

WinTask Quick Start Guide (Web automation)

Introduction

Welcome to WinTask - the premier Windows and Web automation software. WinTask makes it simple to automate the repetitive tasks you need to accomplish such as Web navigation, forms filling, data collections, downloads, Web data extraction etc.

By making these tasks automatic instead of having to do them manually all the time, you are going to:

- save time - the tasks are done for you
- improve quality - the tasks are performed exactly correct every time with no errors
- avoid repetitive and boring tasks - free you up for better things to do

WinTask is simple to learn and use. In short, nearly all the repetitive tasks that you do daily, weekly, and monthly can be automated using WinTask. For example:

- Extract data from a website and insert them in a spreadsheet (addresses capture, ebay prices capture etc.)
- Mass data-entry and transfer from a Windows application to a website.
- Automate regression web testing or software testing.
- Check the availability of Web pages and measure their performance.
- Automate report execution and distribution.
- Automatically access websites to get the latest updates for virus protection.
- Send routine reports with attachments via email on a regular schedule - daily, weekly, monthly etc.
- Add “Macros” to any software that does not have a built-in Macro function.

Our customers have used WinTask in thousands of similar applications. See [Attachment 1](#) of this Quick Start Guide for many examples of how customers are using WinTask to automate their repetitive tasks.

This Quick Start Guide will get you started creating your own automation scripts in just a few minutes.

How It Works

WinTask primarily works in an object-oriented record/playback mode. When you record a task, WinTask creates a Script that includes all the keystrokes, menu selections, clicks on links, and Web functions that you use to perform the task. To perform that task automatically, all you have to do is playback that Script. WinTask then replicates everything you did in performing that task.

To record a task:

1. Turn on the recording mode in WinTask.
2. Perform the task that you want to automate.
3. Turn off the recording mode, name the Script, and save the Script.

To perform a task:

1. Activate the Script.
2. WinTask will perform all the elements of the task.
3. WinTask will close the Script.

That's all there is to it. Also, you can schedule WinTask to perform tasks anytime that you prefer, even when you are not present.

WinTask is based upon a powerful language much like Visual Basic. Advanced users can modify Scripts and create new Scripts directly in this programming language. We have included a complete list of all the programming functions in [Attachment 2](#) so you can see that practically any repetitive task can be automated with WinTask.

Sample Tasks

Try these sample tasks:

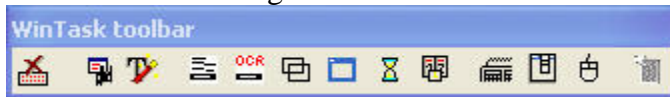
Sample Task 1.


This is an elementary task just to get you familiar with the record and playback functions. In this simple Script we will launch a demo page of our web site, click a link, enter data, and then exit. In this first sample, we will use "Your first Script wizard" which is displayed at WinTask startup.

1. Open WinTask, the "Your first Script wizard" window is displayed asking "What kind of application do you want to automate?".
2. Check **Start Internet Explorer and Record actions** or **Start Mozilla Firefox and Record actions** as we want to launch your preferred browser IE or Firefox, and open the Web page www.wintask.com/demos/index. Press **Next** button.
3. In "Give a name to your script" next screen, type the name "wintask-webexample1" in field Type a name for your Script, and press **Next** button.
4. In "Start a web site" next screen, type "www.wintask.com/demos/index" in field as we want to launch this web site. Press **Next** button.



5. In "Record your actions" next screen, click the **Rec** icon to start recording your actions in your browser.
6. The main page of www.wintask.com/demos opens and a small blinking icon appears in the right side of the bottom taskbar on your screen. This shows that WinTask is recording. The WinTask toolbar is also displayed.



7. Click the link "Form".
8. The Form page loads. In "Subject" field (a listbox), select "Web site". In "Name" field, type your name. Check the checkbox "Urgent request". And click the **Clear** button at the bottom of the form.
9. Close the browser window by clicking the x at the top right of the browser window.
10. Stop recording by pressing the blinking WinTask icon at the bottom right of the taskbar or by clicking the first icon  in the WinTask toolbar.
11. The first Script wizard screen comes back, press **Next** button in "Enhance the Script just recorded" screen as we do not need for now to edit the script generated by Recording mode.



12. In "Run your script" next screen, click **Play** icon for replaying the script that you have just created.

To replay the script later, do the following :

1. Open WinTask.
2. Open the script.
3. Press the **Playback** button (Play icon in WinTask toolbar).

4. Compilation is done in the Output window, there should be no errors and script execution starts.

WinTask does all the rest!



