ConfigurationEditor Pro

INI file editor software component

This manual covers both the FREE and the Pro versions of Configuration Editor, where functionality that is only available in the Pro version is marked with (Pro).

What is Configuration Editor

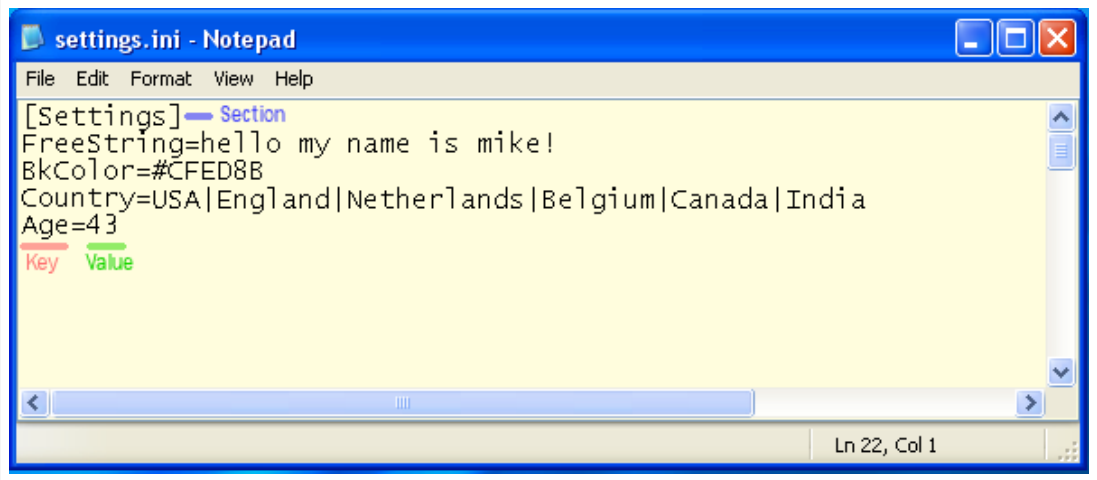
It is an INI file editor and Properties file editor. It can run either as standalone tool or as a component from another application software. You can bundle it with your own software, to relieve yourself from developing a configuration screen for your application.

Getting Started

While it is possible to customize ConfigEditor behavior, most users will find the defaults suitable to their needs.

For using the default behavior, put ConfigEditor.exe in your software folder, along with your software configuration INI or Properties file. Rename it to SETTINGS.INI or SETTINGS.PROPERTIES depending on the file type. Then run ConfigEditor.exe to edit your settings file.

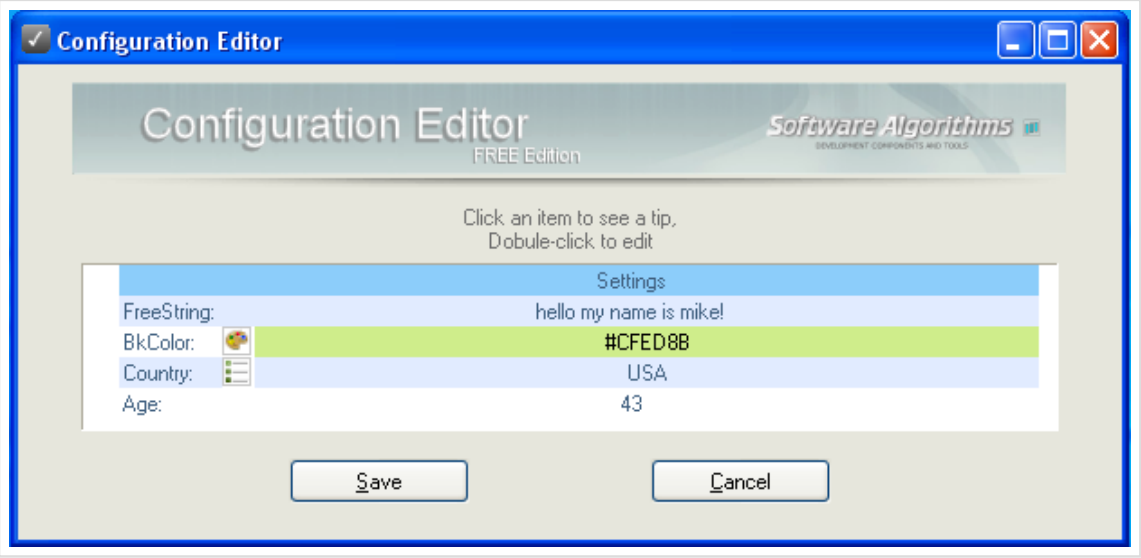
INI files are text files, they can be created and edited with any text editor, like Notepad.



Example:
By default, using a settings file that looks like this:

```
[Settings]
FreeString=hello my name is mike!
BkColor=#CFED8B
Country=USA|England|Netherlands|Belgium|Canada|India
Age=43
```

Will make ConfigEditor show a screen like this:



The user can edit a row value by either double-clicking anywhere on the row, or by selecting it using the keyboard arrow keys then pressing Enter. after editing the line, pressing Enter, or clicking outside the edit area will accept the changes. Pressing the Esc key will discard the changes.

