Edific

Edific is an utility that allows you to make diferent kinds of modifications automatically in text files, windows INI format files and registry entries.

It works with an script file having all the operations you want to execute.

See how you can use and make it work for you:

For comments or suggestions please send mail to edific@dlcsistemas.com

Version History of EdiFic

This utility was created as a solution for a specific problem in a project. We wanted to make automatic changes in a group of Windows 3.1 office computers that were connected to a Novell network: to configure expanded memory in DOS, set some paths, add some scripts in the autoexec or config files, set some parameters in ini files, change configuration data in Novell Netware clients, etc.

We developed this solution and then Edific became the most effective utility to do our job in this project, and also for later maintenance of the PC stations.

Later, in a Data Processing Centre of a client we worked, we had also to modify NT logon scripts for a great amount of people: we just cannot do the work file-by-file and a automatic solution to this problem was required. This was to increase and develop more functionalities on our utility to make the job described above: the multiple files.

Think only to change a share directory to another NT computer and to have to modify 300 scripts of people that had this connection: with Edific was easy and automatic !

For many months ago, we thought that was necessary to make a complete version of this utility, that we have found so interesting, so now its the time to release a documented, tested and increased version of this program.

Version 1.0 – March 1997

- Only for DOS.
- Supports basic managemente of text lines
- Supports basic management of Windows INI files

Version 1.2 – June 1997

- Test and fix of minor errors
- Increased functionality on managing text lines
- Adding strings replacing functions

Version 2.0 – July 1998

- Tested and fixed minor errors
- Adding multiple files management

Version 2.1 – December 1998

- Added flags functionality to the functions (Casesen)
- Tested and fixed minor errors

Version 2.2 – January 2000

- New 32 bits version
- Added registry functionality (only 32 bits version)
- Added more flag functionalities
- Added line numbering functions
- Added line text functions
- Added control execution structures

The next releases of this program are depending on you. Please feedback your opinion to us: if this program is helpful to you, what you think is interesting to add, if you have any question, or simply if you have any comment or opinion, please let us know.

This is a freeware program. You can use it freely but you can't change it.
Please let know your colleages the existence of this utility and our web.

http://www.dlcsistemas.com
mailto://edific@dlcsistemas.com

Notes About Edific

Terms and Conditions

By using this software you are consenting to be bound by this agreement.

Edific and Edific32 are a free software.

You may not:

- Modify or translate the executable files or help files provided.
- Sell this software. These utilities are free.

Under no circumstances and under no legal theory, contract or otherwise shall DLC sistemas or the author be liable to you or any other person for any indirect, special, incidental or consequential damages of any character. There is no warranty for damages of any type caused by using this application.

The author will appreciate any kind of support from you if it does not put the author under some obligation.

You can contact the author by email : edific@dlcsistemas.com

You may find the latest version of Edific at DLC Sistemas home page:

http://www.dlcsistemas.com

Requirements

Supported operating systems:

- MS-Windows 95, 98
- MS-Windows NT 3.51 and 4.0
- MS-Windows 3.x
- MS-DOS 6.x

Note: For Windows 3.x and DOS use EDIFIC, othercase use EDIFIC32

Usage in Production Environments

As you can saw at the history of Edific, you know this utility has a main mission that is to help system administrators, operators and technical systems engineers to do controlled automatic changes in any group of workstations.

To do so, just think about the login scripts. Any network technology now has a point where the PC workstation connects to the server and there is a functionality that allows to execute something that the system administrator wants to do on each user workstation: a connection unit to a share, a message, etc… And also now you can use the Edific batches.

Let's see an example in a Windows NT environment. Think also you can make similar things in any IBM OS/2 LAN Server/Warp Server domain or Novell Netware:

In our example guess that there is 300 users in a NT domain group servers. Think there are a Primary domain controller (PDC) and a Backup domain controller (BDC).

The users in our domain have each one a login script defined in the C:\WINNT\System32\Repl\Import\Scripts directory of the PDC and the corresponding configuration in the Users Manager for Domains applet of the system.

Then you want to change all login scripts that have a connection to a server \\SERVER3\SHAREFIN by another that points to a new added server where the shared resource is now \\SERVERNEW\SHARENEW (i.e. because you've migrate it)

Now you can make all changes by hand in all scripts (with the corresponding time spent and mistakes, up to 300 times) or you can use Edific to do the job:

- First, make a backup copy of your scripts on to a secure directory

```
XCOPY C:\WINNT\System32\Repl\Import\Scripts\*.* C:\BACKUPSCRIPTS\*.* /E/V
```

- Create a text file NEWSRV.SC with this content:

```
LoadAllFile "C:\WINNT\System32\Repl\Import\Scripts\*.Bat"
ReplaceString "\\SERVER3\SHAREFIN" by "\\SERVERNEW\SHARENEW"
SaveAllFile
End
```

- Execute the script as follows:

```
EDIFIC NEWSRV.SC
```

- Be sure all Backup Domain Controllers have the same version of the scripts directory (see replication directories in your NT documentation)

Then you've changed all scripts that had the old resource share by the new. Note that with this, all files were loaded an saved, even the files that had no changes of resource names.

But you can also execute scripts, not only as an "automatic edition file" but as an powerful setting of workstation maintenance.

Let's see a new little sample. Think all your 95 workstations have a program or application you want to modify its configuration: the INI files, that points now to another sharename; the Registry, that has an performance-down configuration parameter, a default configuration that you wants to erase, etc.

You've now two possibilities: or to give the problem to the PC-Installations & Maintenance department (if you have it in your entreprise) or to walk arround all the floors of the building(s) and do the changes yourself PC-by-PC (not a good idea, isn't-it ? :-{)

You have another way also. You can use edific if you know specifically what changes you want to do (i.e. change the param x of the file or registry to the value y)

Guess you want to change the parameter *DefaultDir* of the corporate application software *DataQueryCorp* that is in the key HKEY_LOCAL_MACHINE\SOFTWARE\DQ\DataQueryCorp\2.0 of the all W95 users workstations registry.

Then you can follow the steps below:

- Put the EDIFIC32.EXE program in the scripts repository C:\WINNT\System32\Repl\Import\scripts. (Remember: **never** allow users to write on to the scripts repository on any domain controller – the NETLOGON share)
- Create the following file with NOTEPAD, save it as NEWDIR.SC and put it in the scripts repository too:

```
InsertRegistryValue "HKLM\Software\DQ\DataQueryCorp\2.0\DefaultDir" String "C:\NEWDIR"
End
```

- Modify all logon scripts to add at first or last the line below (note you can do this modification like the example shown above with the command BeforeBeginInsert or AfterEndInsert)

```
@EDIFIC32 NEWDIR.SC /o
```

- Be sure all Backup Domain Controllers have the same version of the scripts directory

Now it's all. When people get connected, the script will execute the Edific and will change the registry. Note that when the logon script is executing no credits are shown because the /o flag.

There are many examples of using Edific. You can combine its functionality in logon scripts with these on Kix32 provided by Microsoft in its Resource Kit for Windows NT.

Introduction

Versions and Call Format

There are two versions of Edific utility. First, there is the 16 bits version, corresponding to DOS systems and Windows 3.x (Edific). The second is the 32 bits version for Windows 95,98 and NT (Edific32).

Both versions have the same behavior and keywords, but the 16 bits version has no support for registry edition commands.

To execute EdiFic, follow this format:

`EDIFIC` *`ScriptFile`* `[/d] [/h] [/l`*`LogFile`*`] [/a`*`LogFile`*`] [/o]`

Where

ScriptFile	Mandatory, is a file with scripts command for Edific to execute
/d	Optional, tells Edific to show the executed commands and its results
/h	Optional, shows the little help on call format
/l*Logfile*	Optional, tells Edific to put all results in this *Logfile*, creating or overwriting it
/a*Logfile*	Optional, tells Edific to put all results int this *Logfile*, appending the information if the file already exists
/o	Optional, tells Edific to not to display the credits

Go next >>

How Works Edific

The script file contains all the commands you want to execute. You can open or save any text file, make substitutions, replacements, line deletions and insertions, etc.

Edific doesn't need any temporary disk repository because it gets all the file in memory to do all the operations the script want.

For this reason, text data files to work with can't be very big, but it depends of the power of the system on Edific will be executed.

To make scripts use your preferred text editor.

See the information on...

Loading and saving files

Modifying a Windows INI file

Working with strings

Editing the registry (!)

Repeating Structures

Working with case letters

Working with delimiters and comments

Working with debug information

Loading and saving files

When you load files with the corresponding command scripts in Edific, the original file is read and closed and the data resides in memory to make changes.

At any time only one file data is on memory. You can create a script with multiple loads and saves and all them will be executed sequentially, this become to work with multiple files, but any command execution affects only the data in memory the program has at a specific time.

When you execute a LoadFile or LoadAllFile command, all data in memory will be discarded. You have to save the data previously to this commands.

Then you see that load and save commands are the only Edific uses to go to disk. By this, all functions between a load and a save are treated in memory. The changes are in memory, so you have to save to get the new data file.

Let's see an example of loading and saving files:

Make a file with any editor (i.e. NOTEPAD) and create a file named SAMPLE.SC at C:\TEMP directory:

```
LoadFile "C:\CONFIG.SYS"
BeforeBeginInsert "@ECHO OFF"
SaveFile "C:\CONFIG2.SYS"
End
```

Go now to a command prompt and execute the script from the directory you have Edific.exe:

EDIFIC C:\TEMP\SAMPLE.SC

You have now the prompt. You can see that you have another file in C:\ called CONFIG2.SYS that has the same content as CONFIG.SYS but the first line added.

Guess now you want to work with multiple files that have similar names except for a number or any else. Think you have forty files in the C:\DATA directory called DATA001.TXT to DATA040.TXT and you would to substituye any string, or delete any line in all of these files. For that you can use the LoadAllFile or SaveAllFile commands. Let's see this example:

```
LoadAllFile "C:\DATA\DATA*.TXT"
ReplaceString "Tihs" by "This"
SaveAllFile
End
```

Now you will have forty files corrected automatically :-)
Or best, you can have a copy of all these files by setting the SaveAllFile of the script above like this:

```
SaveAllFile "C:\NEWDATA\NWDAT*.TXT"
```

Then you will get forty files in the C:\NEWDATA called from NWDAT001.TXT to NWDAT040.TXT that are copy of the C:\DATA\DATA*.TXT but corrected.

You can see more information on loading and saving files by viewing the following samples:

Modifying a file,
Saving a new file,
Modifying multiple files,
Creating multiple files,
Changing the default directory

Modifying Windows INI files

You can use Edific to modify any Windows INI file.
There is a set of command specially oriented to modify INI files. These functions are here.

To work with this, simply get the file by loading and apply the changes you want with the specified functions: you can add data keys, delete keys, delete sections, add sections. You can combine also the strings and line functions to add complexity.

In example, you can see now how to add a driver in a W3.x system or a W95 for compatibility:

```
LoadFile "C:\WINDOWS\SYSTEM.INI"
InsertValue "386Enh" "device" "C:\WINDOWS\SYSTEM\DRVTEST.386"
SaveFile
End
```

Then you get a SYSTEM.INI file with the line *device = C:\WINDOWS\SYSTEM\DRVTEST.386* added to its [386Enh] section.

You can see more information on working with Windows INI files by viewing the following samples:

Replacing a key value,
Setting a new section and key value,
Adding a section,
Deleting a section,
Deleting keys

Working with Strings

A set of powerful string functions is needed when working automatically with text files.

Edific has the support of two groups of functions oriented to line and string treatement. They are line and substring content oriented group and line number oriented group.

The first group of functions looks for matches with a specific string pattern and makes changes in many ways, like changing lines, replacing found text to a new text, deleting lines, adding lines before or after any line that matches the conditions, etc.

The second refers to the position a specific line has inside the file. Then there is functions to add lines before or after an specific linenumber position, adding at top or bottom, deleting the line number-th, etc.

You can see more information on working with lines and strings by viewing the following samples:

Inserting a line after a line,
Inserting a line after a line that contains data,
Inserting a line before a line,
Inserting a line before a line that contains data,
Deleting lines,
Deleting lines that contains data substrings,
Replacing lines,
Replacing lines containing substrings,
Replacing Substrings,
Adding lines with linenumbers,
Adding lines at top or bottom,
Deleting line numbers

Editing the registry

Probably in NT,95 or 98 you should need to make changes to the registry. Here is a set of commands you can use to do these jobs automatically, specially if you want to make changes to multiple computers.