Sample Task 2.

Now that you are a bit more familiar with using WinTask, we will create a script that demonstrates more of WinTask's capabilities. Here is an example of accessing a website, clicking a link and downloading a file. In this case we will download a file from our website - www.wintask.com/manuals.

You could use a Script like this one to download files and information you need on a regular basis from private and public websites.

1. Open WinTask, "*Your first Script wizard*" is displayed. Check **Don't show this wizard anymore** and press **Close** button. The main WinTask window is displayed with the title "*WinTask - (Untitled1)*". If the previous script is displayed in the WinTask window, select **File/New** to create a new script. If at any time, you prefer to use the Script wizard again, in WinTask Editor menu, select **Start/New script wizard**.



2. Press **Record** button (Rec icon  in WinTask toolbar).
3. A dialog box will appear asking "*What do you want to start before recording?*".
4. Check **Internet Explorer** or **Mozilla Firefox** and press **OK** button as we want to launch your browser and open a Web page.
5. The dialog box "*Launching Internet Explorer*" or the dialog box "*Launching Mozilla Firefox*" is displayed ; in field Web address, type the Web page that the browser must open : type "*www.wintask.com/manuals.php*" ; press **OK** button.
6. The page WinTask Manuals is now displayed within your browser and WinTask Recording mode is active.
7. On the Web page, click the link "*Tutorial*".
8. The document "tutorial.pdf" loads and is displayed.
9. Close the browser window.
10. Stop recording by pressing the blinking WinTask icon at the bottom right of the taskbar or by clicking the first icon  in the WinTask toolbar.
11. The WinTask main window comes back, select **File/Save as** to save the WinTask script in any folder you like (the default folder is \WinTask\scripts) with the name "*wintask-webexample2*".
12. In WinTask main window, press the **Play** icon in the toolbar.

To replay the script later, do the following:

1. Open WinTask.
2. Open the script.
3. Press the **Playback** button (Play icon in WinTask toolbar).
4. Compilation is done in the Output window, there should be no errors and script execution starts.

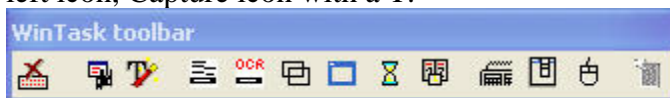
WinTask automatically downloads the Tutorial!


Sample Task 3.

This example shows how you can copy information from one source and paste it into another program. In this case we will copy information from our demo website, www.wintask.com/demos, and paste it in Notepad.

You could use a Script like this one to capture information from various sources and combine them into one resource or program.

1. Open WinTask, the main WinTask window is displayed with the title "*WinTask - (Untitled1)*". If a previous script is opened, select **File/New**.
2. Press **Record** button (Rec icon in WinTask toolbar).
3. A dialog box will appear asking "*What do you want to start before recording?*".
4. Check **Internet Explorer** or **Mozilla Firefox** and press **OK** button as we want to launch your browser and open a Web page.
5. The dialog box "*Launching Internet Explorer*" or the dialog box "*Launching Mozilla Firefox*" is displayed ; in field Web address, type the Web page that the browser must open: type "*www.wintask.com/demos/index*"; press **OK** button.
6. The WinTask Demonstration Pages page is now displayed within your browser.
7. In the WinTask toolbar which is displayed while you are recording, press the third left icon, Capture icon with a T.



8. Capture wizard screen is displayed and we will use the Spy tool in order to select and capture the content of one line.
9. Press **Spy** button; the mouse pointer changes its shape. Use the mouse to point on Web page the sentence starting at "Click the links as specified". Left click the mouse when the pointer is on that line (a black rectangle surrounds the line telling it is selected). Press **Next** button.
10. The "*Specify the HTML element where the data to be captured are*" dialog box is now displayed, it shows the text which will be captured at replay. Press **Next** button to accept.
11. The "*Take only some of the captured data*" dialog box is now displayed. As we want to extract all the captured data, press **Paste into the script** button.
12. Stop recording by pressing the blinking WinTask icon at the bottom right of the taskbar or by clicking the first icon  in the WinTask toolbar