Note that using the default behavior, ConfigEditor will auto-detect the type of each key and value in the following way:

```
[Settings]
```

This line is a section, so it will show as read-only line on the list (also called a Title) with the text "Settings".

```
FreeString=hello my name is mike!
```

Will show as another row on the list, with the key name "FreeString" shown on the first from left column, and the value "hello my name is mike!" will be shown on the second column. ConfigEditor will auto-detect it as a STRING type since the value is not in any format that ConfigEditor recognizes. When double clicked, STRING values are edited by a standard edit box where the user can type any text.

```
BkColor=#CFED8B
```

Since the value is formated as pound sign (#) with a 6 digit hex number, ConfigEditor will auto-detect it as a COLOR type, and will add a row that is best suited to describe and edit a color (the second column is colored using that color value, and when the line is double clicked, a color picker is shown)

```
Country=USA|England|Netherlands|Belgium|Canada|India
```

As the value in this line contains pipes (|) ConfigEditor will auto-detect it as SELECTFIRST type, showing a drop-down selection box, allowing the user to select one of the countries. The first string in the value ("USA") is used as the default selection. If the user selects another country, his selection will be saved at the first place on the line, before the "USA" value. Of course you may use any string you want instead of the key name "Country" or the strings between pipes. Note that pipe chars are usually typed by pressing "Shift \" on your keyboard.

```
Age=43
```

as the value is a number, ConfigEditor will auto-detect it as INTEGER, allowing the user to type only numbers into the edit input box.

NOTE that any spaces or tabs are treated as part of the key/value string by default, so

```
Key=Value
```

has different key and value than:

```
Key = Value
```

and therefore:
 * any spaces in the key name within the INI file must exist exactly the same in the ITP file (if used)
 * any spaces in the value will cause auto-detection to fail recognizing numbers or colors, and detect them as string instead.

to prevent this you may use the:

ConfigEditor.exe /TRIMWHITESPACE
command line parameter (see Command Line chapter).

To prevent ConfigEditor from auto-detecting the value types, you may use the:
ConfigEditor.exe /NODETECT
command line parameter (see Command Line chapter).
For using more value types, you may create a Template file which contains value type definitions. (see Template Files chapter)

Comments in an INI file are lines starting with a semicolon ";" followed by a single line of text. note that pound sign "#" can also be used but deprecated.

```
;this is a comment line
```

Keys starting with an at sign "@" will be hidden (ignored):

```
@Key=Value
```

Customizing ConfigEditor Behavior

It is possible to customize ConfigEditor behavior, such as describing value types, disabling auto-detection, specifying the names of the settings files and more. This can be done using a Template File, Command Line parameters, or the (Pro) Generator.

Editing Template Files (.ITP)

Template files contains value type definitions for a specific settings file. They are text files, which can be created and edited with any text editor, like Notepad. They have the extension ITP and have similar structure to INI files.
Usually, a template file has the same name as the settings file it describes, but ends with .ITP extension.

Each row in the template file is in this format:
KEY=TYPE|FRIENDLY NAME|TIP|OPTIONAL1|OPTIONAL2

KEY corresponds to the key with the same name on the INI file, to tell ConfigEditor which key we are defining.
TYPE is the value type, such as STRING, COLOR or INTEGER, which determines who ConfigEditor will display it. (for complete list of type values see the Value Types chapter)
FRIENDLY NAME is the text to display in the leftmost column of that row. If left empty, ConfigEditor will use the key name itself.
TIP is a text that will be displayed when the user single-clicks a row on the list (tips are shown at the tip area at the top)
OPTIONAL1 and others are specific for that value TYPE. Some value types don't have any optional fields, while others may have many.

Any of the fields can be left empty to use the defaults.

In this example, the settings file SETTINGS.INI will use a template file called SETTINGS.ITP to define its value types. We will use the same INI file as in the Getting Started chapter, and we will add a few new rows, and define value types for each value.

This is the SETTINGS.INI file:

```
[Settings]
FreeString=hello my name is mike!
BkColor=#CFED8B
Country=USA|England|Netherlands|Belgium|Canada|India
Age=43
Mail_Message=hello\ni'm going to work\nbye\nmike
```

and this is the SETTINGS.ITP file:

```
[Settings]
BkColor=COLOR|Background Color|choose a color
Country=SELECTFIRST||where are you?|N
```

```
Age=INTEGER|||18|65|0
Mail_Message=MULTILINE
```

We can see that both files start with the same section [Settings] as ConfigEditor makes sure that keys exist under the same section, otherwise it considers them to be different keys.

The first key (FreeString) in the INI file doesn't appear in template file, this will cause ConfigEditor to show it as the default type which is STRING. The leftmost column on the display row will contain the key name "FreeString" since a friendly name wasn't provided.

The second key (BkColor) appears both in the INI and the template file, where it has COLOR defined as its value type, causing ConfigEditor to display it using the Color Picker input. A friendly name is provided "Background Color" which will be displayed in the leftmost column instead of the key name. The Tip "choose a color" will be shown when the row is single-clicked or selected.

The third key (Country) has SELECTFIRST defined as its value type, causing ConfigEditor to display it using a drop-down list. A friendly name wasn't given (left empty) so the original key name will be displayed on the leftmost column, and a tip was given. The last field (N) is specific for this value type and means that the displayed drop-list will Not be sorted.

The fourth key (Age) has INTEGER defined as its value type. A friendly name or tip weren't given (left empty). The next optional field indicates a minimum (18) and the one after that indicates a maximum (65) - this will cause ConfigEditor to display a sliding scroll bar when the field is edited as well as enforce these limits. The last field (0) indicates the number of padding zeros to add on the left of the number, in this case 0 means no padding.