The structure of the Registry is like a directory hierarchy, where the files are data values and the directories are the keys. Sure you have worked anytime with REGEDIT or REGEDT32.

The data values have a specific data type and also any data value or key has their own access permissions.

In NT, to work with Edific inside the Registry, the execution userid of the process must have write permissions to the keys and data values the edific script wants to modify. Generally the user must have Admin permissions.

You can use only registry commands with the 32bits version of Edific (EDIFIC32). These commands on 16bits version are ignored.

WARNING: It's recommended to make tests when you work with these functions because any mistake could break your system configuration.

You can see more information on working with Registry by viewing the following samples:

Creating keys and inserting values in the registry,
Making data changes to the registry,
Deleting keys and values in the registry,

Repeating Structures

If you want to apply any command to a file a specific number of times, you have two options: to repeat the command in the script n-times, or to do a loop.

You can use the Repeat…EndRepeat to construct loops.

The Repeat sentence is followed by the number of times you want to loop the entire structure till EndRepeat.

Also you can construct Repeat…EndRepeat structures inside another:

```
Repeat 10
  . . .
  Repeat 20

    Command1

  EndRepeat
  . . .
  Repeat 5

    Command2

  EndRepeat
  . . .
EndRepeat
```

You can see more information on working with iterations by viewing the following samples:

Doing Iterations

Working with case letters

On every command of string data treatement you can indicate care with case letters. There is a keyword flag called **CaseSen** you can use at the end of the command to set up this functionality.

By default all comparations ignore case sensitive. Only if you put this flag, matches on “P” to “p” will not arise.

In all cases, the use of the flag **CaseSen** is optional.

You can see more information on working with case by viewing the following samples:

Inserting a line after a line,
Inserting a line after a line that contains data,
Inserting a line before a line,
Inserting a line before a line that contains data,
Deleting lines,
Deleting lines that contains data substrings,
Replacing lines,
Replacing lines containing substrings,
Replacing Substrings,
Making data changes to the registry,
Deleting keys and values in the registry

Working with delimiters and comments

You can specify the parameters data to the commands in the Edific script file by delimiting each one between (“) characters. This is to avoid wrong interpretation of parameters by Edific when spaces inside the parameters are contained.

But, what when you want to put (“) characters in your parameters ?. Well, there are some options to delimite data, not only the (“). See the table below:

Options to define data in script files	***Examples***
No delimiters. You can use that when a number or data without spaces are specified.	1983 letmenow
Delimiting with (“). You should use this with lines with spaces.	“let me now” “It’s a sample”
Delimiting with (‘). You can use this when your data contains (“) or spaces.	‘This is a “sample” of a line’
Delimiting with (‘). You can use this when your data contains (“) or spaces.	‘This is a “sample” of a line’ ‘This is a sample of a line’
Delimiting with (@). You can use this when your data contains (“), (‘) or simply spaces.	@It’s a “sample” of a line@ @Also this@
Delimiting with (#). You can use this when your data contains (“),(‘), (@) or spaces.	#This is a “sample” of a line# #Note this line has only spaces# #H@ve u’ thought about this “case”?#

See the following samples of commands. All they are valid:

```
InsertValue 386Enh ’device’ FILE.435
InsertValue #386Enh# device "FILE.435"
InsertValue 386Enh device FILE.435

BeforeLine @Searching for "this" line@ Insert ’this line #first# :-) ’
BeforeLine #Searching for "this" line# Insert @this line #first# :-) @

Display 1293981
Display "1293981"
```

In the scripts also you can include comments. The comments begins with a (‘) character, and then Edific ignores the line.

```
’ This is a comment line
```

Working with debug information

The debug information the edific can show is a little trace of the operations executed.

Each executed line is shown to the screen of log file in the following format:

DD/MM/YYYY HH:MM:SS Command [Parameter1] [Parameter2] … [ParameterN] Result

Let's see a sample of this. Guess that you have a script file and you run this script by

EDIFIC32 SAMPLE.SC /D /LSAMPLE.LOG

Then you will get a SAMPLE.LOG file like this:

```
09/01/2000 20:28:52 InsertRegistryValue [HKCU\Environment\TestKey\Param Number] [Dword]
[1003644].... Ok
09/01/2000 20:28:52 InsertRegistryValue [HKCU\Environment\TestKey\Param NumberBigEndian]
[NumeroBE] [1003644].... Ok
09/01/2000 20:28:52 InsertRegistryKey [HKEY_CURRENT_USER\Environment\TestKey2].... Ok
09/01/2000 20:28:52 InsertRegistryValue [HKCU\Environment\TestKey\Param Text] [String] [A
string message].... Ok
09/01/2000 20:28:52 InsertRegistryValue [HKCU\Environment\TestKey\Subkey2\Subkey21\
AnotherText] [String] [It's a message line].... Ok
09/01/2000 20:28:52 InsertRegistryValue [HKCU\Environment\TestKey\Param ExpText]
[ExpString] [An expanded string message].... Ok
09/01/2000 20:28:52 InsertRegistryValue [HKCU\Environment\TestKey\Param MultiText]
[MultiString] [The first line.The second line..Still the second line.The third line ]....
Ok
09/01/2000 20:28:52 InsertRegistryValue [HKCU\Environment\TestKey\Param Data] [Binary]
[01F0E578 9FA03E80 910087E9].... Ok
09/01/2000 20:28:52 InsertRegistryValue [HKCU\Environment\TestKey\Subkey1\ParamData1]
[Binary] [9FA03E80 910087E9].... Ok
09/01/2000 20:28:52 DeleteRegistryKey [HKCU\Environment\TestKey\Subkey2].... Ok
09/01/2000 20:28:52 DeleteRegistryValue [HKCU\Environment\TestKey\Param Number].... Ok
```

The result of the execution is Ok when the command can complete its steps. This doesn't mean that if you execute i.e. *ReplaceString "this" by "these"* the command will do any change, but only that the command can complete its execution.

The Error conditions can be shown when, in example, you would access to a key in the registry that you have no right access, or a file you want to overwrite and it's read only, or the command cannot got more memory to work, etc.

Commands by Name

Command	*Abreviation*	*Description*
AfterEndInsert	AFTENDINS	Inserts a line after the end of the file
AfterLine	AFTLIN	Inserts a line after the designed line
AfterLineContaining	AFTLINCON	Inserts a line after another that contains a specific substring
AfterLineNumber	AFTLINNUM	Inserts a line after the number-th line
BeforeBeginInsert	BEFBEGINS	Inserts a line at top of the file
BeforeLine	BEFLIN	Inserts a line before the designed line
BeforeLineContaining	BEFLINCON	Inserts a line before another that contains a specific substring
BeforeLineNumber	BEFLINNUM	Inserts a line before the number-th line
ChangeRegistryValue	CHAREGVAL	Replaces a data value for another in all the registry
DeleteKey	DELKEY	Deletes a key in a Windows INI file
DeleteLine	DELLIN	Deletes a line
DeleteLineContaining	DELLINCON	Deletes a line that contains a specific substring
DeleteLineNumber	DELLINNUM	Deletes the number-th line
DeleteRegistryKey	DELREGKEY	Deletes a key in the registry and all its subkeys
DeleteRegistryValue	DELREGVAL	Deletes a value in the registry
DeleteSection	DELSEC	Deletes a section [] in a Windows INI file
Directory	DIR	Changes the current working directory
Disc	DSK	Changes the current working disk unit
Display	DIS	Displays a text
End	END	End of the script
EndRepeat	ENDREP	End of the repeating structure
InsertRegistryKey	INSREGKEY	Inserts a key in the registry
InsertRegistryValue	INSREGVAL	Inserts/Puts a value in the registry
InsertSection	INSSEC	Inserts a section [] in a Windows INI file
InsertValue	INSVAL	Inserts a value in a Windows INI file
LoadAllFile	LOAALLFIL	Loads all files to operate with
LoadFile	LOAFIL	Loads a file to operate with
Repeat	REP	Begins a repeat structure
ReplaceLine	REPLIN	Replaces a line with another
ReplaceLineContaining	REPLINCON	Replaces a line that contains a specific substring with another
ReplaceString	REPSTR	Replaces a substring with another
SaveAllFile	SAVALLFIL	Saves all files loaded with LoadAllFile
SaveFile	SAVFIL	Saves a file

All Commands by Group

Working with Files and Directories

With these functions you can get files to edit and save they as modified. You can also get multiple files and make changes to all.

Command	*Abreviation*	*Description*
LoadAllFile	LOAALLFIL	Loads all files to operate with
LoadFile	LOAFIL	Loads a file to operate with
SaveAllFile	SAVALLFIL	Saves all files loaded with LoadAllFile
SaveFile	SAVFIL	Saves a file
Directory	DIR	Changes the current working directory
Disc	DSK	Changes the current working disk unit

Working with Windows INI files

Easily, you can change any windows INI file with this functions: add, delete or replace keys, sections and values. These functions in combination with strings ones, and you will make all you want.

Command	*Abreviation*	*Description*
InsertSection	INSSEC	Inserts a section [] in a Windows INI file
InsertValue	INSVAL	Inserts a value in a Windows INI file
DeleteKey	DELKEY	Deletes a key in a Windows INI file
DeleteSection	DELSEC	Deletes a section [] in a Windows INI file

Working with Lines and Substrings

There are a powerful set of commands you will use to edit your string data. You can add lines before or after a text line, that could begin or contain a substring, replace lines, etc.

Command	*Abreviation*	*Description*
AfterLine	AFTLIN	Inserts a line after the designed line
AfterLineContaining	AFTLINCON	Inserts a line after another that contains a specific substring
BeforeLine	BEFLIN	Inserts a line before the designed line
DeleteLine	DELLIN	Deletes a line
DeleteLineContaining	DELLINCON	Deletes a line that contains a specific substring
BeforeLineContaining	BEFLINCON	Inserts a line before another that contains a specific substring
ReplaceLine	REPLIN	Replaces a line with another
ReplaceLineContaining	REPLINCON	Replaces a line that contains a specific substring with another
ReplaceString	REPSTR	Replaces a substring with another

Working with Lines and Linenumbers

And with these commands you can do all the changes like strings funcions (up) but with the reference of the line number within the file.

Command	*Abreviation*	*Description*
AfterEndInsert	AFTENDINS	Inserts a line after the end of the file

AfterLineNumber	AFTLINNUM	Inserts a line after the number-th line
BeforeBeginInsert	BEFBEGINS	Inserts a line at top of the file
BeforeLineNumber	BEFLINNUM	Inserts a line before the number-th line
DeleteLineNumber	DELLINNUM	Deletes the number-th line

Working with Execution Steps

Also you can do simple sequences to repeat operations.

Command	***Abreviation***	***Description***
Repeat	REP	Begins a repeat structure
EndRepeat	ENDREP	End of the repeating structure
End	END	End of the script

Working with Display

If you want, you can display messages to the screen or the log file

Command	***Abreviation***	***Description***
Display	DIS	Displays a text

Working with Registry

And there are a set of commands you can use to make changes to registry entries. Be careful when you touch this component !.

Command	***Abreviation***	***Description***
ChangeRegistryValue	CHAREGVAL	Replaces a data value for another in all the registry
DeleteRegistryKey	DELREGKEY	Deletes a key in the registry and all its subkeys
DeleteRegistryValue	DELREGVAL	Deletes a value in the registry
InsertRegistryKey	INSREGKEY	Inserts a key in the registry
InsertRegistryValue	INSREGVAL	Inserts/Puts a value in the registry

Commands by Function

Working with Files and Directories

With these functions you can get files to edit and save they as modified. You can also get multiple files and make changes to all.

Working with Windows INI files

Easily, you can change any windows INI file with this functions: add, delete or replace keys, sections and values.

Working with Lines and Substrings

There are a powerful set of commands you will use to edit your string data. You can add lines before or after a text line, that could begin or contain a substring, replace lines, etc.

Working with Lines and Linenumbers

And with these commands you can do all the changes like strings funcions (up) but with the reference of the line number within the file.

Working with Execution Steps

Also you can do simple sequences to repeat operations.

Working with Display

If you want, you can display messages to the screen or the log file

Working with Registry

And there are a set of commands you can use to make changes to registry entries. Be careful when you touch this component !.

