Corel SCRIPT Editor commands and functions

.FileClose (Corel SCRIPT Editor)

Command: .FileClose

Function: ReturnValue = .FileClose ( )

This command and function closes the active script.

Syntax	Description
ReturnValue	Specifies the numeric variable that is assigned a value corresponding to whether the active script was closed: TRUE (-1) if closed; otherwise FALSE (0).

Note

- This command must be preceded by a .**FileNew** or .**FileOpen** command or else an error occurs. An error will occur because the .**FileClose** command tries to close the script which is executing the command.
- If this command is not preceded by the .**FileSave** or .**FileSaveAs** command, the script changes will be lost.
- This command corresponds to the Close command in the File menu in the Corel SCRIPT Editor. Click File, Close.

Example

```
.FileOpen "C:\MyScripts\graphics.csc"
.Execute
.FileClose
```

The above example opens the script GRAPHICS.CSC, executes it, and then closes it.

{button ,AL(`cse_file_cse;;;;;',0,"Defaultoverview",)} Related Topics

.FileNew (Corel SCRIPT Editor)

.FileNew

This command creates a new Corel SCRIPT script. The new script becomes the active script and the insertion point is placed at the beginning of the first line.

Note

- This command corresponds to the New command in the File menu in the Corel SCRIPT Editor. Click File, New.

Example

```
.FileNew
```

The above example creates a new script.

{button ,AL(`cse_file_cse;;;;;',0,"Defaultoverview",)} **Related Topics**

.FileOpen (Corel SCRIPT Editor)

Command: .FileOpen FileName

Function: ReturnValue = .FileOpen (FileName)

This command and function loads a previously saved Corel SCRIPT script. The opened script becomes the active script and the insertion point is placed at the beginning of the first line.

Syntax	Description
FileName	String expression that specifies the name and path of the script to open.
ReturnValue	Specifies the numeric variable that is assigned a value corresponding to whether the specified script was opened: TRUE (-1) if opened; otherwise FALSE (0).

Note

- This command corresponds to the Open command in the File menu in the Corel SCRIPT Editor. Click File, Open.

Example

```
.FileOpen "C:\MyScripts\graphics.csc"
.Execute
.FileClose
```

The above example opens the script GRAPHICS.CSC, executes it, and then closes it. The first line can also be specified as follows:

```
ReturnValue = .FileOpen ("C:\MyScripts\graphics.csc")
```

{button ,AL(`cse_file_cse;;;;;',0,"Defaultoverview",)} Related Topics

.FileSave (Corel SCRIPT Editor)

Command: .FileSave

Function: ReturnValue = .FileSave ( )

This command and function saves the active script using its current name and location.

Syntax	Description
ReturnValue	Specifies the numeric variable that is assigned a value corresponding to whether the active script was saved: TRUE (-1) if saved; otherwise FALSE (0).

Note

- This command corresponds to the Save command in the File menu in the Corel SCRIPT Editor. Click File, Save.
- If the script is not named, the Save As dialog box opens.
- If the .**FileSave** command is not preceded by a .**FileNew** or .**FileOpen** command, Corel SCRIPT tries to save the script that is executing the command.

Example

```
.FileSave
```

The above example saves the active script. This line can also be specified as follows:

```
ReturnValue = .FileSave ( )
```

{button ,AL(`cse_file_cse;;;;;',0,"Defaultoverview",)} Related Topics

.FileSaveAs (Corel SCRIPT Editor)

Command: .FileSaveAs FileName

Function: ReturnValue = . .FileSaveAs (FileName)

This command and function saves the active script under a new name or in a different location.

Syntax	Description
FileName	String expression that specifies the name and path to save the script under.
ReturnValue	Specifies the numeric variable that is assigned a value corresponding to whether the active script was saved: TRUE (-1) if saved; otherwise FALSE (0).

Note

- This command corresponds to the Save As command in the File menu in the Corel SCRIPT Editor. Click File, Save As.