The fifth key (Mail_Message) has MULTILINE defined as its value type, causing ConfigEditor to display it using a text box that may contain several lines of text. No other fields where provided in this case (although friendly name and tip fields could have been provided).
The value in the INI file is the text currently stored in this key, note that each \n represents a new line, so the actual text is:
hello
I'm going to work
bye
mike

NOTE: It is recommended to look at the files SETTINGS.INI and SETTINGS.ITP bundled with this package.

Value Types

are used in template files, within the definition of keys. The value type describing an INI entry, such as STRING, COLOR or INTEGER, determines who ConfigEditor will display, edit, and save it. (see the Editing Template Files chapter)

Each row in the template file is in this format:
KEY=TYPE|FRIENDLY NAME|TIP|OPTIONAL1|OPTIONAL2

The part in green is common to all value types:
KEY corresponds to the key with the same name on the INI file, to tell ConfigEditor which key we are defining.
TYPE is the Value Type, such as STRING, COLOR or INTEGER, which determines who ConfigEditor will display it.
FRIENDLY NAME is the text to display in the leftmost column of that row. If left empty, ConfigEditor will use the key name itself.
TIP is a text that will be displayed when the user single-clicks a row on the list (tips are shown at the tip area at the top)
OPTIONAL1 and others are specific for that value TYPE. Some value types don't have any optional fields, while others may have many.

Any of the fields can be left empty to use the defaults.

STRING

This type represents a line of text.
Shown as another row on the list with the value shown on the second column.
Edited by a standard edit box where the user can type any text.
Saved as a single line of text (exactly as displayed).
This is the default type if no type is given.
Format:
KEY=**STRING**|FRIENDLY NAME|TIP

INTEGER

This type represents number between -2,147,483,648 to +2,147,483,647 (no fractions).
Shown as another row on the list with the value shown on the second column.
Edited by a standard edit box where the user can type any whole number, and a sliding scrollbar if a minimum and maximum values are provided.

Saved as a single number (exactly as displayed).
Format:
KEY=**INTEGER**|FRIENDLY NAME|TIP|**MINIMUM**|**MAXIMUM**|**PADDING**
MINIMUM is the smallest number the user is allowed to type in this field. smaller numbers will be adjusted upwards.
MAXIMUM is the largest number the user is allowed to type in this field. larger numbers will be adjusted downwards.
PADDING is the minimal number of digits for this field. if the user enters less digits, the number will be padded from the left with 0s to reach the minimal amount of digits. empty or 0 means no padding.

REAL

This type represents floating point number including decimal fractions (for example 150.4321).
Shown as another row on the list with the value shown on the second column.
Edited by a standard edit box where the user can type any numbers, and a sliding scrollbar if a minimum and maximum values are provided.
Saved as a single number (exactly as displayed).
Format:
KEY=**REAL**|FRIENDLY NAME|TIP|**MINIMUM**|**MAXIMUM**|**PADDING**|**PRECISION**
MINIMUM is the smallest number the user is allowed to type in this field. smaller numbers will be adjusted upwards.
MAXIMUM is the largest number the user is allowed to type in this field. larger numbers will be adjusted downwards.
PADDING is the minimal number of digits for this field. if the user enters less digits, the number will be padded from the left with 0s to reach the minimal amount of digits. empty or 0 means no padding.
PRECISION is the minimal number of digits to the right of the decimal floating point. empty o 0 means default precision of 4.

READONLY

This type represents single line of text that cannot be edited.
Shown as another row on the list with the value shown on the second column.
Editing is disabled for this type.
Save is disabled for this type.
Format:
KEY=**READONLY**|FRIENDLY NAME|TIP

MULTILINE

This type represents several lines of text.
Shown as another row on the list with the value shown on the second column. shown as a single line of text.
Edited by a standard edit box where the user can type any text, and create new lines using the enter key.
Saved as a single line of text, where \n represent new lines.
Format:
KEY=**MULTILINE**|FRIENDLY NAME|TIP

TITLE

This type represents title row, comprised of a single line of text that cannot be edited.
Shown as another row on the list in a different background color, with the value shown on the second column, and an empty first column.
Editing is disabled for this type.
Save is disabled for this type.
Note that since this value is not saved, the key may appear more than once per section within the settings file (not in the template file).
Format:
KEY=**TITLE**|FRIENDLY NAME|TIP

COLOR

This type represents an RGB color value (example: #00FF00 for green).
Shown as another row on the list with the value shown on the second column, colored using that color value.
Edited by a Color Picker input where the user visually selects a color, or types its value.
Saved as a RGB hex number preceded by a # sign, as used in HTML and CSS. (exactly as displayed).
Format:
KEY=**COLOR**|FRIENDLY NAME|TIP

FONT

This type represents an font with a specific size and style.
Shown as another row on the list with the font name shown on the second column.
Edited by a Font Picker input where the user visually selects a Font.

Saved as pipe (|) separated values for font name,size, and properties.
Format:
KEY=**FONT**|FRIENDLY NAME|TIP|**MINIMUM SIZE**|**MAXIMUM SIZE**
MINIMUM SIZE is the smallest font size allowed.
MAXIMUM SIZE is the largest font size allowed.