13. WinTask main window comes back and you can see two more lines inserted in the script.
14. We will now use again the Recording mode in order to launch Notepad and paste the captured text within Notepad: Press **Record** button (Rec icon in WinTask toolbar).
15. A dialog box will appear asking "*What do you want to start before recording?*". Check **An application** and press **OK** button as we want to launch Notepad.
16. The dialog box "*Launching a program*" is displayed ; in Program field, type the word "*notepad*" and press **OK** button.
17. Notepad window is opened and Recording mode is active. Type "*The captured text is:* ".
18. In the Notepad window, select **File/Exit**. Do not save the document.
19. Close your browser window.
20. Stop recording by pressing the blinking WinTask icon at the bottom right of the taskbar or by clicking the first icon in the WinTask toolbar.
21. We have now to include the captured text: in WinTask window. Go to line:
SendKeys("The captured text is: ")
In this line, we have to add the captured text, which is in variable captured_string\$, to the text we have typed. This is done by changing the line as below:
SendKeys("The captured text is : "+captured_string\$)
22. Press the **Play** icon in WinTask toolbar, save the WinTask script in any folder you like (the default folder is \WinTask\scripts) with the name "*wintask-webexample3*".

To replay the script later, do the following :

1. Open WinTask.
2. Open the script.
3. Press the **Playback** button (Play icon in WinTask toolbar).
4. Compilation is done in the Output window, there should be no errors and script execution starts.

WinTask displays the captured paragraph within Notepad!

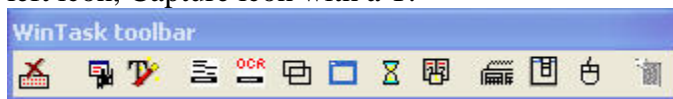
Sample Task 4.

This example shows how you can retrieve data from a Web page (from www.wintask.com/demos/data.htm) and store them in an Excel file (you will need Excel installed for this sample task).

1. Open WinTask, the main WinTask window is displayed with the title "*WinTask - (Untitled1)*". If a previous script is opened, select **File/New**.
2. Launch Excel and save an empty Excel file under the name "*c:\program files\wintask\scripts\data.xlsx*" or under the name "*c:\program files (x86)\wintask\scripts\data.xlsx*" if you use a Windows x64. If you use Excel 2003,

save the empty Excel file under the name "*c:\program files\wintask\scripts\data.xls*" or "*c:\program files (x86)\wintask\scripts\data.xls*".
Exit Excel.

3. Press **Record** button (Rec icon in WinTask toolbar).
4. A dialog box will appear asking "*What do you want to start before recording?*".
5. Check **Internet Explorer** or **Mozilla Firefox** and press **OK** button as we want to launch your browser and open a Web page.
6. The dialog box "*Launching Internet Explorer*" or the dialog box "*Launching Mozilla Firefox*" is displayed ; in field Web address, type the Web page that the browser must open: type "*www.wintask.com/demos/data.htm*" ; press **OK** button.
7. The Capture Data from a Web Table page loads.
8. In the WinTask toolbar which is displayed while you are recording, press the third left icon, Capture icon with a T.



9. Capture wizard screen is displayed and we will use the Spy tool in order to select and capture the content of the table.
10. Press **Spy** button; the mouse pointer changes its shape. Use the mouse to point on Web page the table starting at "Name". Left click the mouse when the pointer is on that table. Press **Next** button.
11. The "*Specify the HTML element where the data to be captured are*" dialog box is now displayed, it shows the text which will be captured at replay. Press **Next** button to accept.
12. The "*Take only some of the captured data*" dialog box is now displayed. Select the three columns in the table of the top. Press **Next** button.
13. The "*Specify where to copy the captured data*" dialog box is now displayed. As Excel is the default, just press **Next** button.
14. The "*Specify the Excel file where to copy the extracted data*" dialog box is now displayed. Click **Browse** button and select in the Open dialog box the Excel file that you have created in step 2. Press **Paste into the script** button.
15. The WinTask toolbar comes back, you are still in Recording mode. Close the Internet Explorer window.
16. Stop recording by pressing the blinking WinTask icon at the bottom right of the taskbar or by clicking the first icon in the WinTask toolbar.
17. Press the **Play** icon in WinTask toolbar. Save the WinTask script in any folder you like (the default folder is \WinTask\scripts) with the name "*wintask-webexample4*". Compilation is done in the Output window, there should be no errors and script execution starts.
18. Open the Excel file to check the captured data.

To replay the script later, do the following :

1. Open WinTask.
2. Open the script.
3. Press the **Playback** button (Play icon in WinTask toolbar).

4. Compilation is done in the Output window, there should be no errors and script execution starts.
 5. Open the Excel file data.xls
- The Excel file contains the captured Names, Emails and Phones!