Example

```
.FileOpen "C:\MyScripts\graphics.csc"
.FileSaveAs "D:\graphics.csc"
.FileClose
```

The above example opens the script GRAPHICS.CSC, saves it on the D drive with the same name, and then closes it. The second line can also be specified as follows:

```
ReturnValue = .FileSaveAs ("D:\graphics.csc")
```

{button ,AL(`cse_file_cse;;;;;',0,"Defaultoverview",)} Related Topics

.CheckSyntax (Corel SCRIPT Editor)

.CheckSyntax

This command checks for syntax errors in the active script. Common syntax errors include misspelling commands, missing operators, and missing punctuation. If errors are found, error messages appear in the Compiler Output window.

Note

- This command corresponds to the Check Syntax command in the Debug menu in the Corel SCRIPT Editor. Click Debug, Check Syntax.

Example

```
.CheckSyntax
```

{button ,AL(`cse_debug_cse;;;;;',0,"Defaultoverview",)} **Related Topics**

.Execute (Corel SCRIPT Editor)

.Execute

This command runs the active script. This command ignores all debugging information, including script breakpoints during script execution. To debug a script, use the .**Run** command. Running a script in debug mode is noticeably slower than running a script using the .**Execute** command.

Note

- This command corresponds to the Execute command in the Debug menu in the Corel SCRIPT Editor. Click Debug, Execute.

Example

```
.FileOpen "C:\MyScripts\graphics.csc"
.Execute
.FileClose
```

The above example opens the script GRAPHICS.CSC, executes it, and then closes it.

{button ,AL(`cse_debug_cse;;;;;',0,"Defaultoverview",)} **Related Topics**

.Run (Corel SCRIPT Editor)

.Run

This command runs the active script in debug mode. Script execution stops at breakpoints, or when the end of the script is reached. Running a script in debug mode is noticeably slower than running a script using the .**Execute** command.

Note

- This command corresponds to the Run command in the Debug menu in the Corel SCRIPT Editor. Click Debug, Run.

Example

```
.FileOpen "C:\MyScripts\graphics.csc"
.Run
.FileClose
```

The above example opens the script GRAPHICS.CSC, runs it, and then closes it.

{button ,AL(`cse_debug_cse;;;;;',0,"Defaultoverview",)} **Related Topics**

.AddLineAfter (Corel SCRIPT Editor)

Command: .AddLineAfter Text

Function: ReturnValue = .AddLineAfter (Text)

This command and function inserts a line and text after the line containing the insertion point.

Syntax	Description
Text	String expression that specifies the text to add to the inserted line.
ReturnValue	Specifies the numeric variable that is assigned a value corresponding to whether the line and text were added: TRUE (-1) if added; otherwise FALSE (0).

Example

```
.AddLineAfter "REM This is a comment to be added"
```

The above example adds a comment line after the line containing the insertion point. This line can also be specified as follows:

```
ReturnValue = .AddLineAfter ("REM This is a comment to be added")
```

{button ,AL(`cse_edit_commands;;;;;',0,"Defaultoverview",)} Related Topics

.AddLineBefore (Corel SCRIPT Editor)

Command: .AddLineBefore Text

Function: ReturnValue = .AddLineBefore (Text)

This command and function inserts a line and text before the line containing the insertion point.

Syntax	Description
Text	String expression that specifies the text to add to the inserted line.
ReturnValue	Specifies the numeric variable that is assigned a value corresponding to whether the line and text were added: TRUE (-1) if added; otherwise FALSE (0).

Example

```
.AddLineBefore "REM This is a comment to be added"
```

The above example adds a comment line before the line containing the insertion point. This line can also be specified as follows:

```
ReturnValue = .AddLineBefore ("REM This is a comment to be added")
```

{button ,AL(`cse_edit_commands;;;;;',0,"Defaultoverview",)} Related Topics

.DeleteLine (Corel SCRIPT Editor)

Command: .DeleteLine

Function: ReturnValue = .DeleteLine ( )

This command and function deletes the line containing the insertion point. The insertion point is then placed in the line that follows the deleted line.

Syntax	Description
ReturnValue	Specifies the numeric variable that is assigned a value corresponding to whether the line was deleted: TRUE (-1) if inserted; otherwise FALSE (0).