You can see also the Complete Function List by name or by group

Working with files and Directories

Command	*Abreviation*	*Description*
LoadAllFile	LOAALLFIL	Loads all files to operate with
LoadFile	LOAFIL	Loads a file to operate with
SaveAllFile	SAVALLFIL	Saves all files loaded with LoadAllFile
SaveFile	SAVFIL	Saves a file
Directory	DIR	Changes the current working directory
Disc	DSK	Changes the current working disk unit

Working with Windows INI files

Command	*Abreviation*	*Description*
InsertSection	INSSEC	Inserts a section [] in a Windows INI file
InsertValue	INSVAL	Inserts a value in a Windows INI file
DeleteKey	DELKEY	Deletes a key in a Windows INI file
DeleteSection	DELSEC	Deletes a section [] in a Windows INI file

Working with Lines and Substrings

Command	*Abreviation*	*Description*
AfterLine	AFTLIN	Inserts a line after the designed line
AfterLineContaining	AFTLINCON	Inserts a line after another that contains a specific substring
BeforeLine	BEFLIN	Inserts a line before the designed line
DeleteLine	DELLIN	Deletes a line
DeleteLineContaining	DELLINCON	Deletes a line that contains a specific substring
BeforeLineContaining	BEFLINCON	Inserts a line before another that contains a specific substring
ReplaceLine	REPLIN	Replaces a line with another
ReplaceLineContaining	REPLINCON	Replaces a line that contains a specific substring with another
ReplaceString	REPSTR	Replaces a substring with another

Working with Lines and Linenumbers

Command	*Abreviation*	*Description*
AfterEndInsert	AFTENDINS	Inserts a line after the end of the file
AfterLineNumber	AFTLINNUM	Inserts a line after the number-th line
BeforeBeginInsert	BEFBEGINS	Inserts a line at top of the file
BeforeLineNumber	BEFLINNUM	Inserts a line before the number-th line
DeleteLineNumber	DELLINNUM	Deletes the number-th line

Working with Execution Steps

Command	*Abreviation*	*Description*
Repeat	REP	Begins a repeat structure
EndRepeat	ENDREP	End of the repeating structure
End	END	End of the script

Working with Display

Command	*Abreviation*	*Description*
Display	DIS	Displays a text

Working with Registry

Command	*Abreviation*	*Description*
ChangeRegistryValue	CHAREGVAL	Replaces a data value for another in all the registry
DeleteRegistryKey	DELREGKEY	Deletes a key in the registry and all its subkeys
DeleteRegistryValue	DELREGVAL	Deletes a value in the registry
InsertRegistryKey	INSREGKEY	Inserts a key in the registry
InsertRegistryValue	INSREGVAL	Inserts/Puts a value in the registry

List of Samples

SaveFile

Syntax

SaveFile ["*filename*"]

Also

SAVFIL ["*filename*"]
GuardaFichero ["*filename*"]
GrabaFichero ["*filename*"]

Parameters

Parameter	***Description***
Filename	Optional. Saves an edited file with this name of file. If not specified, SaveFile uses the one when the file was loaded with LoadFile or LoadAllFile. Wildcards are not allowed.

Operation

Saves the current edited and modified file with the file name specified. If no file was loaded before, this command does nothing.

Related commands

SaveAllFile, LoadFile, LoadAllFile
Working with Files and Directories

Samples

Sample 1: Modifying a file

Sample 2: Saving a new file

Modifying a file

Sample of

SaveFile, LoadFile

Script
File SAMPLE.SC:

```
' Script to modify a file
LoadFile "C:\TEST\SPACE.TXT"
. . .
SaveFile
End
```

Execution

EDIFIC SAMPLE.SC

Results

File SPACE.TXT saved with changes

Comments

The script SAMPLE.SC has loaded the file SPACE.TXT, has made changes and has saved it with its name.

Saving a new file

Sample of

SaveFile, LoadFile

Script

File SAMPLE.SC:

```
' Script to create a file
LoadFile "C:\TEST\SPACE.TXT"
. . .
SaveFile "C:\TEST\SKY.TXT"
. . .
SaveFile "C:\TEST\STAR.TXT"
End
```

Execution

EDIFIC SAMPLE.SC

Results

There is a new file SKY.TXT that is file SPACE.TXT with changes. Also there is a file STAR.TXT that is like SKY.TXT with changes.

Comments

The script SAMPLE.SC has loaded the file SPACE.TXT, has made changes and has saved to the new file SKY.TXT. Now there are two files, SPACE.TXT and SKY.TXT. SPACE.TXT has not modified at all.

After the first SaveFile command, you still can do changes to the edited file, because SaveFile doesn't empty the memory of the edited file. Then the script changed the data and the second SaveFile saved it in the STAR.TXT file.

LoadFile

Syntax

LoadFile "*filename*"

Also

LOAFIL "*filename*"
CargaFichero "*filename*"

Parameters

Parameter	***Description***
Filename	Is the file to work with. Wildcards are not allowed.

Operation

Loads the desired text file. After load, all operations will work with its contents. Any work not saved before this command will be discarded.

Related commands

LoadAllFile, SaveFile, SaveAllFile
Working with Files and Directories

Samples

Sample 1: Modifying a file

Sample 2: Saving a new file

LoadAllFile

Syntax

LoadAllFile “*filename*”

Also

LOAALLFIL “*filename*”
CargaTodoFichero “*filename*”

Parameters

Parameter	***Description***
Filename	Specifies all the files this command will load and operate with. Wilcards are allowed.

Operation

Loads sequencially all the desired text files indicated by the wilcards. You can use *,? to describe the files as you do when putting wilcards arguments to OS commands. After load, all operations will work with the contents of the current file. Any work not saved before this command will be discarded.
This command works with SaveAllFile. When a SaveAllFile command is encountered within the execution of the script, control jumps to the next command of the first previous LoadAllFile, getting the next file to load.

Related commands

LoadFile, SaveFile, SaveAllFile
Working with Files and Directories

Samples

Sample 1: Modifying multiple files

Sample 2: Creating multiple new files

Creating multiple files

Sample of

SaveAllFile, LoadAllFile

Script

File SAMPLE.SC:

```
' Script to create multiple files
LoadAllFile "C:\TEST\S*.TXT"
. . .
SaveAllFile "C:\TEST\F*.??M"
End
```

Execution

EDIFIC SAMPLE.SC

Results

Let's guess there are four files in the C:\TEST directory: SPACE.TXT, SKY.TXT, STAR.TXT, BLUE.TXT. The files that begins by S will be loaded by the execution of the script and files FPACE.TXM, FKY.TXM and FTAR.TXM will be created as result of SaveAllFile. Note that you have the same names of the files but replacing S to F and last T to M.

Comments

The script SAMPLE.SC has loaded first the file SKY.TXT, next SPACE.TXT and so on. All the commands between LoadAllFile and SaveAllFile have been executed for each of these files.
SaveAllFile gets the name corresponding to the current working file and applies the wildcards to save a new file. This new file is like the original but with the changes.

You can try all combinations of wilcards to get any set of filename results.

Modifying multiple files

Sample of

SaveAllFile, LoadAllFile

Script

File SAMPLE.SC:

```
' Script to modify multiple files
LoadAllFile "C:\TEST\S*.TXT"
. . .
SaveAllFile
End
```

Execution

EDIFIC SAMPLE.SC

Results

Let's guess there are four files in the C:\TEST directory: SPACE.TXT, SKY.TXT, STAR.TXT, BLUE.TXT. The files that begins by S will be modified by the execution of the script.

Comments

The script SAMPLE.SC has loaded first the file SKY.TXT, next SPACE.TXT and so on. All the commands between LoadAllFile and SaveAllFile have been executed for each of these files.
Also the files have been saved with changes.

SaveAllFile

Syntax

SaveAllFile ["*filename*"]

Also

SAVALLFIL ["*filename*"]
GrabaTodoFichero ["*filename*"]
GuardaTodoFichero ["*filename*"]

Parameters

Parameter	***Description***
Filename	Optional. Specifies the filename pattern the files will be saved as. If this name are not specified, the original name at LoadAllFile command will be used. Wildcards are allowed here.

Operation

SaveAllFile works in conjunction with LoadAllFile. This pair of commands work to make changes to a set of files. LoadAllFile loads sequencially all the desired text files indicated by the wilcards. When SaveAllFile is encountered, the current working text is saved aplying the wildcards that *filename* contains to the original name of the file loaded previously. Then you can create a new set of files.

If you have not specified any *filename* parameter, files will be saved with the original name.

When SaveAllFile detects that there are still files to get, this command jumps control to the corresponding LoadAllFile command, and next file will be loaded for work with.

Related commands

LoadAllFile, SaveFile, LoadFile
Working with Files and Directories

Samples

Sample 1: Modifying multiple files

Sample 2: Creating multiple new files

Disc

Syntax

Disc “*unit*”

Also

DSK “*unit*”
Disco “*unit*”

Parameters

Parameter	***Description***
Unit	Tells what disk unit to switch to

Operation

Disc switches to the *unit* hard disk as a default to get or put files.

Related commands

Directory
Working with Files and Directories

Samples

Sample : Changing default dir

Changing the default directory

Sample of

Disc, Directory

Script
File SAMPLE.SC:

```
' Script to change default directory
Disc "C:"
Directory "\TEMP"
LoadFile "STAR.TXT"
. . .
SaveFile
End
```

Execution

EDIFIC SAMPLE.SC

Results

Let's guess there is a file named STAR.TXT in the C:\TEST directory. With Disc and Directory commands, LoadFile could get the file.

Comments

At the exit of Edific, the default directory is the specified by Disc and Directory.

Directory

Syntax

Directory “*path*”

Also

DIR “*path*”
Directorio “*path*”

Parameters

Parameter	***Description***
Path	Tells what directory to switch to

Operation

Directory switches to the *path* directory as default to get or put files.

Related commands

Disc
Working with Files and Directories

Samples

Sample : Changing default dir

InsertValue

Syntax

InsertValue “*section*” “*key*” “*value*”

Also

INSVAL “*section*” “*key*” “*value*”
InsertaValor “*section*” “*key*” “*value*”

Parameters

Parameter	***Description***
Section	Is the section attribute of the INI file
Key	Is the key to add or change value to
Value	Is the data assigned to the key

Operation

InsertValue works with Windows INI text files to insert a key value. If the key exists, is replaced with the specified value. If the key doesn't exists, the key will be created with the value. If the section doesn't exists, the specified section is created at the end of file to allow put the corresponding key.

Related commands

InsertSection, DeleteKey, DeleteSection
Working with Windows INI Files

Samples

Sample 1: Replacing a key value

Sample 2: Setting a new section and key value

Replacing a key value

Sample of

InsertValue

Script and Files
File SAMPLE.SC:

```
' Script to change a windows INI value
LoadFile "C:\TEST\APPLIC.INI"
InsertValue "Begin" "AutoRun" "1"
SaveFile
End
```

File APPLIC.INI:

```
[Begin]
WinPos = 32,200,500,639
AutoRun = 0
[Directory]
TempDir=C:\TEMP
WorkDir=C:\TEST
SwapDir=C:\TEMP2
```

Execution

EDIFIC SAMPLE.SC

Results

The file APPLIC.INI will be modified by setting AutoRun to 1.

File APPLIC.INI (result):

```
[Begin]
WinPos = 32,200,500,639
AutoRun = 1
[Directory]
TempDir=C:\TEMP
WorkDir=C:\TEST
SwapDir=C:\TEMP2
```

Comments

You can also use parameters in InsertValue without quotations if data doesn't contains spaces
It doesn't matter about case attribute on matching key or section text, but if you specify any of this data, the resulting file will get these properties (i.e. if you put "AUTOupdate" in the command of the script file, at the final file you will get "AUTOupdate" in place of "AutoUpdate").

Setting up a new section and key value

Sample of

InsertValue

Script and Files
File SAMPLE.SC:

```
' Script to change a windows INI file
LoadFile "C:\TEST\APPLIC.INI"
InsertValue "Begin" "AutoUpdate" "0"
InsertValue "Error" "Message" "This is an error message"
SaveFile
End
```

File APPLIC.INI:

```
[Begin]
WinPos = 32,200,500,639
AutoRun = 0
[Directory]
TempDir=C:\TEMP
WorkDir=C:\TEST
SwapDir=C:\TEMP2
```

Execution

EDIFIC SAMPLE.SC

Results

The file APPLIC.INI will be modified by adding the key value AutoUpdate and the section [Error]

File APPLIC.INI (result):

```
[Begin]
WinPos = 32,200,500,639
AutoRun = 0
AutoUpdate = 0
[Directory]
TempDir=C:\TEMP
WorkDir=C:\TEST
SwapDir=C:\TEMP2
[Error]
Message = This is an error message
```

Comments

You can also use parameters in InsertValue without quotations if data doesn't contains spaces.
It doesn't matter about case attribute on matching key or section text, but if you specify any of this data, the resulting file will get these properties (i.e. if you put "AUTOupdate" in the command of the script file, at the final file you will get "AUTOupdate" in place of "AutoUpdate").