These short Sample Tasks just touched on the surface of WinTask's capabilities. Try out WinTask on some of your repetitive tasks and see how easily it will automate these tasks.

Assistance

As you test out WinTask you may have questions or want more detailed information. We recommend:

Use the Tutorial - It includes detailed instructions to help you use WinTask in your applications. If you did not download the Tutorial in Sample Task 2 earlier, you can download it at www.wintask.com/manuals.php

Contact us by email with your questions, info@wintask.com. We would be pleased to assist you in any way possible.

If you need immediate help, use www.wintask.com/support or ask in WinTask's forum.

Purchase WinTask

When you are ready to buy WinTask, simply navigate to www.wintask.com/buy-wintask.php and complete the order form. We will promptly send you the complete WinTask program. Of course it comes with a 30-day guarantee of your satisfaction.

Attachment 1. How our customers are using WinTask.

We asked our customers to tell us how they are using WinTask. Here are a few of the many replies.

“The WinTask product is easy to use, performs as we expected and provides the functionality we require for automating test scripts. We are quite pleased with WinTask and have found it to be very useful with many of the repetitive tasks we need to perform as part of quality assurance testing. Here we used WinTask to extract a set of item numbers from an excel spreadsheet and then execute requests against our Inter/Intranet sites in order to ensure the sites were extracting information and responding to requests within a specified timeframe throughout the day.”

“Each institution has to build information having its systems as source. Then, they need to enter then in a browser form (!!!) manually. Super Wintask does the work. It reads the input file, generally in access, enters information and navigates though the browser, built a log, and at the end generates an input for other system.”

“I recorded a WinTask macro that automates my favourite searches on eBay and extracts prices from the results page.”

“I use WinTask to capture prices off webpages and "dump" them in a file or database. I use it too logging into websites with my password and place orders for products.”

“I am getting my script to: look at a spreadsheet, grab a url, paste it into internet explorer, grab a value from the webpage, paste the value into another spreadsheet. With the actual spreadsheet about 700 lines, I could easily automate this tedious task.”

“Now I can easily to collect the prices of 33 insurance companies in 20 towns for 5 client profiles: 3300 values every three months! A very impossible job without WinTask.”

“I have found WinTask particularly useful for website promotion. I use e-mail safe lists (spam free), FFA lists and Search engine submitters. WinTask easily allows me to log in, submit etc. and also perform other routine maintenance tasks. WinTask also saves me money because I don't need to buy the dedicated software that is available to do these tasks. At the moment I am saving about one and a half hours per day and I expect this saving to increase as I add more promotion services.”

“Our job is basically to navigate through customer pages and record data such as availability and response time. We enjoyed working with WinTask and had no trouble getting through the development process. Since the help subsystem is complete, we could develop easily our scripts.”

“I used WinTask on a small project we had where we needed to automate some data entry tasks on a system that had no macro language of its own. I wrote a basic script just by reading (briefly) a few of the help pages.”

“WinTask downloads 24 MorningStar Quicktake Reports and stores them in a MorningStar file on the D drive where I can access them with an Excel Macro. Before using WinTask, I had to download manually the 24 reports every week.”

“I’m using WinTask to automate my convenience store - every morning I have to go to a program called Ruby Manager and download sales to bring the data from the tills back to the PC. I set up a script to download the sales, export them to another program, and then print out the category report.”

“WinTask allows us to download a database with student information every morning at 5 AM at my school in Chicago.”

“I wrote a WinTask macro to control MS Explorer and to use arrow down to the next file, push F2 and rename the file, setting a number in the front of the filename and pushing enter.”

“I am using Wintask to avoid the boring task of running, every night, the Workspace Assistant of Tradestation, transferring the results into Excel workspaces, handle them to obtain series of recommendations for traders for next morning. The automation consists in running WA + Excel seven times per night, in total around two hours, with just one click.”

Attachment 2. List of all the functions in WinTask that can be used to automate tasks.

WinTask includes a powerful programming language similar to Visual Basic. We have listed all the functions in that language below with a brief description. You can see how comprehensive they are - practically any repetitive task can be accomplished.

When you use the Recording Mode, these functions are automatically generated by WinTask and compiled into a complete Script. Advanced users can modify or create Scripts directly in the programming language using these functions.

The functions marked with an asterisk are not available in WinTask Lite. WinTask Scheduler under XP/2003 is not available in WinTask Lite. WinTask Scheduler is not available under Vista, Windows 7 and Windows 2008, you can use the Vista, Windows 7, Windows 2008 Server built-in Scheduler.