Example

```
.DeleteLine
```

The above example deletes the line containing the insertion point. This line could also be specified as follows:

```
ReturnValue = .DeleteLine ( )
```

{button ,AL(`cse_edit_commands;;;;;',0,"Defaultoverview",)} Related Topics

.GetLineText (Corel SCRIPT Editor)

ReturnValue$ = .GetLineText ( )

This function returns the text from the line containing the insertion point.

Syntax	Description
ReturnValue$	Specifies the string variable that is assigned the text in the line containing the insertion point.

Example

```
RV_Text$ = .GetLineText ( )
MESSAGE RV_Text$
```

The above example passes the text in the line containing the insertion point to the **RV_Text** variable. This text is then displayed in a message box.

{button ,AL(`cse_edit_commands;;;;;',0,"Defaultoverview",)} **Related Topics**

.ReplaceLine (Corel SCRIPT Editor)

Command: .ReplaceLine Text

Function: ReturnValue = .ReplaceLine (Text)

This command and function replaces the text in the line containing the insertion point with specified text.

Syntax	Description
Text	String expression that specifies the text to insert in the line containing the insertion point.
ReturnValue	Specifies the numeric variable that is assigned a value corresponding to whether the text was inserted: TRUE (-1) if inserted; otherwise FALSE (0).

Example

```
.ReplaceLine "REM This is a comment to be added"
```

The above example replaces the line containing the insertion point with a comment line. This line can also be specified as follows:

```
ReturnValue = .ReplaceLine ("REM This is a comment to be added")
```

{button ,AL(`cse_edit_commands;;;;;',0,"Defaultoverview",)} Related Topics

.GetColumnNumber (Corel SCRIPT Editor)

ReturnValue& = .GetColumnNumber ( )

This function returns the column number position of the insertion point.

Syntax	Description
ReturnValue&	Specifies the numeric variable that is assigned the column number position of the insertion point.

Example

```
RV_Number& = .GetColumnNumber ( )
```

The column number position is assigned to **RV_Number**.

{button ,AL(`cse_nav_cse;;;;;',0,"Defaultoverview",)} **Related Topics**

.GetLineNumber (Corel SCRIPT Editor)

ReturnValue& = .GetLineNumber ( )

This function returns the line number of the line containing the insertion point.

Syntax	Description
ReturnValue&	Specifies the numeric variable that is assigned the line number of the line containing the insertion point.

Example

```
.GoToEndOfDoc
RV_Number& = .GetLineNumber ( )
MESSAGE RV_Number& & " are in the active script"
```

The above example sends the insertion point to the last line in the active script. The line number is assigned to **RV_Number**. The last line displays a message box indicating the number of line in the script.

{button ,AL(`cse_nav_cse;;;;;',0,"Defaultoverview",)} **Related Topics**

.GoToColumn (Corel SCRIPT Editor)

Command: .GoToColumn ColumnNumber

Function: ReturnValue = .GoToColumn (ColumnNumber)

This command and function sends the insertion point to a specified column in the line containing the insertion point.

Syntax	Description
ColumnNumber	Specifies the column to move the insertion point to.
ReturnValue	Specifies the numeric variable that is assigned a value corresponding to whether the insertion point was sent to the specified column: TRUE (-1) if sent; otherwise FALSE (0).

Note

- If the column does not exist in the active script, the insertion point is not moved.

Example

```
.GoToColumn 14
```

The above example sends the insertion point to the fourteenth column in the line containing the insertion point. This column can also be specified as follows:

```
ReturnValue = .GoToColumn (14)
```

{button ,AL(`cse_nav_cse;;;;;',0,"Defaultoverview",)} **Related Topics**

.GoToEndOfDoc (Corel SCRIPT Editor)

Command: .GoToEndOfDoc

Function: ReturnValue = .GoToEndOfDoc ( )

This command and function sends the insertion point to the beginning of the last line in the active script.

Syntax	Description
ReturnValue	Specifies the numeric variable that is assigned a value corresponding to whether the insertion point was sent to the last line: TRUE (-1) if sent; otherwise FALSE (0).

Example

```
.GoToEndOfDoc
RV_Number& = .GetLineNumber ( )
MESSAGE RV_Number& & " are in the active script"
```

The above example sends the insertion point to the last line in the active script. The line number is assigned to **RV_Number**. The last line displays a message box indicating the number of the line in the script.