InsertSection

Syntax

InsertSection “*section*”

Also

INSSEC “*section*”
InsertaSeccion “*section*”

Parameters

Parameter	***Description***
Section	Is the section attribute of the INI file

Operation

InsertSection works with Windows INI text files to insert a section block. If the section exists the command does nothing. If the section doesn’t exists, an empty block section will be created at end of file.

Related commands

InsertValue, DeleteKey, DeleteSection
Working with Windows INI Files

Samples

Sample: Adding a Section to a Windows INI file

Adding a section to a Windows INI file

Sample of

InsertSection

Script and Files
File SAMPLE.SC:

```
' Script to change a windows INI file
LoadFile "C:\TEST\APPLIC.INI"
InsertSection "Directory"
InsertSection "Work"
SaveFile
End
```

File APPLIC.INI:

```
[Begin]
WinPos = 32,200,500,639
AutoRun = 0
[Directory]
TempDir=C:\TEMP
WorkDir=C:\TEST
SwapDir=C:\TEMP2
```

Execution

EDIFIC SAMPLE.SC

Results

The file APPLIC.INI will be modified by adding the empty section [Work]

File APPLIC.INI (result):

```
[Begin]
WinPos = 32,200,500,639
AutoRun = 0
[Directory]
TempDir=C:\TEMP
WorkDir=C:\TEST
SwapDir=C:\TEMP2
[Work]
```

Comments

The [Begin] section alredy was in the file.
You can also use parameters in InsertSection without quotations if data doesn't contains spaces. It doesn't matter about case attribute on matching key or section text (i.e. is the same *InsertSection "Directory"* and *InsertSection "direcTORY"*)

DeleteSection

Syntax

DeleteSection “*section*”

Also

DELSEC “*section*”
EliminaSeccion “*section*”

Parameters

Parameter	***Description***
Section	Is the section attribute of the INI file

Operation

DeleteSection works with Windows INI text files to delete a section block. If the section exists, all its keys are deleted with the section.

Related commands

InsertValue, InsertSection, DeleteKey
Working with Windows INI Files

Samples

Sample: Deleting a Section in a Windows INI file

Deleting a section in a Windows INI file

Sample of

DeleteSection

Script and Files
File SAMPLE.SC:

```
' Script to change a windows INI file
LoadFile "C:\TEST\APPLIC.INI"
DeleteSection "directory"
DeleteSection "Test"
SaveFile
End
```

File APPLIC.INI:

```
[Begin]
WinPos = 32,200,500,639
AutoRun = 0
[Directory]
TempDir=C:\TEMP
WorkDir=C:\TEST
SwapDir=C:\TEMP2
```

Execution

EDIFIC SAMPLE.SC

Results

The file APPLIC.INI will be modified by erasing the section [Directory]

File APPLIC.INI (result):

```
[Begin]
WinPos = 32,200,500,639
AutoRun = 0
```

Comments

All the keys of the corresponding [Directory] section block have been deleted.
The [Test] section didn't exists.
You can also use parameters in DeleteSection without quotations if data doesn't contains spaces. It doesn't matter about case attribute on matching key or section text (i.e. is the same *DeleteSection "Directory"* and *DeleteSection "direcTORY"*)

DeleteKey

Syntax

DeleteKey “*section*” “*key*”

Also

DELKEY “*section*” “*key*”
EliminaClave “*section*” “*key*”

Parameters

Parameter	***Description***
Section	Is the section attribute of the INI file
Key	Is the key attribute of the INI file

Operation

DeleteKey works with Windows INI text files to delete a key value. If the key exists, the key and its value is deleted.

Related commands

InsertValue, InsertSection, DeleteSection
Working with Windows INI Files

Samples

Sample: Deleting a Key in a Windows INI file

Deleting a key in a Windows INI file

Sample of

DeleteKey

Script and Files
File SAMPLE.SC:

```
' Script to change a windows INI file
LoadFile "C:\TEST\APPLIC.INI"
DeleteKey "directory" "Workdir"
DeleteKey "Test" "message"
SaveFile
End
```

File APPLIC.INI:

```
[Begin]
WinPos = 32,200,500,639
AutoRun = 0
[Directory]
TempDir=C:\TEMP
WorkDir=C:\TEST
SwapDir=C:\TEMP2
```

Execution

EDIFIC SAMPLE.SC

Results

The file APPLIC.INI will be modified by erasing the key WorkDir.

File APPLIC.INI (result):

```
[Begin]
WinPos = 32,200,500,639
AutoRun = 0
[Begin]
WinPos = 32,200,500,639
AutoRun = 0
[Directory]
TempDir=C:\TEMP
SwapDir=C:\TEMP2
```

Comments

The WorkDir key has been removed.
The [Test] section didn't exists.
You can also use parameters in DeleteKey without quotations if data doesn't contains spaces. It doesn't matter about case attribute on matching key or section text (i.e. is the same *DeleteKey "Directory" "WORKDIR"* and *DeleteKey "direcTORY" "workDIR"*)

AfterLine

Syntax

AfterLine "*lineref*" Insert "*lineput*" [CaseSen] [Ident] [NoRep]

Also

AFTLIN "*lineref*" INSERT "*lineput*" [CaseSen] [Ident] [NoRep]
DespuesDeLinea "*lineref*" Inserta "*lineput*" [CaseSen] [Ident] [NoRep]

Parameters

Parameter	***Description***
Lineref	Is a reference string
Lineput	Is the string to insert as a line
CaseSen	Optional. This keyword tells Edific that the match with the *lineref* must be case sensitive
Ident	Optional. This keyword tells Edific that the match must be exact. If this word is not specified, AfterLine matches when beginning of any line is the same as *lineref*
NoRep	Optional. This keyword tells Edific not to insert the line if *lineput* is already after the line matching the conditions

Operation

Inserts the line *lineput* after a line *lineref* that match the conditions.
AfterLine looks all the lines of the current loaded file for the lines *lineref*.
The conditions are determined by the flag keywords (CaseSen, Ident, NoRep).
If **CaseSen** is specified, *lineref* must match case sensitive. If **Ident** is not specified, the match is valid if any line has the same *lineref* first string.
If **NoRep** is specified, the insertion of the line will be not done if the *lineput* already exists after the *lineref* line.

Related commands

AfterLineContaining, BeforeLine , BeforeLineContaining, DeleteLine, DeleteLineContaining, ReplaceLine, ReplaceLineContaining, ReplaceString
Working with Lines and Substrings

Samples

Sample: Inserting lines after

Inserting lines after

Sample of

AfterLine

Script and Files
File SAMPLE.SC:

```
' Script to add lines
LoadFile "C:\TEST\ODYSSEY.TXT"
AfterLine "piling THEM" Insert "covered smoothly with rugs"
AfterLine "piling them" Insert "covered smoothly with rugs" NoRep
AfterLine "piling them" Insert "(this line cannot appear)" Ident
AfterLine "PILING them" Insert "(this line ALSO cannot appear)" CaseSen
SaveFile
End
```

File ODYSSEY.TXT:

```
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
piling them with lovely purple blankets
and thick woolen cloaks on top of all.
```

Execution

EDIFIC SAMPLE.SC

Results

The file ODYSSEY.TXT modified with the AfterLine commands.

File ODYSSEY.TXT (result):

```
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
```

Comments

Let's see what's happend with the AfterLine commands:
The first AfterLine matches the fourth line and puts the corresponding line after
The second, also matches the fourth line, but doesn't puts any line because **NoRep** flag is specified, and the "covered..." line was already there by the first command.
The third AfterLine doesn't match any line to work because flag **Ident** is specified. This means that the line must match exactly with "piling them", not only the firsts characters.
The fourth command also doesn't match because case is not the same as the fourth line.

Even this samples have one or none flags at the end of AfterLine commands, you can use all three flags in combinations to do all operations you would.

AfterLineContaining

Syntax

AfterLineContaining "*substringref*" Insert "*lineput*" [CaseSen] [NoRep]

Also

AFTLINCON "*substringref*" INSERT "*lineput*" [CaseSen] [NoRep]
DespuesDeLineaConteniendo "*substringref*" Inserta "*lineput*" [CaseSen] [NoRep]

Parameters

Parameter	***Description***
Substringref	Is a reference string
Lineput	Is the string to insert as a line
CaseSen	Optional. This keyword tells Edific that the match with the *substringref* must be case sensitive
NoRep	Optional. This keyword tells Edific not to insert the line if *lineput* is already after the line matching the conditions

Operation

Inserts the line *lineput* after a line that contains a string *substringref* and matches the conditions.
If **CaseSen** is specified, *substringref* must match case sensitive.
If **NoRep** is specified, the insertion of the line will be not done if the *lineput* already exists after the corresponding matching line.

Related commands

AfterLine, BeforeLine, BeforeLineContaining, DeleteLine, DeleteLineContaining, ReplaceLine, ReplaceLineContaining, ReplaceString
Working with Lines and Substrings

Samples

Sample: Inserting lines after containing

Inserting lines after containing

Sample of

AfterLineContaining

Script and Files
File SAMPLE.SC:

```
' Script to add lines
LoadFile "C:\TEST\ODYSSEY.TXT"
AfterLineContaining "lovely purple" Insert "covered smoothly with rugs"
AfterLineContaining "lovely purple" Insert "covered smoothly with rugs" NoRep
AfterLineContaining "lovely PurPlE" Insert "(this line cannot appear)" CaseSen
SaveFile
End
```

File ODYSSEY.TXT:

```
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
piling them with lovely purple blankets
and thick woolen cloaks on top of all.
```

Execution

EDIFIC SAMPLE.SC

Results

The file ODYSSEY.TXT modified with the AfterLineContaining commands.

File ODYSSEY.TXT (result):

```
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
```

Comments

Let's see what's happend with the AfterLineContaining commands:
The first AfterLineContaining matches the fourth line (because has an substring that is "lovely purple") and puts the corresponding line after.
The second, also matches the fourth line, but doesn't puts any line because **NoRep** flag is specified, and the "covered..." line was already there by the first command.
The third command doesn't match any line because case in "lovely PurPlE" isn't the same as the substring in the fourth line.

Even this samples have one or none flags at the end of AfterLineContaining commands, you can use all two flags in combinations to do all operations you would.

BeforeLine

Syntax

BeforeLine "*lineref*" Insert "*lineput*" [CaseSen] [Ident] [NoRep]

Also

BEFLIN "*lineref*" INSERT "*lineput*" [CaseSen] [Ident] [NoRep]
AntesDeLinea "*lineref*" Inserta "*lineput*" [CaseSen] [Ident] [NoRep]

Parameters

Parameter	***Description***
Lineref	Is a reference string
Lineput	Is the string to insert as a line
CaseSen	Optional. This keyword tells Edific that the match with the *lineref* must be case sensitive
Ident	Optional. This keyword tells Edific that the match must be exact. If this word is not specified, BeforeLine matches when beginning of any line is the same as *lineref*
NoRep	Optional. This keyword tells Edific not to insert the line if *lineput* is already before the line matching the conditions

Operation

Inserts the line *lineput* before a line *lineref* that match the conditions.
BeforeLine looks all the lines of the current loaded file for the lines *lineref*.
The conditions are determined by the flag keywords (CaseSen, Ident, NoRep).
If **CaseSen** is specified, *lineref* must match case sensitive. If **Ident** is not specified, the match is valid if any line has the same *lineref* first string.
If **NoRep** is specified, the insertion of the line will be not done if the *lineput* already exists before the *lineref* line.

Related commands

BeforeLineContaining, AfterLine, AfterLineContaining, DeleteLine, DeleteLineContaining, ReplaceLine, ReplaceLineContaining, ReplaceString
Working with Lines and Substrings

Samples

Sample: Inserting lines before

Inserting lines before

Sample of

BeforeLine

Script and Files

File SAMPLE.SC:

```
' Script to add lines
LoadFile "C:\TEST\ODYSSEY.TXT"
BeforeLine "piling THEM" Insert "to range beds under the sun-porch,"
BeforeLine "piling them" Insert "to range beds under the sun-porch," NoRep
BeforeLine "piling them" Insert "(this line cannot appear)" Ident
BeforeLine "PILING them" Insert "(this line ALSO cannot appear)" CaseSen
SaveFile
End
```

File ODYSSEY.TXT:

```
At his word Helen of Argos
ordered her house-maidens
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
```

Execution

EDIFIC SAMPLE.SC

Results

The file ODYSSEY.TXT modified with the BeforeLine commands.