WinTask x64 (under Windows 7 64 bit and Windows 2008 R2) includes a Scheduler which can open the desktop, run the scheduled task and close the desktop.

Window management functions

#ActionTimeout - Specifies the number of seconds that WinTask should wait before reporting an error

#ExecuteDelay - Inserts a wait for n ticks between every statement of the script

IgnoreErrors - Manages errors

#SendkeysDelay - Slows down keystrokes

#UseExact - Controls the way WinTask sends its commands to the proper Window

CheckedW* - Tells if the specified checkbox/radio button is checked or not

ChooseItem - Selects an item in a combobox or a listbox

ChooseMenu - Selects a menu item

Click - Clicks mouse on a button

ClickMouse - Clicks mouse button

ClickOnBitmap - Clicks mouse button on an image inside a window

ClickOnText - Clicks mouse button on a text inside a window

ClickOnTextOCR* - Clicks mouse button on a text recognized by the OCR engine inside a window

ClickScrollBar et WinScrollBar - Scrolls inside a window

ClickSpin - Clicks a spin control

CloseWindow - Closes the specified window

CloseWindowRegEx* - Closes the window whose name is specified by a Regular Expression

CursorX, CursorY - Returns the position of the cursor

EnabledW - Tests if the specified window is active and can receive actions

ExistW - Tells if the specified window exists

Focus\$ - Returns window name which is in focus

GetFocusWindowHandle - Returns the handle of focused window

GetTopWindowHandle - Returns the handle of the window on top

GetWindowHandle - Returns the handle of specified window

GetWindowName\$ - Returns the name of the window specified by its handle

ListItem\$ - Returns the specified item in the listbox or combobox

MaximizeWindow - Maximizes the specified window

MinimizeWindow - Minimizes the specified window

MouseShape - Returns the mouse cursor shape as an integer
MouseX , MouseY - Returns the X,Y position of the mouse
MoveMouse - Moves the mouse to the specified X
MoveWindow - Moves the specified window
RestoreWindow - Restores the specified window
SelectedItem\$* - Returns the item selected in the specified listbox or combobox
SendKeys - Sends keystrokes to the window specified by the last UseWindow
SendKeysEncrypted* - Sends encrypted keystrokes to the window specified by the last UseWindow
SizeWindow - Modifies the size of the specified window
Top\$ - Returns the name of the main window which has focus
TopInstance - Returns the instance number of the main window in focus
UseWindow - Specifies the window where the script will now send its keystrokes
UseWindowHandle - Specifies the window (through the window handle) to which subsequent keyboard, mouse and menu actions are directed
UseWindowRegExp* - Using a Regular Expression, specifies the window where the script sends keys
WinScrollBar et ClickScrollBar - Scrolls inside a window
WriteCombo - Writes text in the edit zone of a listbox
WriteEdit - Writes text in the specified edit zone
WriteEditEncrypted* - Writes encrypted text in the specified edit zone

Capture functions

#UsePageExact - Controls the html pages finding method
Capture\$ - Captures the text in the specified window
CaptureArea\$ - Captures the text in the specified window area
CaptureAreaOCR\$* - Using OCR, captures the text in the specified window area
CaptureBitmap - Captures an image and stores it in a .BMP file
CaptureHTML - Captures the text of a specified html element
CaptureOCR\$* - Using OCR, captures the text in the specified window
CaptureTableHTML - Captures a range of cells in a specified html table
CopyLink* - Copies the link associated with the specified html element into a string
GetHTMLEditText - Captures the text of the specified html Edit element
HardCopy* - Saves a screenshot
UseOCREngine* - Specifies the OCR engine to use (WinTask or MODI OCR engine)

Synchronization functions

#ErrorCode\$* - Gives the error code for the error that triggered OnAction Error procedure
#ErrorFunction\$* - Gives the function where the error triggering OnAction Error procedure occurred
#ErrorLine\$ - Gives the script line where the timeout has occurred
#ErrorMsg\$* - Gives the error message for the error that triggered OnAction Error procedure
#ErrorScript\$* - Gives script name where the error triggering OnAction Error procedure occurred
#PauseTimeout - Specifies the maximum delay for a Pause statement
#TextlookFrequency - Defines the frequency of scrutation of window where texts are watched for
Disable* - Disables the management of a specified event
Enable* - Reactivates the management of a specified event
OnAction ... EndAction* - Manages events
OnAction Error ... EndAction* - Inserts and manages error events
Pause - Waits for a specified amount of time
Pause ... Until - Waits for an action
Sleep* - Makes the current script sleep whereas the events are still managed