The first line can also be specified as follows:

```
ReturnValue = .GoToEndOfDoc ( )
```

{button ,AL(`cse_nav_cse;;;;;',0,"Defaultoverview",)} **Related Topics**

.GoToLine (Corel SCRIPT Editor)

Command: .GoToLine LineNumber

Function: ReturnValue = .GoToLine (LineNumber)

This command and function sends the insertion point to the beginning of a specified line.

Syntax	Description
LineNumber	Specifies the line to move the insertion point to in the active script.
ReturnValue	Specifies the numeric variable that is assigned a value corresponding to whether the insertion point was sent to the specified line: TRUE (-1) if sent; otherwise FALSE (0).

Note

- This command corresponds to the Go To dialog box. Click Search, Go To Line.
- If the line does not exist in the active script, the insertion point is not moved.

Example

```
.GoToLine 14
```

The above example sends the insertion point to the fourteenth line in the active script. This line can also be specified as follows:

```
ReturnValue = .GoToLine (14)
```

{button ,AL(`cse_nav_cse;;;;;',0,"Defaultoverview",)} Related Topics

.MoveLineDown (Corel SCRIPT Editor)

Command: .MoveLineDown

Function: ReturnValue = .MoveLineDown ( )

This command and function moves the insertion point to the line after the line containing the insertion point. The insertion point is placed at the beginning of the line.

Syntax	Description
ReturnValue	Specifies the numeric variable that is assigned a value corresponding to whether the insertion point was moved: TRUE (-1) if moved; otherwise FALSE (0).

Note

- The insertion point is not moved if it is in the first line of the script.

Example

```
.MoveLineDown
```

The above example moves the insertion point down one line in a script. This line can also be specified as follows:

```
ReturnValue = .MoveLineDown ( )
```

{button ,AL(`cse_nav_cse;;;;;',0,"Defaultoverview",)} Related Topics

.MoveLineUp (Corel SCRIPT Editor)

Command: .MoveLineUp

Function: ReturnValue = .MoveLineUp ( )

This command and function moves the insertion point to the line before the line containing the insertion point. The insertion point is placed at the beginning of the line.

Syntax	Description
ReturnValue	Specifies the numeric variable that is assigned a value corresponding to whether the insertion point was moved: TRUE (-1) if moved; otherwise FALSE (0).

Note

- The insertion point is not moved if it is in the first line of the script.

Example

```
.MoveLineUp
```

The above example moves the insertion point up one line in a script. This line can also be specified as follows:

```
ReturnValue = .MoveLineUp ( )
```

{button ,AL(`cse_nav_cse;;;;;',0,"Defaultoverview",)} **Related Topics**

.MakeCAO (Corel SCRIPT Editor)

Command: .MakeCAO CAOName

Function: ReturnValue = .MakeCAO (CAOName)

This command and function creates a Corel Add-on file.

Syntax	Description
CAOName	String expression that specifies the CAO name and path. If the CAO already exists, it will be overwritten. If the path is not specified, the CAO is saved in the active folder. You should provide a CAO extension with the CAO name.
ReturnValue	Specifies the numeric variable that is assigned a value corresponding to whether the CAO was created: TRUE (-1) if created; otherwise FALSE (0).

Note

- This command corresponds to the Make Corel Add-on dialog box in the Corel SCRIPT Editor. Click File, Make CAO.

Example

```
.MakeCAO "C:\Myfiles\tools.CAO"
```

The above example creates a CAO named TOOLS. This line can also be specified as follows:

```
ReturnValue = .MakeCAO ("C:\Myfiles\tools.CAO")
```

{button ,AL(`;cse_make;;;;;',0,"Defaultoverview",)} Related Topics

.MakeCSB (Corel SCRIPT Editor)

Command: .MakeCSB CSBName

Function: ReturnValue = .MakeCSB (CSBName)

This command and function creates a Corel SCRIPT Binary file. A Binary file is a compiled version of a script file.