File ODYSSEY.TXT (result):

```
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
```

Comments

Let's see what's happend with the BeforeLine commands:
The first BeforeLine matches the fourth line and puts the corresponding line before it.
The second, also matches the fourth line, but doesn't puts any line because **NoRep** flag is specified, and the "to range..." line was already there by the first command.
The third BeforeLine doesn't match any line to work because flag **Ident** is specified. This means that the line must match exactly with "piling them", not only the firsts characters.
The fourth command also doesn't match because case is not the same as the fourth line.

Even this samples have one or none flags at the end of AfterLine commands, you can use all three flags in combinations to do all operations you would.

BeforeLineContaining

Syntax

BeforeLineContaining "*substringref*" Insert "*lineput*" [CaseSen] [NoRep]

Also

BEFLINCON "*substringref*" INSERT "*lineput*" [CaseSen] [NoRep]
AntesDeLineaConteniendo "*substringref*" Inserta "*lineput*" [CaseSen] [NoRep]

Parameters

Parameter	***Description***
Substringref	Is a reference string
Lineput	Is the string to insert as a line
CaseSen	Optional. This keyword tells Edific that the match with the *substringref* must be case sensitive
NoRep	Optional. This keyword tells Edific not to insert the line if *lineput* is already before the line matching the conditions

Operation

Inserts the line *lineput* before a line that contains a string *substringref* and matches the conditions.
If **CaseSen** is specified, *substringref* must match case sensitive.
If **NoRep** is specified, the insertion of the line will be not done if the *lineput* already exists before the corresponding matching line.

Related commands

BeforeLine, AfterLine, AfterLineContaining, DeleteLine, DeleteLineContaining, ReplaceLine, ReplaceLineContaining, ReplaceString
Working with Lines and Substrings

Samples

Sample: Inserting lines before containing

Inserting lines before containing

Sample of

BeforeLineContaining

Script and Files
File SAMPLE.SC:

```
' Script to add lines
LoadFile "C:\TEST\ODYSSEY.TXT"
BeforeLineContaining "lovely purple" Insert "to range beds under the sun-porch,"
BeforeLineContaining "lovely purple" Insert "to range beds under the sun-porch," NoRep
BeforeLineContaining "lovely PurPlE" Insert "(this line cannot appear)" CaseSen
SaveFile
End
```

File ODYSSEY.TXT:

```
At his word Helen of Argos
ordered her house-maidens
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
```

Execution

EDIFIC SAMPLE.SC

Results

The file ODYSSEY.TXT modified with the BeforeLineContaining commands.

File ODYSSEY.TXT (result):

```
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
```

Comments

Let's see what's happend with the BeforeLineContaining commands:
The first BeforeLineContaining matches the fourth line (because has an substring that is "lovely purple") and puts the corresponding line before.
The second, also matches the fourth line, but doesn't puts any line because **NoRep** flag is specified, and the "to range..." line was already there by the first command.
The third command doesn't match any line because case in "lovely PurPlE" isn't the same as the substring in the fourth line.

Even this samples have one or none flags at the end of BeforeLineContaining commands, you can use all two flags in combinations to do all operations you would.

DeleteLine

Syntax

DeleteLine "*lineref*" [CaseSen] [Ident]

Also

DELLIN "*lineref*" [CaseSen] [Ident]
EliminaLinea "*lineref*" [CaseSen] [Ident]

Parameters

Parameter	***Description***
Lineref	Is a reference string
CaseSen	Optional. This keyword tells Edific that the match with the *lineref* must be case sensitive
Ident	Optional. This keyword tells Edific that the match must be exact. If this word is not specified, DeleteLine matches when beginning of any line is the same as *lineref*

Operation

Erases any line that matches with *lineref* and the specified flags.
The conditions are determined by the flag keywords (CaseSen, Ident).
If **CaseSen** is specified, *lineref* must match case sensitive. If **Ident** is not specified, the match is valid if any line has the same *lineref* first string.

Related commands

DeleteLineContaining, BeforeLine, BeforeLineContaining, AfterLine, AfterLineContaining, ReplaceLine, ReplaceLineContaining, ReplaceString
Working with Lines and Substrings

Samples

Sample: Deleting lines

Deleting lines

Sample of

DeleteLine

Script and Files
File SAMPLE.SC:

```
' Script to delete lines
LoadFile "C:\TEST\ODYSSEY.TXT"
DeleteLine "ordered her"
DeleteLine "Piling Them" CaseSen
DeleteLine "covered smoothly" Ident
SaveFile
End
```

File ODYSSEY.TXT:

```
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
```

Execution

EDIFIC SAMPLE.SC

Results

The file ODYSSEY.TXT modified with the DeleteLine commands.

File ODYSSEY.TXT (result):

```
At his word Helen of Argos
to range beds under the sun-porch,
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
```

Comments

Let's see what's happend with the DeleteLine commands:
The first DeleteLine matches the second line and erases it.
The second doesn't match anything because has enabled the case sensitive
The third DeleteLine doesn't match any line because flag **Ident** is specified. This means that the line must match exactly with "covered smoothly", not only the firsts characters.

Even this samples have one or none flags at the end of DeleteLine commands, you can use all two flags in combinations to do all operations you would.

DeleteLineContaining

Syntax

DeleteLineContaining "*substringref*" [CaseSen]

Also

DELLINCON "*substringref*" [CaseSen]
EliminaLineaConteniendo "*substringref*" [CaseSen]

Parameters

Parameter	***Description***
Substringref	Is a reference string
CaseSen	Optional. This keyword tells Edific that the match with the *substringref* must be case sensitive

Operation

Erases any line that contains the *substringref*.
If **CaseSen** is specified, *substringref* must match case sensitive.

Related commands

DeleteLine, BeforeLine, BeforeLineContaining, AfterLine, AfterLineContaining, ReplaceLine, ReplaceLineContaining, ReplaceString
Working with Lines and Substrings

Samples

Sample: Deleting lines containing substrings

Deleting lines containing substrings

Sample of

DeleteLineContaining

Script and Files
File SAMPLE.SC:

```
' Script to delete lines
LoadFile "C:\TEST\ODYSSEY.TXT"
DeleteLineContaining "ORDERED her"
DeleteLineContaining "With LoveLY" CaseSen
SaveFile
End
```

File ODYSSEY.TXT:

```
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
```

Execution

EDIFIC SAMPLE.SC

Results

The file ODYSSEY.TXT modified with the DeleteLineContaining commands.

File ODYSSEY.TXT (result):

```
At his word Helen of Argos
to range beds under the sun-porch,
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
```

Comments

Let's see what's happend with the DeleteLineContaining commands:
The first DeleteLineContaining matches the second line and erases it.
The second doesn't match anything because has enabled the case sensitive

ReplaceLine

Syntax

ReplaceLine "*linerefl*" By "*lineput*" [CaseSen] [Ident]

Also

REPLIN "*linerefl*" BY "*lineput*" [CaseSen] [Ident]
ReemplazaLinea "*linerefl*" Por "*lineput*" [CaseSen] [Ident]

Parameters

Parameter	***Description***
Lineref	Is the replaced string
Lineput	Is the replacing string
CaseSen	Optional. This keyword tells Edific that the match with the *lineref* must be case sensitive
Ident	Optional. This keyword tells Edific that the match must be exact. If this word is not specified, ReplaceLine matches when beginning of any line is the same as *lineref*

Operation

Replaces the line *lineref* by *lineput*. Seeks *lineref* in the loaded file and replaces all ocurrences by *lineput*. Flag keywords modify the matches.

If **CaseSen** is specified, *lineref* must match case sensitive. If **Ident** is not specified, the match is valid only if any line has the same *lineref* first string.

Related commands

ReplaceLineContaining, ReplaceString, BeforeLineContaining, BeforeLine, AfterLine, AfterLineContaining, DeleteLine, DeleteLineContaining
Working with Lines and Substrings

Samples

Sample: Replacing lines

Replacing lines

Sample of

ReplaceLine

Script and Files

File SAMPLE.SC:

```
' Script to delete lines
LoadFile "C:\TEST\ODYSSEY.TXT"
ReplaceLine "BIGS a" By "covered smoothly with rugs"
ReplaceLine "ORDERED her" By "(This line cannot appear)" CaseSen
ReplaceLine "At his word" By "(This line ALSO cannot appear)" Ident
SaveFile
End
```

File ODYSSEY.TXT:

```
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
piling them with lovely purple blankets
bigs and warms
and thick woolen cloaks on top of all.
```

Execution

EDIFIC SAMPLE.SC

Results

The file ODYSSEY.TXT modified with the ReplaceLine commands.

File ODYSSEY.TXT (result):

```
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
```

Comments

Let's see what's happend with the ReplaceLine commands:
The first ReplaceLine matches the fifth line and substitutes it. Note that It doesn't care about case here.
The second command doesn't match anything because the case sensitive flag is active
The third ReplaceLine also doesn't match any line, because the **Ident** flag is active.

ReplaceLineContaining

Syntax

ReplaceLineContaining "*substringref*" By "*lineput*" [CaseSen]

Also

REPLINCON "*substringref*" BY "*lineput*" [CaseSen]
ReemplazaLineaConteniendo "*substringref*" Por "*lineput*" [CaseSen]

Parameters

Parameter	***Description***
substringref	Is the substring the function looks for
Lineput	Is the replacing string
CaseSen	Optional. This keyword tells Edific that the match with the *substringref* must be case sensitive

Operation

Replaces the line that contains the substring *substringref* by *lineput*. Seeks *substringref* in the loaded file and replaces the entire line that contains this substring by the new line *lineput*. Flag keywords modify the matches.

If **CaseSen** is specified, *substringref* must match with case sensitive.

Related commands

ReplaceLine, ReplaceString, BeforeLineContaining, BeforeLine, AfterLine, AfterLineContaining, DeleteLine, DeleteLineContaining
Working with Lines and Substrings

Samples

Sample: Replacing lines containing substrings

Replacing lines containing substrings

Sample of

ReplaceLineContaining

Script and Files
File SAMPLE.SC:

```
' Script to delete lines
LoadFile "C:\TEST\ODYSSEY.TXT"
ReplaceLineContaining "AND wA" By "covered smoothly with rugs"
ReplaceLineContaining "HER house-mai" By "(This line cannot appear)" CaseSen
SaveFile
End
```

File ODYSSEY.TXT:

```
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
piling them with lovely purple blankets
bigs and warms
and thick woolen cloaks on top of all.
```

Execution

EDIFIC SAMPLE.SC

Results

The file ODYSSEY.TXT modified with the ReplaceLineContaining commands.

File ODYSSEY.TXT (result):

```
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
```

Comments

Let's see what's happend with the ReplaceLineContaining commands:
The first ReplaceLineContaining command matches the fifth line and substitutes it. Note that It doesn't care about case here.
The second command doesn't match anything because the case sensitive flag is active.

ReplaceString

Syntax

ReplaceLineContaining "*substringref*" By "*substringput*" [CaseSen]

Also

REPSTR "*substringref*" BY "*substringput*" [CaseSen]
Reemplaza "*substringref*" Por "*substringput*" [CaseSen]

Parameters

Parameter	***Description***
substringref	Is the substring the function looks for
substringput	Is the replacing string
CaseSen	Optional. This keyword tells Edific that the match with the *substringref* must be case sensitive

Operation

Replaces all the specified substing *substringref* by the substring specified. Note that doesn't replace all the line like the other functions. The substring will be replaced every time it's found, even when multiple matches in the same line.

If **CaseSen** is specified, *substringref* must match with case sensitive.

Related commands

ReplaceLine, ReplaceLineContaining, BeforeLineContaining, BeforeLine, AfterLine, AfterLineContaining, DeleteLine, DeleteLineContaining
Working with Lines and Substrings

Samples

Sample: Replacing substrings

Replacing substrings

Sample of

ReplaceString

Script and Files

File SAMPLE.SC:

```
' Script to delete lines
LoadFile "C:\TEST\ODYSSEY.TXT"
ReplaceString "TH" By "X"
ReplaceString "red" By "(RED)" CaseSen
ReplaceString "OF" By "(OF)" CaseSen
SaveFile
End
```

File ODYSSEY.TXT:

```
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
```

Execution

EDIFIC SAMPLE.SC

Results

The file ODYSSEY.TXT modified with the ReplaceString commands.