User dialog

BeginDialog...EndDialog* - Defines a dialog box with its controls
CallDialog* - Displays a dialog box defined previously
Inputbox\$ - Displays a simple dialog box where the user can type a value
InputboxSecret\$* - Displays a simple dialog box where the user can type a hidden value
MsgBox - Displays a Windows standard message box
MsgFrame - Displays a message
MsgFrameTitle - Displays a message with a title
RemoveFrame - Removes the message displayed by MsgFrame
SelectDir* - Returns the name of the directory selected by the user in the standard Browse for Folder dialog
SelectFile* - Returns the name of the file selected by the user in the standard dialog box File Open
SelectMultipleFile* - Returns the name of the files selected in the standard dialog box File Open

File management functions

AppendXMLNode* - Adds a node in the specified XML file
ChDir - Sets the current working directory
CloseExcelCom - Closes the background Excel instance loaded by WriteExcel or ReadExcel
Create - Creates a file
CreateExcelFile - Creates an Excel worksheet
CreateUnicodeFile - Creates a Unicode file
CurDir\$ - Returns the current working directory
DelTree - Deletes all the files and sub-directories below the specified directory
Dir - Puts file names from a directory into arrays
DiskFree - Returns the available space on the specified resource
EnumXMLAttributes* - Retrieves the attribute names and values for the specified XML node
EnumXMLChildren* - Enumerates the child node descriptors for the specified XML node
Eof - End of file
Exist - Tells if specified file exists
ExistDir - Checks for the existence of the specified directory
FileAttr\$ - Gives the attributes of the specified file
FileCopy - Copies a file to an other file
FileDate\$ - Date of last modification of the specified file
FileSize - Size of the specified file
FileTime\$ - Time of last modification of the specified file
FileVersion\$ - Version number of the specified file
GetReadPos - Value of the reading pointer of the specified file
GetXMLAttribute* - Retrieves the content of an attribute in the specified XML file
Kill - Deletes one or several files
MkDir - Creates a directory
Name - Renames or moves one or several files
Read - Reads data from a file
ReadExcel - Reads a range from an Excel workbook
ReadIni\$* - Reads a parameter in the specified INI file
Rmdir - Deletes a directory
SetAttr - Sets the attributes of one or several files
SetReadPos - Sets the reading pointer to the specified value
SetXMLAttribute* - Modifies or adds an attribute in the specified XML file
WinDir\$ - Returns the name of the directory where Windows is installed
Write - Writes data in a file

WriteExcel - Writes in an Excel workbook
WriteIni* - Writes to the specified INI file

Flow control functions

#ErrorLine\$ - Gives the script line where the timeout has occurred
#ExecTimeout - Sets the maximum delay before stopping script execution
#ExecuteDelay - Slows down a running script by inserting a wait for n ticks between every statement
#IgnoreErrors - Manages errors
#LastErrorLine* - Gives the line number where the error triggering OnAction Error procedure occurred
#ScriptAfterTimeout* - Specifies the script to run after execution timeout has elapsed
Command\$ - Allows a calling script to use the parameters from the called script
End - Stops the current running script
Function ... ExitFunction ... EndFunction - Defines a function
Goto ou Go to - Makes the execution of the script continue at another line
If ... Then ... Else ... Endif - Decision making statement
Repeat ... until ... - Loop with test at the end of the loop
Run - Launches a compiled script as a sub-program
Select Case ... EndSelect - Multiple decision making statement
Shell - Executes a program (.exe, .com, .bat, .doc, .txt, ...)
Stop - Stops all the scripts
Sub ... Exitsub ... EndSub - Defines a procedure
While ... Wend - Loop statement with test at the beginning