Syntax	Description
CSBName	String expression that specifies the CSB name and path. If the CSB already exists, it will be overwritten. If the path is not specified, the CSB is saved in the active folder. You should provide a CSB extension with the CSB name.
ReturnValue	Specifies the numeric variable that is assigned a value corresponding to whether the CSB was created: TRUE (-1) if created; otherwise FALSE (0).

Note

- This command corresponds to the Make Corel SCRIPT Binary dialog box in the Corel SCRIPT Editor. Click File, Make CSB.

Example

```
.MakeCSB "C:\Myfiles\printers.CSB"
```

The above example creates a CSB named PRINTERS. This line can also be specified as follows:

```
ReturnValue = .MakeCSB ("C:\Myfiles\printers.CSB")
```

{button ,AL(`;cse_make;;;;;',0,"Defaultoverview",)} Related Topics

.MakeDLL (Corel SCRIPT Editor)

Command: .MakeDLL DLLName

Function: ReturnValue = .MakeDLL (DLLName)

This command and function creates a dynamic link library (DLL) from the active script. DLLs contain a library of functions that can be loaded by Corel SCRIPT or other applications at run time.

Syntax	Description
DLLName	String expression that specifies the DLL name and path. If the DLL already exists, it will be overwritten. If the path is not specified, the DLL is saved in the active folder. You should provide a DLL extension with the DLL name.
ReturnValue	Specifies the numeric variable that is assigned a value corresponding to whether the DLL was created: TRUE (-1) if created; otherwise FALSE (0).

Note

- This command corresponds to the Make DLL File dialog box in the Corel SCRIPT Editor. Click File, Make DLL.

Example

```
.MakeDLL "C:\Myfiles\finance.DLL"
```

The above example creates a DLL named FINANCE. This line can also be specified as follows:

```
ReturnValue = .MakeDLL ("C:\Myfiles\finance.DLL")
```

{button ,AL(`;cse_make;;;;;',0,"Defaultoverview",)} Related Topics

.MakeEXE (Corel SCRIPT Editor)

Command: .MakeEXE EXEName

Function: ReturnValue = .MakeEXE (EXEName)

This command and function creates a Corel SCRIPT executable from the active script. An executable is an application that runs without opening or starting the Corel SCRIPT Editor, or any other Corel application.

Syntax	Description
EXEName	String expression that specifies the executable name and path. If the executable already exists, it will be overwritten. If the path is not specified, the executable is saved in the active folder. You should provide an EXE extension with the executable name.
ReturnValue	Specifies the numeric variable that is assigned a value corresponding to whether the executable was created: TRUE (-1) if created; otherwise FALSE (0).

Note

- This command corresponds to the Make Corel SCRIPT Executable File dialog box in the Corel SCRIPT Editor. Click File, Make EXE.

Example

```
.MakeEXE "C:\Myfiles\Calendar.EXE"
```

The above example creates an executable named CALENDAR. This line can also be specified as follows:

```
ReturnValue = .MakeEXE ("C:\Myfiles\Calendar.EXE")
```

{button ,AL(`;cse_make;;;;;',0,"Defaultoverview",)} Related Topics

.SetVisible (Corel SCRIPT Editor)

.SetVisible Show

This command makes the Corel SCRIPT Editor application visible or hidden on your Windows desktop. When the Editor is hidden, it runs in the Windows background and is not visible on screen.

Syntax	Description
Show	Numeric expression that specifies whether Corel SCRIPT Editor is visible or hidden. Set to TRUE (-1) to show the Editor; otherwise set to FALSE (0).

Note

- In Windows 95, pressing CTRL+ALT+DEL opens the Close Program dialog box which indicates active applications, both visible and invisible. From Windows NT, pressing CTRL+ALT+DEL opens the Windows NT Security dialog box. Click Task list to see a listing of active applications, both visible and invisible.
- See Executing application commands in the background for more information.

Example

```
.SetVisible -1
```

The above example displays the Corel SCRIPT Editor.

{button ,AL(`;cse_make;;;;;',0,"Defaultoverview",)} **Related Topics**

String Expression

A string expression is a combination of literals, string variables, string constants, and string operators that return a string.

Numeric Expression

A numeric expression is a combination of numbers, variables, constants, functions, and operators that return a number.