FILE

This type represents an existing file name with full path.
Shown as another row on the list with the full file name shown on the second column.
Edited by a File Selection box where the user visually selects a file. the selector checks that the file exists.
Saved as a single line of text containing the full file name with path (exactly as displayed).
Format:
KEY=**FILE**|FRIENDLY NAME|TIP|**ALLOWED EXTENSION 1**|**ALLOWED EXTENSION 2|...**
ALLOWED EXTENSION 1 and onwards specify the file extensions visible in the file selector, in the format of:
*.extension
Example:
MyFile=FILE|||*.txt|*.rtf|*.pdf

FILESAVE

This type represents a theoretical file name with full path, for the purpose of saving.
Shown as another row on the list with the full file name shown on the second column.
Edited by a File Selection box where the user visually selects a file, or types it in.
Saved as a single line of text containing the full file name with path (exactly as displayed).
Format:
KEY=**FILESAVE**|FRIENDLY NAME|TIP|**ALLOWED EXTENSION 1**|**ALLOWED EXTENSION 2|...**
ALLOWED EXTENSION 1 and onwards specify the file extensions visible in the file selector, in the format of:
*.extension
Example:
MyFile=FILESAVE|||*.txt|*.rtf|*.pdf

FOLDER

This type represents an existing folder (full path).
Shown as another row on the list with the full folder path shown on the second column.
Edited by a Folder Selection box where the user visually selects a folder. the selector checks that the folder exists.
Saved as a single line of text containing the full folder path (exactly as displayed).
Format:
KEY=**FOLDER**|FRIENDLY NAME|TIP

SELECTFIRST

This type represents a list of strings, where only 1 can be selected.
Shown as another row on the list with the selected string shown on the second column.
Edited by a drop-down list box where the user can select one of the strings.
Saved as pipe (|) separated values for each string on the list, where the selected string is the first value.
Format:
KEY=**SELECTFIRST**|FRIENDLY NAME|TIP|**SORT**
SORT indicates whether the displayed drop-list should be sorted. Y for yes, N for no.

SELECTSTRING

This type represents a list of strings, where only 1 can be selected.
Shown as another row on the list with the selected string shown on the second column.
Edited by a drop-down list box where the user can select one of the strings.
Saved as a text line containing the selected string (exactly as displayed).
Format:
KEY=**SELECTSTRING**|FRIENDLY NAME|TIP|**SORT**|**STRING 0**|**STRING 1|...**
SORT indicates whether the displayed drop-list should be sorted. Y for yes, N for no.
STRING 0 and onwards specify the strings to choose from

SELECTINDEX

This type represents a list of strings, where only 1 can be selected.
Shown as another row on the list with the selected string shown on the second column.
Edited by a drop-down list box where the user can select one of the strings.
Saved as a text line containing the index of the selected string (where the first string index is 0).
Format:
KEY=**SELECTINDEX**|FRIENDLY NAME|TIP|**SORT**|**STRING 0**|**STRING 1**|**...**
SORT indicates whether the displayed drop-list should be sorted. Y for yes, N for no.
STRING 0 and onwards specify the strings to choose from

SELECTINDEXSTRING

This type represents a list of strings, where only 1 can be selected.
Shown as another row on the list with the selected string shown on the second column.
Edited by a drop-down list box where the user can select one of the strings.
Saved as a text line containing the index of the selected string (where the first string index is 0) followed by a pipe (|) then the selected string as text.
Format:
KEY=**SELECTINDEXSTRING**|FRIENDLY NAME|TIP|**SORT**|**STRING 0**|**STRING 1**|**...**
SORT indicates whether the displayed drop-list should be sorted. Y for yes, N for no.
STRING 0 and onwards specify the strings to choose from

TOGGLEINDEX

This type represents a boolean that can get one of two strings.
Shown as another row on the list with the selection shown on the second column.
Edited by toggling between the two strings when the user double-clicks the row.
Saved as 0 or 1 which is the index of the selected string.
Format:
KEY=**TOGGLEINDEX**|FRIENDLY NAME|TIP|**STRING0**|**STRING1**
STRING0 is the first string (if empty will default to "No")
STRING1 is the second string (if empty will default to "Yes")

PASSWORD

This type represents a password.
Shown as another row on the list with asterisks shown on the second column.
Edited by a password edit box where the user can type any text, but not see it.
Saved as a single line of text (clear text, no encryption).
Format:
KEY=**PASSWORD**|FRIENDLY NAME|TIP|**MINIMUM LENGTH**|**MAXIMUM LENGTH**
MINIMUM LENGTH is the shortest password allowed.
MAXIMUM LENGTH is the longest password allowed.

EMAIL

(Pro) This type represents a line of text that contains an e-mail address.
Shown as another row on the list with the value shown on the second column.
Edited by a standard edit box where the user can type any text.
Saved as a single line of text (exactly as displayed).
It shows a special icon indicating an email.
Format:
KEY=**EMAIL**|FRIENDLY NAME|TIP

MD5PASSWORD

(Pro) This type represents a password, encrypted using the MD5 hash algorithm.
Shown as another row on the list with asterisks shown on the second column.
Edited by a password edit box where the user can type any text, but not see it.
Saved as a hash code.
Format:
KEY=**MD5PASSWORD**|FRIENDLY NAME|TIP|**MINIMUM LENGTH**|**MAXIMUM LENGTH**
MINIMUM LENGTH is the shortest password allowed.
MAXIMUM LENGTH is the longest password allowed.