File ODYSSEY.TXT (result):

```
At his word Helen of Argos
orde(RED) her house-maidens
to range beds under Xe sun-porch,
piling Xem wiX lovely purple blankets
cove(RED) smooXly wiX rugs
and Xick woolen cloaks on top of all.
```

Comments

Let's see what's happend with the ReplaceString commands:
The first ReplaceString command matches "th" substrings and substitutes this by "X". Note that multiple matches also have been treated.
The second command match "red" because the case sensitive flag is active and the string is the same.
Note that in the third command case sense makes function do nothing.

AfterLineNumber

Syntax

AfterLineNumber *number* Insert *"lineput"*

Also

AFTLINNUM *number* INSERT *"lineput"*
DespuesDeLineaNumero *number* Inserta *"lineput"*

Parameters

Parameter	***Description***
Number	Is an positive integer number
Lineput	Is the string to insert

Operation

Puts the line *lineput* after the one that is the *number*-th line in the current loaded file.
The *number* parameter must be greater than 0 and minor or equal than the number of lines of the file at the moment the command is executed. Othercase is ignored
Note that first line in a file is line 1.

Related commands

AfterEndInsert, BeforeBeginInsert, BeforeLineNumber, DeleteLineNumber
Working with Lines and Linenumbers

Samples

Sample: Adding lines with linenumbers

Adding lines with linenumbers

Sample of

AfterLineNumber, BeforeLineNumber

Script and Files

File SAMPLE.SC:

```
' Script to delete lines
LoadFile "C:\TEST\ODYSSEY.TXT"
AfterLineNumber 6 Insert "(this line cannot appear)"
AfterLineNumber 2 Insert "ordered her house-maidens"
AfterLineNumber 6 Insert "(this line MUST appear at the end)"
BeforeLineNumber 6 Insert 'this line appears before "and thick.. " line'
SaveFile
End
```

File ODYSSEY.TXT:

```
At his word Helen of Argos
to range beds under the sun-porch,
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
```

Execution

EDIFIC SAMPLE.SC

Results

The file ODYSSEY.TXT modified with the AfterLineNumber and BeforeLineNumber commands.

File ODYSSEY.TXT (result):

```
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
piling them with lovely purple blankets
covered smoothly with rugs
this line appears before "and thick.. " line
and thick woolen cloaks on top of all.
(this line MUST appear at the end)
```

Comments

Let's see what's happend with the AfterLineNumber commands:
The first AfterLineNumber command do not inserts any line because the file only has five lines. When the second command is executed, then inserting the new line give to the file one more line. It's for this reason the third command can insert now a line after the sixth.

The BeforeLineNumber command works like AfterLineNumber. Note also that you can use diferent character delimiter (‘, *, @) in place of (“).

BeforeLineNumber

Syntax

BeforeLineNumber *number* Insert *"lineput"*

Also

BEFLINNUM *number* INSERT *"lineput"*
AntesDeLineaNumero *number* Inserta *"lineput"*

Parameters

Parameter	***Description***
Number	Is an positive integer number
Lineput	Is the string to insert

Operation

Puts the line *lineput* before the one that is the *number*-th line in the current loaded file.
The *number* parameter must be greater than 0 and minor or equal than the number of lines of the file at the moment the command is executed. Othercase is ignored
Note that first line in a file is line 1.

Related commands

BeforeBeginInsert, AfterEndInsert, AfterLineNumber, DeleteLineNumber
Working with Lines and Linenumbers

Samples

Sample: Adding lines with linenumbers

AfterEndInsert

Syntax

AfterEndInsert *"lineput"*

Also

AFTENDINS *"lineput"*
InsertaFinal *"lineput"*

Parameters

Parameter	***Description***
Lineput	Is the string to insert

Operation

Puts the line *lineput* at the end of the current edited file.

Related commands

BeforeBeginInsert, BeforeLineNumber, AfterLineNumber, DeleteLineNumber
Working with Lines and Linenumbers

Samples

Sample: Adding lines at end or top

Adding lines at end or top

Sample of

AfterEndInsert, BeforeBeginInsert

Script and Files
File SAMPLE.SC:

```
' Script to delete lines
LoadFile "C:\TEST\ODYSSEY.TXT"
BeforeBeginInsert "(This is the first line)"
AfterEndInsert "(This is the final line)"
SaveFile
End
```

File ODYSSEY.TXT:

```
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
```

Execution

EDIFIC SAMPLE.SC

Results

The file ODYSSEY.TXT modified with the AfterEndInsert and BeforeBeginInsert commands.

File ODYSSEY.TXT (result):

```
(This is the first line)
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
(This is the final line)
```

Comments

AfterEndInsert has put its line at the end of the working file.
BeforeBeginInsert has put its line at top.

BeforeBeginInsert

Syntax

BeforeBeginInsert *"lineput"*

Also

BEFBEGINS *"lineput"*
InsertaPrincipio *"lineput"*

Parameters

Parameter	***Description***
Lineput	Is the string to insert

Operation

Puts the line *lineput* at the top of the current edited file.

Related commands

AfterEndInsert, BeforeLineNumber, AfterLineNumber, DeleteLineNumber
Working with Lines and Linenumbers

Samples

Sample: Adding lines at end or top

DeleteLineNumber

Syntax

DeleteLineNumber *number*

Also

DELLINNUM *number*
EliminaLineaNumero *number*

Parameters

Parameter	***Description***
Number	Is an positive integer number

Operation

Erases the line *number*-th in the current loaded file.
The *number* parameter must be greater than 0 and minor or equal than the number of lines of the file at the moment the command is executed. Othercase is ignored
Note that first line in a file is line 1.

Related commands

BeforeBeginInsert, AfterEndInsert, AfterLineNumber, BeforeLineNumber
Working with Lines and Linenumbers

Samples

Sample: Deleting lines with linenumbers

Deleting lines with linenumbers

Sample of

DeleteLineNumber

Script and Files
File SAMPLE.SC:

```
' Script to delete lines
LoadFile "C:\TEST\ODYSSEY.TXT"
DeleteLineNumber 4
DeleteLineNumber 4
SaveFile
End
```

File ODYSSEY.TXT:

```
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
```

Execution

EDIFIC SAMPLE.SC

Results

The file ODYSSEY.TXT modified with the DeleteLineNumber commands.

File ODYSSEY.TXT (result):

```
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
and thick woolen cloaks on top of all.
```

Comments

Here the file has been modified by the two DeleteLineNumber. Note that when the first command is executed, the fourth line will be now the fifth in the original file.

Repeat

Syntax

Repeat *number*

Also

REP *number*
Repetir *number*

Parameters

Parameter	***Description***
Number	Repetitions for the structure

Operation

Is the begin of any Repeat…End Repeat structure. The value is the number of iterations the structure will do.

Related commands

EndRepeat
Working with Execution Steps

Samples

Sample: Doing iterations

EndRepeat

Syntax

EndRepeat

Also

ENDREP
FinRepetir

Parameters

None

Operation

Is the end of any Repeat…End Repeat structure. At the EndRepeat, control jumps to the following command after the nearest Repeat in the script.

Related commands

Repeat
Working with Execution Steps

Samples

Sample: Doing iterations

End

Syntax

End

Also

Fin

Parameters

None

Operation

Is the end of the script. Any code or command after this sentence will not be executed.

Samples

Sample: Displaying messages and Ending Scripts

Doing Iterations

Sample of

Repeat, EndRepeat

Script and Files

File SAMPLE.SC:

```
' Script to do iterations
LoadFile "C:\TEST\ODYSSEY.TXT"
Repeat 3
  BeforeLine "ordered" Insert "(this line was included by the 1st repeat)"
  Repeat 2
     AfterLine "ordered" Insert "(this line was included by the 2nd repeat)"
  EndRepeat
EndRepeat
SaveFile
End
```

File ODYSSEY.TXT:

```
At his word Helen of Argos
ordered her house-maidens
to range beds under the sun-porch,
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
```

Execution

EDIFIC SAMPLE.SC

Results

The file ODYSSEY.TXT modified with the BeforeLine and AfterLine commands up to 3 and 6 times respectively.

File ODYSSEY.TXT (result):

```
At his word Helen of Argos
(this line was included by the 1st repeat)
(this line was included by the 1st repeat)
(this line was included by the 1st repeat)
ordered her house-maidens
(this line was included by the 2nd repeat)
(this line was included by the 2nd repeat)
(this line was included by the 2nd repeat)
(this line was included by the 2nd repeat)
(this line was included by the 2nd repeat)
(this line was included by the 2nd repeat)
to range beds under the sun-porch,
piling them with lovely purple blankets
covered smoothly with rugs
and thick woolen cloaks on top of all.
```

Comments

All code between the outer block Repeat and EndRepeat has been executed for 3 times. Inside this block it was another Repeat block that also has been executed 2 times in each iteration of the first Repeat. This makes 6 times (Its correct this ? Well, now you see we know to multiply, isn't-it ? **:-)**

Displaying Messages and Ending Scripts

Sample of

End, Display

Script and Files
File SAMPLE.SC:

```
Display "This message will be shown"
End
Display "This not !"
```

Execution

EDIFIC SAMPLE.SC

Results

The display of the screen will be like this:

```
This message will be shown
```

Display

Syntax

Display “*message*”

Also

DIS “*message*”
Mensaje “*message*”

Parameters

Parameter	***Description***
Message	A string that is the message to show

Operation

Puts out the *message* string to the default standard output.
If the flags /l or /a in the Edific invocation are set, output will be to the log file.

Samples

Sample: Displaying messages and Ending Scripts

InsertRegistryKey

Syntax

InsertRegistryKey “*keyreg*”

Also

INSREGKEY “*keyreg*”
InsertaClaveRegistro “*keyreg*”

Parameters

Parameter	***Description***
Keyreg	Is a string refering the key to create in the registry. Must be specified from root keys

Operation

Creates a key in the registry. The *keyreg* string specifies the key entirely from the root key. A valid example for this can be:

HKEY_CURRENT_USER\Control Panel\Draft

When InsertRegistryKey is executed with this key, Draft key in HKEY_CURRENT_USER\Control Panel will be created if did not exists before.

The key is created with *Generic Class* Class-name and default permissions.

You can use the following keywords as abreviations.

Warning: Be careful about the use of registry. Any mistake could destroy configuration information of the computer. Please make test in non production areas.

Related commands

InsertRegistryValue, DeleteRegistryKey, DeleteRegistryValue, ChangeRegistryValue
Working with Registy

Samples

Sample: Creating keys and setting values

Creating Keys and setting values

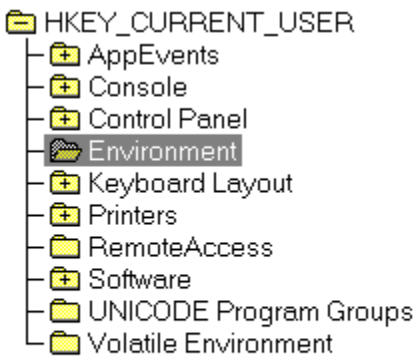
Sample of

InsertRegistryKey, InsertRegistryValue

Script and Files
File SAMPLE.SC:

```
' Script to manage registry
InsertRegistryValue "HKCU\Environment\TestKey\Param Number" Dword 1003644
InsertRegistryValue "HKCU\Environment\TestKey\Param NumberBigEndian" NumeroBE 1003644
InsertRegistryKey "HKEY_CURRENT_USER\Environment\TestKey2"
InsertRegistryValue "HKCU\Environment\TestKey\Param Text" String "A string message"
InsertRegistryValue "HKCU\Environment\TestKey\Param ExpText" ExpString "An expanded string
message"
InsertRegistryValue "HKCU\Environment\TestKey\Param MultiText" MultiString "The first
line.The second line..Still the second line.The third line "
InsertRegistryValue "HKCU\Environment\TestKey\Param Data" Binary "01F0E578 9FA03E80
910087E9"
End
```

Registry View:

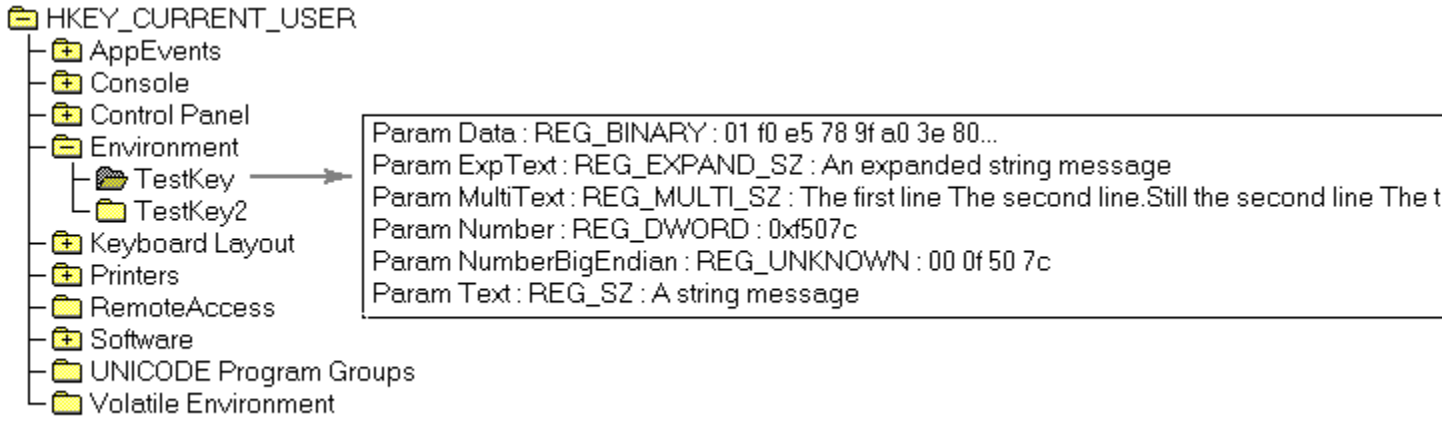


Execution

EDIFIC32 SAMPLE.SC

Results

Registry View:



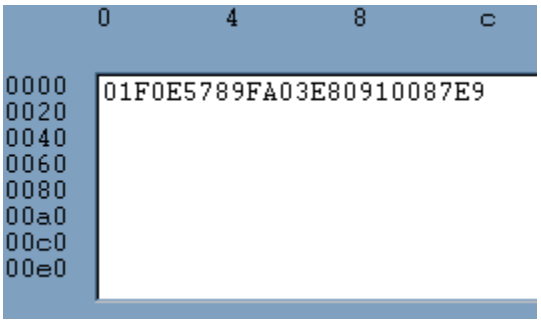
Comments

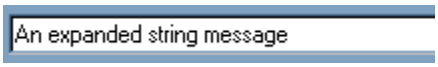
This is a sample of setting all type of data values into the registry. First, the HKCU means HKEY_CURRENT_USER, this is the root key.
At the first command, the TestKey doesn't exists, but it's not a problem to InsertRegistryValue because it creates all key needed to set the value. In case the value already were in the registry an was another data type, the command also has no problem with that and changes the type and value to the specifien in parameters.
You can see the sample of inserting multiple strings (MultiString type). The lines are delimited by the "." character in the text, and if you put two points together "..", this will be converted in a real point char in the line and doesn't become a line separator. See the result in the graphic above.
At the binary data inserting command, you can see the data is defined like hexadecimal values in a string. Spaces in the line will be ignored. If you put here any invalid character (valid characters are from 0 to 9 and from A to F) will become a 0 value. You can see the result value in the graphic.

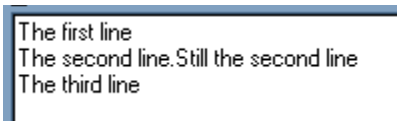
The InsertRegistryKey command only creates an empty subkey.
Note that is not necessary to open or edit any file to operate with registry. These functions are independent.

The Environment key has their corresponding Path values, etc. But no any subkey.

The TestKey2 key was created empty, without any value inside.


0 4 8 c
0000 01F0E5789FA03E80910087E9
0020
0040
0060
0080
00a0
00c0
00e0


An expanded string message


The first line
The second line.Still the second line
The third line

1003644


A string message

This type is not recognized by Regedt32 or Regedit programs

InsertRegistryValue

Syntax

InsertRegistryValue “*valreg*” *DataType* “*info*”

Also

INSREGVAL “*valreg*” *DataType* “*info*”
InsertaValorRegistro “*valreg*” *DataType* “*info*”

Parameters

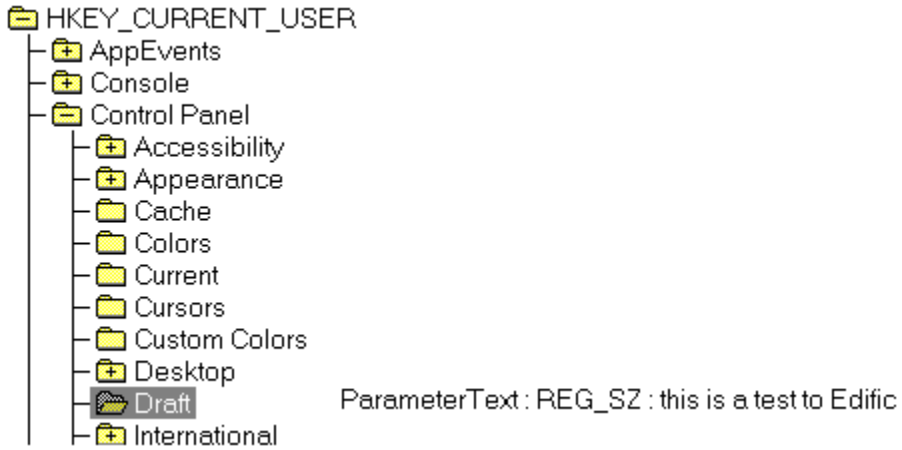
Parameter	***Description***
Valreg	Is a string refering the value or attribute to set in the registry. Must be specified from root keys
DataType	Identifies the type of the value in the registry. Is a keyword from the set { String, ExpString, MultiString, Binary, Dword, DwordBE}
Info	Is a string with the information to insert

Operation

Set the data of any value in the registry with the *Info* data.
To do this operation, the command needs to have permisions to access the value key.
The *valreg* is a string referencing the value entirely from the corresponding root key. For example, you can specify

HKEY_CURRENT_USER\Control Panel\Draft\ParameterText

Where ParameterText is the value and the rest is the location of the attribute. You can see this in the following view:



When InsertRegistryValue is executed, Draft key in HKEY_CURRENT_USER\Control Panel will be created if did not exists before.

If the attribute doesn’t exists, it will be created with the corresponding data type and information data.
If the attribute already exists in the registry, the data type and info specified will be set in. Note that the data type is set even the attribute had another.

For specify the location of the value, you can use the following keywords as abreviations.

The data types of the information you can set with InsertRegistryValue are:

Keyword	*Is a Data type*	*And the info parameter is*
String	REG_SZ	A string with the data
ExpString	REG_EXPAND_SZ	A string with the data
MultiString	REG_MULTI_SZ	A string with text lines delimited by the “.” Character
Binary	REG_BINARY	A string with hexadecimal values of the binary data. Spaces are ignored
DWord	REG_DWORD	A string with a number in decimal base
DwordBE	REG_DWORD_BIG_ENDIAN	A string with a number in decimal base

If the Data type is not recognized, the value will have a REG_NONE type.

The *Info* parameter has data format string. Examples of this type of data can be:

Keyword	*Sample of Info*	*Comments*
String	“A sample Message”	
ExpString	“%SystemRoot%\config”	Contains environment variables
MultiString	“DEC..DLL.NEXT..DLL”	The “..” sequence is interpreted exactly as a “.” in the final data, not a separator.
Binary	“03 F9 8E 9E A0 19 GG”	Data in hexa. The spaces will be ignored. The last value is invalid: will be converted to 00
DWord	196738	A positive 32 bit number (you can use or not the “ delimiters)
DwordBE	196738	The same the above. The data will be put in the registry in the Big Endian format

Warning: Be careful about the use of registry. Any mistake could destroy configuration information of the computer. Please make test in non production areas.

Related commands

InsertRegistryKey, DeleteRegistryKey, DeleteRegistryValue, ChangeRegistryValue
Working with Registy

Samples

Sample: Creating keys and setting values

ChangeRegistryValue

Syntax

ChangeRegistryValue “*keyreg*” *DataType* “*searchinfo*” “*replaceinfo*” *ChangeType* [CaseSen]

Also

CHAREGVAL “*keyreg*” *DataType* “*searchinfo*” “*replaceinfo*” *ChangeType* [CaseSen]
CambiaValorRegistro “*keyreg*” *DataType* “*searchinfo*” “*replaceinfo*” *ChangeType* [CaseSen]

Parameters

Parameter	***Description***
Keyreg	Is a string refering the key to start changing values in the registry. Must be specified from root keys
DataType	Identifies the type of the value in the registry to search for. Is a keyword from the set {String, ExpString, MultiString, Binary, Dword, DwordBE}
SearchInfo	Is a string with the information to search in the registry values
ReplaceInfo	Is a string with the information to set in the registry values
ChangeType	Identifies the mode the change operation will act. This is a keyword from the set {PartialMatch&Replace, PartialMatch&Set, Ident}
CaseSen	Optional. Is a keyword that indicates the search will be with care on case.

Operation

Changes the data of any value in the registry beginning by the *keyreg* specified and all its subkeys, that has the same information data as the *searchinfo* parameter, has the same data type as indicated in *DataType* and complains the CaseSen flag if activated and the matching mode.

The function begins with *keyreg*. It seeks for any value at this key that matches the *SearchInfo* data in the way the *ChangeType* indicates. If match is found, then the data value is changed by the *ReplaceInfo* parameter also in the way set by *ChangeType*. In any case, the function will look for child subkeys and does the same for each value until all values are investigated.

The *keyreg* is a string referencing any key entirely from the corresponding root key. For example, you can specify

HKEY_CURRENT_USER\Control Panel\Draft

For specify the location of the value, you can use the following keywords as abreviations.

The data types of the information you can specify with ChangeRegistryValue are:

DataType Keyword	***Is a Data type on Registry***	***And the SearchInfo and ReplaceInfo parameters are***
String	REG_SZ	A string with the data
ExpString	REG_EXPAND_SZ	A string with the data
MultiString	REG_MULTI_SZ	A string with text lines delimited by the “.” Character
Binary	REG_BINARY	A string with hexadecimal values of the binary data. Spaces are ignored
DWord	REG_DWORD	A string with a number in decimal base
DwordBE	REG_DWORD_BIG_ENDIAN	A string with a number in decimal base

The *SearchInfo* and *ReplaceInfo* parameters have data format strings. Examples of this type of data can be:

DataType Keyword	***Sample of SearchInfo and ReplaceInfo***	***Comments***
String	“A sample Message”	

ExpString	“%SystemRoot%\config”	Contains environment variables
MultiString	“DEC..DLL.NEXT..DLL”	The “..” sequence is interpreted exactly as a “.” in the final data, not a separator.
Binary	“03 F9 8E 9E A0 19 GG”	Data in hexa. The spaces will be ignored. The last value is invalid: will be converted to 00
DWord	196738	A positive 32 bit number (you can use or not the “ delimiters)
DwordBE	196738	The same the above. The data will be put in the registry in the Big Endian format

The *ChangeType* keywords sets the way the matching and the replace will work. The three modes of change values are:

ChangeType mode	***You can use these keywords***	***How ChangeRegistryValue will act***
Match contained data and substitutes it	PartialMatch&Replace PM&R Contenido&Reemplaza	Search for *SearchInfo* contained in the data value and replaces it if match to the *ReplaceInfo* data. Note that only substitutes the *SearchInfo* data found.
Match contained data and sets new data	PartialMatch&Set PM&S Contenido&Inserta	Search for *SearchInfo* contained in the data value and changes all the data value to *ReplaceInfo* data. Note that in this mode, all data value will be discarded.
Match exact data and changes it	Ident IDEM Identico	Search for *SearchInfo* exactly in the data value and replaces it if match to the *ReplaceInfo* data. Note that only will be a match if the information is the same as *SearchInfo* data

Note that only the matches arise if the value has the same registry data type as specified by *DataType*.

ChangeRegistryValue is a powerful command to do changes, but also is dangerous if you make mistakes on params.

Warning: Be careful about the use of registry. Any mistake could destroy configuration information of the computer. Please make test in non production areas.

