mode **2**, full path to a back-up folder.

Remarks

By default, back-up copies are created in the "**backup**" subfolder of a search folder.

Example

The following example illustrates the use of the **SetBackupMode** method.

```
// Create the object
var tw = new ActiveXObject("TextWorkbench.Application");

// Assign parameters
...
// We want to copy the original file versions to a folder "backup"
// which is to be created in the search folder
tw.SetBackupMode(1, "backup");
...

// Go!
tw.Replace();
```

See Also

SetTargetMode

SetTargetMode

Controls how and where the processed files are stored.

```
twObj.SetTargetMode(lMode, sTargetFolder);
```

Arguments

lMode

Specifies the desired mode. The following values are possible.

- **0** - same as a search folder. The existing files will be overwritten.
- **1** - subfolder of a search folder. The processed files will be stored in a folder **sBackupFolder** created in the search folder. The target subfolder is created in the root search folder. The target subfolder structure mirrors the search folder structure recursively. Target folders are excluded from the search.
- **2** - static folder. The processed files will be created in a supplied static folder. The folder path may be a fully qualified path, or may include environment variables, e.g. **%WINDIR%\TEMP**.

sBackupFolder

For mode **1**, the name of the subfolder to which the processed files are copied. For mode **2**, full path to a target folder.

Remarks

By default, the processed files replace the existing files in their location.

The mode **1** cannot be used with FTP folders.

Example

The following example illustrates the use of the **SetTargetMode** method.

```
// Create the object
var tw = new ActiveXObject("TextWorkbench.Application");

// Assign parameters
...
// We want to copy the processed files to a folder "result"
// which is to be created in the search folder
tw.SetTargetMode(1, "result");
...

// Go!
tw.Replace();
```

See Also

SetBackupMode

EnableCollector

Enables the Collector function on the *twObj* and specifies its parameters.

```
twObj.EnableCollector(lMode,
lEncoding,
sCollectorFilePath,
bOverwrite,
bUseReplacementText,
lSeparator,
sSeparator);
```

Arguments

lMode

Specifies how the collector file(s) are to be created. See the Collector topic on full description of these modes. The following values are possible.

- **0** - **Single File**. This is the simplest mode. The single file mode stores all the found data in a single file, path and name provided.
- **1** - **Individual file for each processed file**. This mode creates a new collector file for every new found file.

- **2** - **Individual file for each match**. This is the most powerful though the most complicated mode. This mode creates a new collector file every time a new occurrence of the matching text is found.

lEncoding

Specifies the collector file encoding. The following values are possible.

- **0** - **ANSI**;
- **1** - **Unicode**;
- **2** - **UTF-8**.

sCollectorFilePath

Path and name of the file to collect the text. If the file and/or path do not exist, they will be created. Path and file format depends on the mode specified in *lMode*.

bOverwrite

If **true**, the existing file will be replaced. In the file does not exist, it will be created.
If **false**, appends the collected data to the file contents.

If the file does not exist, it will be created.

bUseReplacementTextq

If **true**, the collector file(s) will contain text as specified in the **sReplace** property.

You can store the replacement text even if simply searching files for text. This option is generally useful for searches with Regular Expressions, as it allows you to alter the found text in any aspect.

lSeparator

Specifies how the collector entries are separated.

- **0** - no additional data is written to the collector file.
- **1** - finalizes each write operation to a collector file with a CR+LF pair.
- **2** - insert text specified in **sSeparator**, after each write.

sSeparator

If **lSeparator** is **2**, specifies the text to be inserted after each write.

Remarks

The **Collector** function is rather complex and intricate concept. You can get more information on different collector modes, as well as other useful information in the Collector Tab topic.

Example

The following example illustrates the use of the **EnableCollector** method.

```
try
{
// Create the object
var tw = new ActiveXObject("TextWorkbench.Application");

// Assign parameters
tw.sFolders = "g:\\web";
tw.sFileTypeMask = "*.html";
tw.sExcludeMask = "";
tw.bRecurSubfolders = true;
tw.bCase = false;
tw.bRegExp = true;
tw.bRenamerMode = false;
tw.sFind = "\\<td.#\\>"; // find TD tags
tw.sReplace = "\L\0"; // make them lowercase

// Set collector modes: store all td tags to a file
tw.EnableCollector(0, // single file
"C:\\coll.txt", // collector file path
1, // overwrite existing coll. file
1, // use replacement text,
1, // add cr+lf to each line
""); // no additional separator

// Go!
tw.Search();
}
catch(e)
{
// do some ex. handling stuff
}
```

See Also

Collector Tab

Methods

Replace, Search