SHA160PASSWORD

(Pro) This type represents a password, encrypted using the SHA160 (also known as SHA1) hash algorithm.
available options are identical to MD5PASSWORD.

SHA256PASSWORD

(Pro) This type represents a password, encrypted using the SHA256 hash algorithm.
available options are identical to MD5PASSWORD.

SHA384PASSWORD

(Pro) This type represents a password, encrypted using the SHA384 hash algorithm.
available options are identical to MD5PASSWORD.

SHA512PASSWORD

(Pro) This type represents a password, encrypted using the SHA512 hash algorithm.
available options are identical to MD5PASSWORD.

HEXPASSWORD

(Pro) This type represents a password, encoded into HEX (hexadecimal) string.
Shown as another row on the list with asterisks shown on the second column.
Edited by a password edit box where the user can type any text, but not see it.
Saved as a HEX (hexadecimal) byte string representation of a unicode16 NULL terminated string (for example, the password "ABC" will be saved as 4100420043000000.
Format:
KEY=**HEXPASSWORD**|FRIENDLY NAME|TIP|**MINIMUM LENGTH**|**MAXIMUM LENGTH**
MINIMUM LENGTH is the shortest password allowed.
MAXIMUM LENGTH is the longest password allowed.

DATE

(Pro) This type represents a date (year month and day).
Shown as another row on the list with selected date shown on the second column.
Edited by a locally formatted date box and a popup calendar where the user can select a date or type it.
Saved as a single line of text in a standard 10 char natural date format (example: 2012-01-30).
Format:
KEY=**DATE**|FRIENDLY NAME|TIP|**MINIMUM DATE**|**MAXIMUM DATE**
MINIMUM DATE is the oldest date allowed.
MAXIMUM DATE is the newest date allowed.

TIME

(Pro) This type represents a time within the day (hour minute and second).
Shown as another row on the list with selected time shown on the second column.
Edited by a locally formatted time box where the user can adjust a time or type it.
Saved as a single line of text in a standard 8 char time format (example: 20:59:00).
Format:
KEY=**TIME**|FRIENDLY NAME|TIP|**MINIMUM TIME**|**MAXIMUM TIME**
MINIMUM TIME is the oldest time allowed.
MAXIMUM TIME is the newest time allowed.

STRINGHEX

(Pro) This type represents a string, encoded into HEX (hexadecimal) string.
Shown as another row on the list with decoded string shown on the second column.
Edited by a standard edit box where the user can type any text.
Saved as a HEX (hexadecimal) byte string representation of a unicode16 NULL terminated string (for example, the string "ABC" will be saved as 4100420043000000. an empty string is 0000.
Note that this field type can save chars that are usually too strange to save, such as control chars.
Format:

KEY=**STRINGHEX**|FRIENDLY NAME|TIP

IMAGEFILE

(Pro) This type represents an existing picture file name with full path.
available options are identical to FILE, it only differs by the display icon (which is more suitable for picture files).

DRIVE

(Pro) This type represents a list of available drives, where only 1 can be selected.
Shown as another row on the list with the selected drive shown on the second column.
Edited by a drop-down list box where the user can select one of the drives.
Saved as a text line containing the selected drive name, a space, and either the type of drive surrounded by round braces, or the volume label surrounded by square brackets (exactly as displayed).
Format:
KEY=**DRIVE**|FRIENDLY NAME|TIP

REMOVABLEDRIVE

(Pro) This type represents a list of available removable drives (such as USB drives), where only 1 can be selected.
Shown as another row on the list with the selected drive shown on the second column.
Edited by a drop-down list box where the user can select one of the drives.
Saved as a text line containing the selected drive name, a space, and the type of drive surrounded by round braces (exactly as displayed).
Format:
KEY=**REMOVABLEDRIVE**|FRIENDLY NAME|TIP

CDROMDRIVE

(Pro) This type represents a list of available CD-ROM or DVD drives, where only 1 can be selected.
available options are identical to REMOVABLEDRIVE.

REMOTEDRIVE

(Pro) This type represents a list of available Network mapped drives, where only 1 can be selected.
available options are identical to REMOVABLEDRIVE.

RAMDRIVE

(Pro) This type represents a list of available RAM drives, where only 1 can be selected.
available options are identical to REMOVABLEDRIVE.

FIXEDDRIVE

(Pro) This type represents a list of available Fixed non-removable drives (such as Hard Disks), where only 1 can be selected.
available options are identical to REMOVABLEDRIVE.

WINDOW

(Pro) This type represents a list of the running application windows, where only 1 can be selected.
Shown as another row on the list with the selected window title shown on the second column.
Edited by a drop-down list box where the user can select one of the windows.
Saved as pipe (|) separated values containing the selected window HWND handle| title| and class name.
Format:
KEY=**WINDOW**|FRIENDLY NAME|TIP