String management functions

Asc - Returns the numeric ASCII code of the first character in a specified string
Chr\$ - Converts an ASCII value in its equivalent ASCII character
Encrypt* - Encrypts the specified string
ExtractBetween\$ - Extracts a string between two strings
Instr - Returns the position of one string within another
InstrRev - Returns the position of one string within another, searching backward through the string
Lcase\$ - Converts all uppercase characters in the specified string to lowercase
Left\$ - Extracts the specified number of characters from the left hand portion of the specified string
Len - Returns the length of the specified string
Ltrim\$ - Returns the specified string minus its leading spaces and tabulations
Mid\$ - Retrieves a substring from the specified string
Replace\$ - Finds and replaces some or all occurrences of a substring within the specified string
Right\$ - Returns the rightmost portion of the specified string for the numbers of characters specified
Rtrim\$ - Returns the specified string minus its trailing spaces and tabulations
SplitIntoArray - Converts the specified string into an array of strings
Str\$ - Transforms the specified numeric value in a string
Trim\$ - Returns the specified string minus its trailing spaces and leading spaces
Ucase\$ - Converts all lowercase characters in the specified string to uppercase
Val - Returns the numeric value of a string

Date/time functions

Date\$ - Returns the current date
DateBetween\$ - Returns the number of specified time intervals between two dates

DateToDate\$ - Returns a new datetime based on adding an interval to the specified date
Day\$ - Returns the current day number within the current month
Hour\$ - Returns the current hour as a two-character string
Hundreds\$ - Returns hundredth seconds of system time as an integer from 0 to 99
Min\$ - Returns the minutes of the current hour as a two character string
Month\$ - Number of the current month as a string
Sec\$ - Returns the seconds of the current hour as a two character string
Time\$ - Returns the system clock as a string
WeekDay - Returns the current day of the week
Year\$ - Returns the current year as a string

System functions

#HideIcon - Hides the WinTask icon in the taskbar
#HideTrayIcon - Hides the WinTask icon in the system tray
#IgnoreErrors - Manages errors
Allocate* - Reserves a memory area for data used by external DLL
Beep - Forces the PC to emit a sound through the PC speaker
CapsLock - Forces the capslock key to the specified state
ChDir - Specifies the current working directory
Curdir\$ - Returns the current working directory
DeleteRegKey* - Deletes the specified key in Registry
DeleteRegValue* - Deletes a value in Registry
Dir - Puts in arrays all or some files present in a directory
DirTree - Puts file names and directory names into arrays
DiskFree - Returns the available space on the specified resource
Envir\$ - Returns the value of an environment variable
ExecExcelMacro - Executes an Excel macro in the specified Excel book
Exist - Checks for the existence of the specified file
External* - Calls an external DLL
External\$* - Calls a Windows DLL
GetCpuLoad* - Returns the CPU load percentage
GetMemUsage* - Returns the memory used percentage
GetProcessCpuLoad* - Returns the CPU percentage used by a process
GetProcessList* - Gives the list of active process and their attributes
GetWindowsList* - Gives the list of parent window names present on desktop
ImpersonateUser* - Allows WinTask to acquire additional rights
IsRunning - Tells whether a program is loaded in memory or not
KillApp* - Kills the specified application
KillAppChildren* - Kills the specified application and its associated children
KillProcess* - Kills the specified process
LockKbd - Locks the keyboard
LockMouse - Locks the mouse
MkDir - Creates a directory
NumLock - Forces the numlock key to the specified state
OsVersion\$ - Returns Windows version
PeekInteger* - Reads one or several bytes in memory and returns an integer
PeekString\$* - Reads a string in memory
PokeInteger* - Writes in memory a value of type integer or Unsigned
PokeString* - Writes in memory a string
Random - Returns a random integer
ReadIni\$* - Reads a parameter in the specified .INI file
ReadReg* - Reads an integer or a string from Registry

Reboot - Reboots the PC or Windows
RevertToSelf* - Cancels an impersonation made previously by ImpersonateUser
Rmdir - Deletes a directory
SendEmail* – Sends an email using the SMTP outgoing mail server defined in WinTask Scheduler
Shell - Executes a program
ShellWait - Executes a program (.exe, .com, .bat, .doc, .txt, ...) and waits for its termination before running next statement
UnlockKbd - Unlocks the keyboard
UnlockMouse - Unlocks the mouse
WinDir\$ - Returns the directory where Windows is installed
WriteIni* - Writes in the specified .INI file
WriteReg* - Creates or modifies a string or numeric value in Registry

Clipboard and log functions

#Current line - Returns the current executed line in the script
Comment* - Writes a comment in the log file
GetClipboard\$ - Returns the text contained in Clipboard
LogFile* - Forces the script to log its actions in the specified logfile
SetClipboard - Puts the specified string into the Clipboard
StopLog* - Stops recording in the logfile

Compilation

Dim - Defines an array
Include - Includes the specified source file in the current script
Local - Defines a local variable
Rem - Inserts a comment
Unsigned - Defines an Unsigned variable