Related commands

InsertRegistryKey, DeleteRegistryKey, DeleteRegistryValue, InsertRegistryValue
Working with Registy

Samples

Sample: Making data changes to the registry

Making data changes to the registry

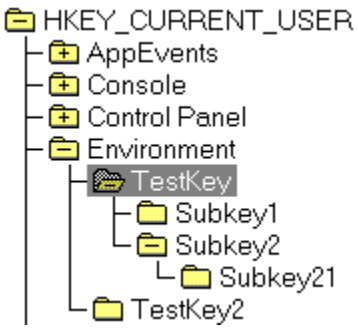
Sample of

ChangeRegistryValue

Script and Files
File SAMPLE.SC:

```
' Script to manage registry

ChangeRegistryValue "HKCU\Environment\TestKey" Binary "910087" "BACC" PM&R Casesen

ChangeRegistryValue "HKCU\Environment\TestKey" MultiString "Second line" "THIS CANNOT
APPEAR" PM&R Casesen
ChangeRegistryValue "HKCU\Environment\TestKey" MultiString "Second line" "NEW second line"
PM&R

ChangeRegistryValue "HKCU\Environment\TestKey" ExpString "An expanded str" "THIS CANNOT
APPEAR" Ident
ChangeRegistryValue "HKCU\Environment\TestKey" ExpString "An expanded STRING MESSage"
"THIS CANNOT APPEAR" Ident CaseSen
ChangeRegistryValue "HKCU\Environment\TestKey" ExpString "An expanded STRING MESSage"
"Replacing line by 'Ident' mode" Ident
ChangeRegistryValue "HKCU\Environment\TestKey" String "message" 'All strings with
"message" substring has been replaced to THIS line' PM&S

ChangeRegistryValue "HKCU\Environment\TestKey" Dword "1003644" "65" PM&R
ChangeRegistryValue "HKCU\Environment\TestKey" DwordBE "1003644" "66" PM&R

End
```

Registry View:

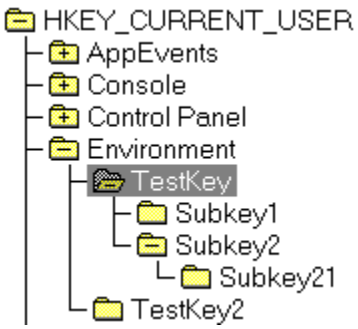


Execution

EDIFIC32 SAMPLE.SC

Results

Registry View:



Comments

The scripts shows a few samples of working with ChangeRegistryValue.
First, the HKCU means HKEY_CURRENT_USER, this is the root key.
All testings were done with HKEY_CURRENT_USER\Environment\TestKey as a start key.

The first command changed the data of the Param Data and Subkey1\ParamData1 by replacing the hexa codes "910087" by "BACC". Note that doesn't care about different data length. Also the **CaseSen** keyword is ignored here.
It's important to know that the comparation of binary data is done byte-to-byte. That is, if you want to replace "10087" in place of the byte-boundary "910087", that wouldn't work. See the results here.

The second command do not made anything because the search for info was beginning by "S" and the data value in Param MultiText has the "s" char, and the command has to match in Case sensitive.

The third command changes the data because doesn't care about case. Note that the data was found two times in the lines, so the replace was made also two times. See the results here

The fourth command doesn't make anything because **Ident** keyword has been specified and any value data is not identical as "An expanded str". Also the fifth, but this time by the **CaseSen** flag.

The sixth command changed the value in this way.

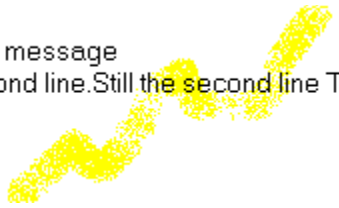
The seventh searched for the substring "message" in all the String data type values. See the results here.

The 8th and 9th commands changed the values of two values. Note that when working with numeric values (Dword and DwordBE), neither mode (*ChangeType)* nor case sensitive affects the search and replace, but it's mandatory to set any mode keyword. In all cases, working with numbers, the function works as if **Ident** keyword were specified.

Here you can see the results of the 8th command and the 9th command.

Note that is not necessary to open or edit any file to operate with registry. These functions are independent.

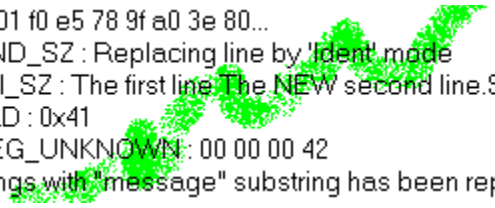
Param Data : REG_BINARY : 01 f0 e5 78 9f a0 3e 80...
Param ExpText : REG_EXPAND_SZ : An expanded string message
Param MultiText : REG_MULTI_SZ : The first line The second line.Still the second line The third line
Param Number : REG_DWORD : 0xf507c
Param NumberBigEndian : REG_UNKNOWN : 00 0f 50 7c
Param Text : REG_SZ : A string message



ParamData1 : REG_BINARY : 9f a0 3e 80 91 00 87 e9

AnotherText : REG_SZ : It's a message line

Param Data : REG_BINARY : 01 f0 e5 78 9f a0 3e 80...
Param ExpText : REG_EXPAND_SZ : Replacing line by 'ident' mode
Param MultiText : REG_MULTI_SZ : The first line The NEW second line.Still the NEW second line The third line
Param Number : REG_DWORD : 0x41
Param NumberBigEndian : REG_UNKNOWN : 00 00 00 42
Param Text : REG_SZ : All strings with "message" substring has been replaced to THIS line



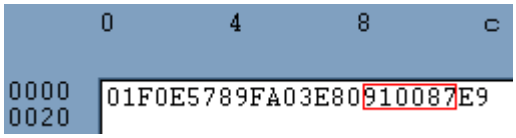
ParamData1 : REG_BINARY : 9f a0 3e 80 ba cc e9

AnotherText : REG_SZ : All strings with "message" substring has been replaced to THIS line

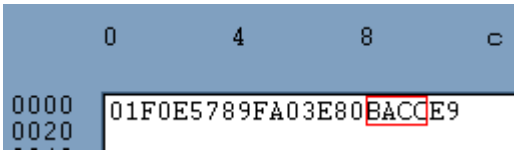
Results of *ChangeRegistryValue "HKCU\Environment\TestKey" Binary "910087" "BACC" PM&R Casesen*

HKEY_CURRENT_USER\Enviroment\TestKey\Param Data

Before Excution

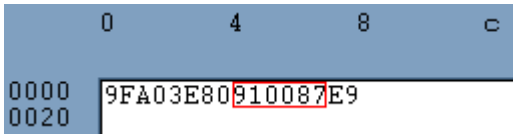


After Execution

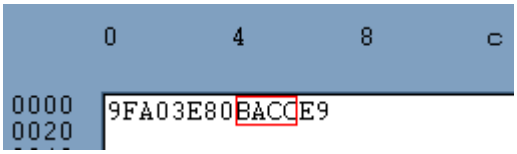


HKEY_CURRENT_USER\Enviroment\TestKey\Subkey1\ParamData1

Before Excution



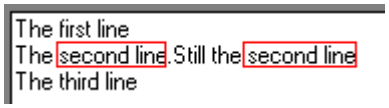
After Execution



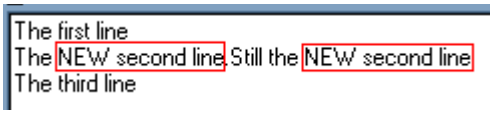
Results of *ChangeRegistryValue "HKCU\Environment\TestKey" MultiString "Second line" "NEW second line" PM&R*

HKEY_CURRENT_USER\Enviroment\TestKey\Param MultiText

Before Excution



After Execution



Results of *ChangeRegistryValue "HKCU\Environment\TestKey" ExpString "An expanded STRING MESSage" "Replacing line by 'Ident' mode" Ident*

HKEY_CURRENT_USER\Enviroment\TestKey\Param ExpText

Before Excution

After Execution

Results of *ChangeRegistryValue "HKCU\Environment\TestKey" String "message" 'All strings with "message" substring has been replaced to THIS line' PM&S*

HKEY_CURRENT_USER\Enviroment\TestKey\Param Text

Before Excution

After Execution

HKEY_CURRENT_USER\Enviroment\TestKey\Subkey1\Subkey21\AnotherText

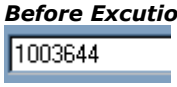
Before Excution

After Execution

Results of *ChangeRegistryValue "HKCU\Environment\TestKey" Dword "1003644" "65" PM&R*

HKEY_CURRENT_USER\Enviroment\TestKey\Param Number

Before Excution

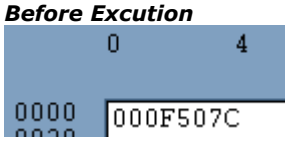


After Execution

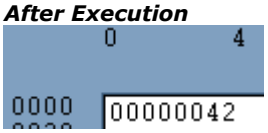
Results of *ChangeRegistryValue "HKCU\Environment\TestKey" DwordBE "1003644" "66" PM&R*

HKEY_CURRENT_USER\Enviroment\TestKey\Param NumberBigEndian

Before Excution



After Execution



(0x42 is 66 in decimal)

DeleteRegistryKey

Syntax

DeleteRegistryKey *"keyreg"*

Also

DELREGKEY *"keyreg"*
EliminaClaveRegistro *"keyreg"*

Parameters

Parameter	***Description***
keyreg	Is a string refering the key to delete in the registry. Must be specified from root keys

Operation

Delete the key *keyreg* and all its values and subkeys. If *keyreg* is a value in place a key, the commad will do nothing.
To do this operation, the command needs to have permisions to access the value key.
The *keyreg* is a string referencing the key entirely from the corresponding root key. For example, you can specify

HKEY_CURRENT_USER\Control Panel\Draft

For specify the location of the value, you can use the following keywords as abreviations.

Do not specify root keys alone in the command as a *keyreg*.

Warning: Be careful about the use of registry. Any mistake could destroy configuration information of the computer. Please make test in non production areas.

Related commands

InsertRegistryKey, InsertRegistryValue, DeleteRegistryValue, ChangeRegistryValue
Working with Registy

Samples

Sample: Deleting values and keys in the registry

Deleting values and keys in the registry

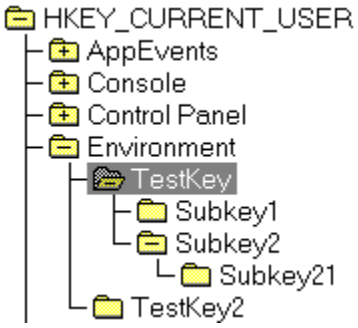
Sample of

DeleteRegistryKey, DeleteRegistryValue

Script and Files
File SAMPLE.SC:

```
' Script to manage registry

DeleteRegistryValue "HKCU\Environment\TestKey\Param Number"
DeleteRegistryKey "HKCU\Environment\TestKey\Subkey1"

End
```

Registry View:



Execution

EDIFIC32 SAMPLE.SC

Results

Registry View:

Comments

The first command erased the value "Param Number". See the difference between before and after the execution of the script.

The second command erased the **all** the data in the key "Subkey1". Also all subkeys has been deleted.

Note that is not necessary to open or edit any file to operate with registry. These functions are independent.

Param Data : REG_BINARY : 01 f0 e5 78 9f a0 3e 80...
Param ExpText : REG_EXPAND_SZ : An expanded string message
Param MultiText : REG_MULTI_SZ : The first line The second line.Still the second line The third line
Param NumberBigEndian : REG_UNKNOWN : 00 0f 50 7c
Param Text : REG_SZ : A string message

DeleteRegistryValue

Syntax

DeleteRegistryValue “*valreg*”

Also

DELREGVAL “*valreg*”
EliminaValorRegistro “*keyreg*”

Parameters

Parameter	***Description***
valreg	Is a string refering the value to delete in the registry. Must be specified from root keys

Operation

Delete the value *valreg*. If the *valreg* specifies a key, the command will do nothing.
To do this operation, the command needs to have permisions to access the value key.
The *valreg* is a string referencing the key entirely from the corresponding root key. For example, you can specify

HKEY_CURRENT_USER\Control Panel\Draft\ParameterText

Where ParameterText is the value and the rest is the location of the attribute. You can see this in the following view:

For specify the location of the value, you can use the following keywords as abreviations.

Warning: Be careful about the use of registry. Any mistake could destroy configuration information of the computer. Please make test in non production areas.

Related commands

InsertRegistryKey, InsertRegistryValue, DeleteRegistryKey, ChangeRegistryValue
Working with Registy

Samples

Sample: Deleting values and keys in the registry

Root Registry Keywords

Root key	***Short name***
HKEY_LOCAL_MACHINE	HKLM
HKEY_CLASSES_ROOT	HKCR
HKEY_CURRENT_USER	HKCU
HKEY_USERS	HKU
HKEY_DYN_DATA	HKDD
HKEY_PERFORMANCE_DATA	HKPD

Note that all root keys are not available in W95,W98,W3.x systems