SELECTEXEC

(Pro) This type represents a list of strings obtained from an external console application that prints out list of strings separated by pipes (|) or new lines. only 1 can be selected. (see the **Plugins** subfolder under the installation folder, or the ConfigEditor SDK for further detail).
Shown as another row on the list with the selected string shown on the second column.
Edited by a drop-down list box where the user can select one of the strings.
Saved as pipe (|) separated values for each string on the list, where the selected string is the first value.
Note that this is NOT very suitable for listing dynamic things, like available USB COM ports or Process IDs, since the external application is only

ran once when ConfigEditor starts.
It is recommended that the application will only run for a few milliseconds, as ConfigEditor will wait until the application ends.
Format:
KEY=**SELECTEXEC**|FRIENDLY NAME|TIP|**SORT**|**EXECUTABLE FILE**|**PARAMETERS**
SORT indicates whether the displayed drop-list should be sorted. Y for yes, N for no.
EXECUTABLE FILE is the file name of the plugin application to execute
PARAMETERS is the command line parameters passed to the plugin application (if any)

EXEC

(Pro) This is a special type, which is defined by the output obtained from an external console application that prints out a definition line with the same format as in a template file. (see the **Plugins** subfolder under the installation folder, or the ConfigEditor SDK for further detail).
It is Shown, Edited, and Saved exactly as the type defined by the application output.
Note that this is NOT very suitable for listing dynamic things, like available USB COM ports or Process IDs, since the external application is only ran once when ConfigEditor starts.
It is recommended that the application will only run for a few milliseconds, as ConfigEditor will wait until the application ends.
The format of this line is different that the normal template definitions.
Format:
KEY=**EXEC**|**EXECUTABLE FILE**|**PARAMETERS**
EXECUTABLE FILE is the file name of the plugin application to execute
PARAMETERS is the command line parameters passed to the plugin application (if any)

Command Line Parameters

When launched, ConfigEditor can optionally accept several parameters for changing the default behavior, and integrating it with your software.
As an example, running the following command on the command prompt:
C:\MyApp\ConfigEditor.exe /NODETECT /INI:"C:\MyApp\my.ini"
will cause ConfigEditor to edit the file "my.ini" without auto-detecting value types, so all values will be edited as strings.
(this example assumes the files are in the folder "C:\MyApp" , you may need to change that if this is not the case).

Here is a list of available parameters:

/INI:"settingsfile.ini"

This parameter will cause ConfigEditor to edit the specified file instead of the default file "SETTINGS.INI" (This will also cause ConfigEditor to look for a template file that has the same name as the settings file, but ends with .ITP extension, unless the /ITP parameter is used). You can specify a java Properties file instead of an INI file.
Example 1:
ConfigEditor.exe /INI:"C:\MyApp\my.ini"
Example 2:
ConfigEditor.exe /INI:"C:\MyApp\my.properties"

/ITP:"templatefile.itp"

This parameter will use the specified template file instead of the default template file. By default, ConfigEditor looks for a template file that has the same name as the settings file, but ends with .ITP extension - this parameter changes this behavior.

/NOTIFY:handle

This parameter will cause ConfigEditor to send window message notifications the the window handle specified by "handle". (see the ConfigEditor SDK for further detail)
Example 1: will send window message notifications to HWND 1234:
ConfigEditor.exe /NOTIFY:1234

/NODETECT

This will prevent auto-detecting value types, so all values will be edited as STRING.

/SEPARATOR:char

Set the separator character used for separating items (such as in SELECT, SELECTINDEX and templates). If this isn't used the default is a pipe | char. it CANNOT be one of the following: space, doublequotes, forward slash.
Example 1: will set separator to semicolon

ConfigEditor.exe /SEPARATOR:;

/COMMENTS:number

This determines how comments in the INI file will be treated. The number specifies a mode of behavior.
Comments in an INI file are lines starting with a semicolon (;)
0 means Ignore comments
1 means show comments as read-only rows on the list
2 means show comments as tips, each comment is a tip for the next row.
Example 1:
ConfigEditor.exe /COMMENTS:1

/SHOWCOMMENTS

This is the same as /COMMENTS:1

/TIPCOMMENTS

This is the same as /COMMENTS:2

/COMMENTSLEFT

Show comments on the left column.

/INDENTS:number

Sets the amount of spaces palced before each row, so the key name will be indented.
Example 1 (indent each row 5 places to the right):
ConfigEditor.exe /INDENTS:5

/NOINDENTS

This is the same as /INDENTS:0

/NOICONS

Don't show item type icons.

/NOTITLES

Don't show section names as titles.

/NOTITLESLEFT

Show titles on the right column.

/NOCOMMENTSLEFT

Show comments on the right column.

/NOGRID

Hide the grid lines between rows on the list.

/NOAUTOSIZE

Don't auto-fit the ConfigEditor window size to the length of the list.

/NOTRIMWHITESPACE

Whitespace before and after keys and values in the INI file will be considered as part of the name. So that "key = value" is a different row than "key=value". (this is the default).

/DETECT

This causes ConfigEditor to auto-detect value types when no template is specified. (this is the default)

/ICONS

Show item type icons. (this is the default)

/TITLES

Show section names as titles. (this is the default)

/TITLESLEFT

Show titles on the left column.

/GRID

Shows grid lines between rows on the list.

/AUTOSIZE

Auto-fits the ConfigEditor window size to the length of the list. (this is the default)

/TRIMWHITESPACE

Ignore whitespace before and after keys and values in the INI file. So that "key = value" will be treated as "key=value".

/HIDEUNTEMPLATED

If a template is used, Hide (ignore) rows for keys that don't appear in the template file.

/BACKGROUNDCOLOR:RRGGBB

(Pro) Sets the background color of the main window margins. This is an RGB value in hex digits, as used in HTML and CSS.
Example 1: set the color to blue:
ConfigEditor.exe /BACKGROUNDCOLOR:0000FF

/TEXTCOLOR:RRGGBB

(Pro) Sets the color of text on the main window outside the main list (such as tips). This is an RGB value in hex digits, as used in HTML and CSS.
Example 1: set the color to blue:
ConfigEditor.exe /TEXTCOLOR:0000FF

/LISTTEXTCOLOR:RRGGBB

(Pro) Sets the color of text on the main list. This is an RGB value in hex digits, as used in HTML and CSS.
Example 1: set the color to blue:
ConfigEditor.exe /LISTTEXTCOLOR:0000FF

/LISTBKCOLOR:RRGGBB

(Pro) Sets the background color of the rows on the main list. This is an RGB value in hex digits, as used in HTML and CSS.
Example 1: set the color to blue:
ConfigEditor.exe /LISTBKCOLOR:0000FF

/LISTODDBKCOLOR:RRGGBB

(Pro) Sets the background color of odd numbered rows on the main list. This is an RGB value in hex digits, as used in HTML and CSS.
Example 1: set the color to blue:
ConfigEditor.exe /LISTODDBKCOLOR:0000FF

/SECTIONBKCOLOR:RRGGBB

(Pro) Sets the background color of section title rows on the main list. This is an RGB value in hex digits, as used in HTML and CSS.
Example 1: set the color to blue:
ConfigEditor.exe /SECTIONBKCOLOR:0000FF

/COMMENTBKCOLOR:RRGGBB

(Pro) Sets the background color of comments when shows as rows. This is an RGB value in hex digits, as used in HTML and CSS.
Example 1: set the color to blue:
ConfigEditor.exe /COMMENTBKCOLOR:0000FF

/X:number

(Pro) Sets the X (horizontal) position of the main window.
Example 1: position ConfigEditor 10 pixels from the left of the screen:
ConfigEditor.exe /X:10

/Y:number

(Pro) Sets the Y (vertical) position of the main window.
Example 1: position ConfigEditor 10 pixels from the top of the screen:
ConfigEditor.exe /Y:10

/W:number

(Pro) Sets the Width (horizontal size) of the main window.
Example 1: Set ConfigEditor window width to 300:
ConfigEditor.exe /W:300

/H:number

(Pro) Sets the Height (vertical size) of the main window.
Example 1: Set ConfigEditor window height to 300:
ConfigEditor.exe /H:300

/MAXIMIZE

(Pro) Maximizes the size of the main window.

/FIRSTCOLUMNWIDTH:number

(Pro) Sets the Width (horizontal size) of the first column (leftmost column) on the list.

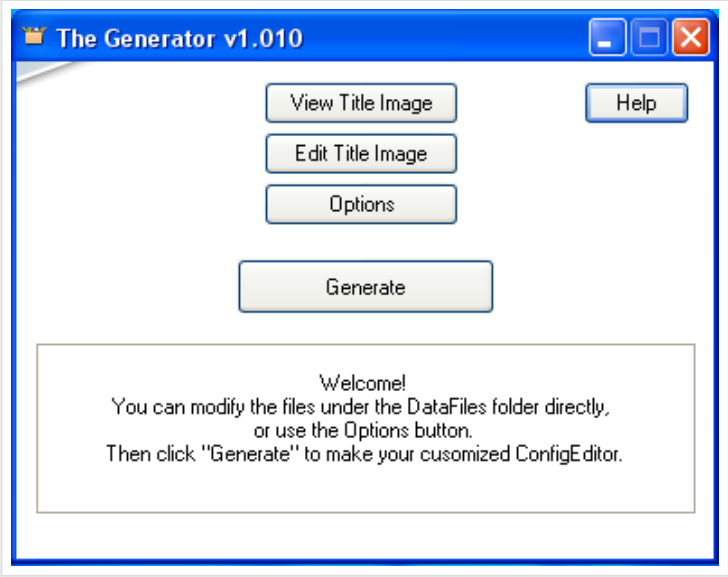
The Generator

The (Pro) version of ConfigEditor comes with a tool called The Generator, that can create customized versions of ConfigEditor. It does that by replacing the title image of ConfigEditor, and changing some of the defaults, such as colors and behavior.
It uses 2 files that are located in the DataFiles subfolder of the installation folder:

banner.bmp - The title image. A standard image file that will be placed at the top of the main ConfigEditor window. You can edit or replace this file to give ConfigEditor a new look and branding that will elegantly fit your application UI design.

options.ini - Stores default options for ConfigEditor behavior, color and size (similar to those on the command line parameters). You can edit this file directly using a text editor, or by clicking the **Options** button. See notes inside this file for more information.

When you click the **Generate** button, The settings from the 2 files above will be used to generate a new ConfigEditor.exe file, that has these defaults built-in.



Troubleshooting

Problem: types are incorrectly detected after upgrading from 1.xxx version
Reason: seprator in version 2.0 was changed from semicolon ; to pipe |
Solution: Use the /SEPARATOR:; command line, or edit your INI and template file to replace all separator semicolons with pipes.

Problem: types are incorrectly detected, or wrongly set by ITP file
Reason: Key name or value contains unintended space or tab chars
Solution1: Make sure any spaces in the key name within the INI file must exist exactly the same in the ITP file (if used), and any redundent spaces are removed from the value field.
Solution2: to prevent this you may use the "ConfigEditor.exe /TRIMWHITESPACE" command line parameter (see Command Line chapter).

Problem: Config Editor give rerror code or windows asks for User Access Control
Reason: Some Windows settings or anti-virus may be restricting access rights to the system files
Solution1: Try to intall or run it from a folder on the root drive (ie: C:\MySoftware)
Solution2: You can sign the ConfigEditor executable with your digital signature, see:
http://msdn.microsoft.com/en-us/library/windows/desktop/aa387764(v=vs.85).aspx
Solution3: Run ConfigEditor as administrator

FAQ

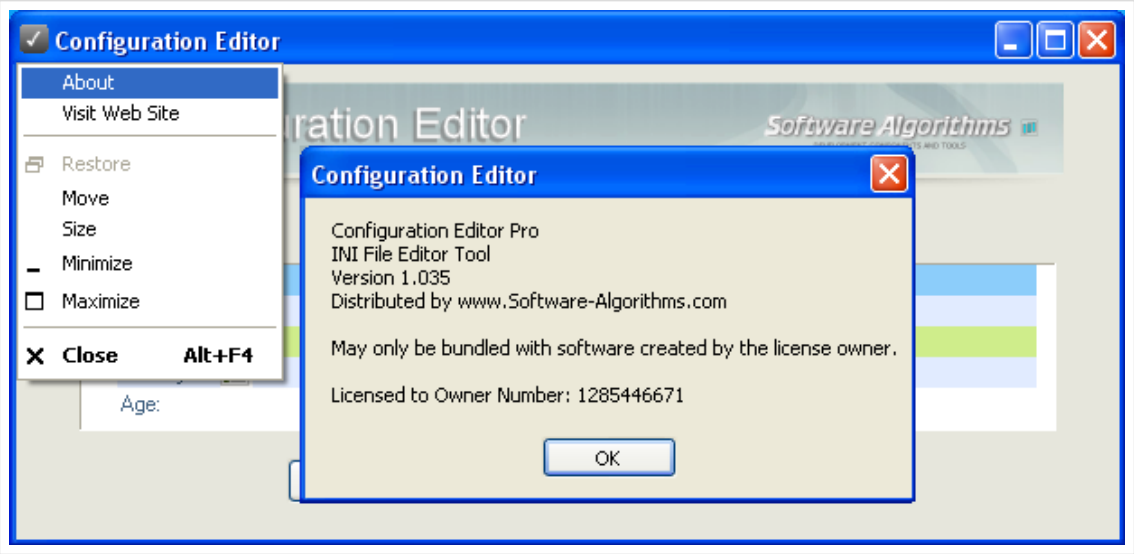
Can I tell ConfigEditor to use another settings file?
This can be done using command line parameter, for example: ConfigEditor.exe /INI:"C:\MySettings.ini"
See the Command Line chapter for further detail.
How do i create an INI file?
INI files are text files, they can be created and edited with any text editor, like Notepad.
How do i make my software use ConfigEditor to edit settings?
Simply execute ConfigEditor.exe from your application (for example when the user clicks your "Settings" button). Preferably you'll want to provide your softwares window handle to ConfigEditor to be notified when it closes:
ConfigEditor.exe /NOTIFY:123456
see the ConfigEditor SDK for further detail.
Can my software control the ConfigEditor window?
Yes, you can use the ConfigEditor SDK for that.
Can you add more value types to the Pro version?
Yes, contact us with your request, and we will usually to add it to the Pro version, with no extra charge for you.
Is it safe and free from Ads and Spyware?
ConfigEditor is %100 free from spyware and adware.
Can you make it open source?
ConfigEditor source code can be purchased, as we feel the "purchasable source" approach provides the best of all GPL/MIT and commercial licensing: You can get the source code and feel secure you have %100 control of the product you use, and on the other hand, the maker gets compensation for the hard work, time, and money invested in the product.
How can i protect the ConfigEditor i generated from being distributed by others?

The ConfigEditor.exe (Pro) you generated using your Generator will be marked with your Licensed Owner Number which is a number unique to your license (See License section below). If another company is distributing your ConfigEditor they are violating the agreement and the law, so both you and the manufacturer have the right to sue them.

License

The Free version, is free for both commercial and non-commercial use or distribution by anyone, totaly free of charge. See FREE License Agreement for further details (CEFLicense.txt).

For the Pro version, the generated application file called ConfigEditor.exe can be distributed only when bundled with software created by the license owner. The Generator can be used only on the owners computer. For the full license terms please see the Software License Agreement (License.txt).
The owner license number is imprinted into ConfigEditor.exe, and can be viewed by clicking the About option on the system menu:



If another company is distributing your ConfigEditor they are violating the agreement and the law, so both you and the manufacturer have the right to sue them.

Support

You are welcome to contact our support team with any questions or suggestions. Visit ConfigEditor support page.

Download

Download ConfigEditor FREE kit, which includes ConfigEditor FREE, Examples, Documentation, and the SDK.

ConfigEditor FREE version 2.018 (full kit)

Buy Pro

You can buy Configuration Editor Pro from the link below. Your purchase will also show your support for Configuration Editor!
After purchase you will receive a download link, and your license key. Your license owner name will be the company or personal name you enter during purchase. Before buying, please use the freeware ConfigEditor FREE to make sure you are satisfied with the product.

Buy Now