Services management *

IsServiceStarted* - Tells if the specified service is started or not
StartService* - Starts the specified service
StopService* - Stops the specified service

Web functions

#HTMLBrowser\$ - Specifies the browser to use in the subsequent Web functions
#HTMLPosRetry - Controls the way HTML element coordinates are found
#IgnoreHTMLCase – Enables/disables character case in HTML descriptors
#UsePageExact - Controls the html pages finding method
CaptureHTML - Captures the text of a specified html element
CaptureIE\$ - Captures in text mode what appears in a HTML window
CaptureTableHTML - Captures a range of cells in a specified html table
CheckedHTML* – Gives the check state of an HTML check box or radio button
ClickHTMLElement - Clicks the specified html element in the current page
CloseBrowser - Closes the opened instance of Internet Explorer browser

CopyLink* - Copies the link associated with the specified html element into a string
CurrentPage\$ - Gives the title of the current Web page
EnabledHTMLElement* - Checks if the specified html element is enable or disable
ExistHTMLElement* - Checks for the existence of the specified html element
ExtractLink* - Returns all the links of the child elements of the specified html element
GetFrameSource\$* - Returns the source code of the specified frame in the current Web page
GetHTMLEditText - Captures the content of an html Edit field within a Web form
GetPageSource\$* - Returns the source of the current Web page
ListHTMLItem\$ - Returns the specified item from the specified html listbox or combobox
Navigate - Navigates to the specified url
OverHTMLElement - Moves the mouse over the specified html element
PreviousPage - Clicks the Back button in the current browser window
SavePictureAs - Saves an html element referring to a picture
SaveTargetAs - Simulates a right click on an html element and selection of Save Target As option
SelectedHTMLItem\$* - Returns the item selected in the specified combobox/listbox html element
SelectHTMLItem - Fills Combobox/Listbox in the current Web page
StartBrowser - Starts Internet Explorer browser
UsePage - Specifies the html page used by web functions
WriteHTML - Types a string in a html edit zone
WriteHTMLEncrypted* - Types an encrypted string in a html edit zone
WriteHTMLPaste - Pastes a string in a html edit zone

FTP functions

#FTPTimeout - Specifies the number of seconds which WinTask should wait before reporting a runtime error when it tries to execute a FTP function
FTPChDir - Specifies the new FTP current folder
FTPConnect - Makes a connection to the specified FTP server
FTPCurrentDir - Returns the FTP current folder
FTPDisconnect - Terminates the connection to a FTP server
FTPExistDir - Checks if the specified FTP folder exists or not
FTPExistFile - Checks if the specified FTP file exists or not
FTPGetFile - Downloads one or several files to the local PC from the FTP server
FTPKill - Deletes one or several files from the FTP server
FTPMkDir - Creates a folder on the FTP server
FTPName - Renames one or several files in the FTP server
FTPPutFile - Uploads one or several files from the local PC to a FTP folder
FTPRmdir - Deletes a folder and its contents on the FTP server

Real calculation functions

#DecimalSeparator - Specifies the decimal separator character used for floating point numbers
#Precision - Specifies the number of decimal places for floating point calculation functions
Add\$ - Adds two strings representing floating point numbers
Divide\$ - Divides two strings representing floating point numbers
Multiply\$ - Multiplies two strings representing floating point numbers
Subtract\$ - Subtracts two strings representing floating point numbers

Response time* functions

ResetTimer* - Resets the specified clock
StartTimer* - Starts the specified clock
StopTimer* - Stops the specified clock
Timer* - Returns the value of the specified clock

ODBC* functions

#DbDateFormat* - Controls the date fields format
DbBof* - Tells if recordset contains no records
DbClose* - Closes the recordset
DbConnect* - Establishes the data source connection through the odbc driver
DbDisconnect* - Closes the data source connection
DbEof* - Tells if pointer is at the end of the recordset
DbExecute* - Executes a sql command on the opened data source
DbGetFieldNumeric* - Retrieves the value of a numeric field in a recordset
DbGetfieldString* - Retrieves the value of a string field in a recordset
DbMove* - Moves the current record pointer within the recordset at specified position
DbMoveFirst* - Positions the current record on the first record in the recordset
DbMoveLast* - Positions the current record on the last record in the recordset
DbMoveNext* - Positions the current record on the next record in the recordset
DbMovePrev* - Positions the current record on the previous record in the recordset
DbRecordCount* - Returns the number or records in the recordset
DbSelect* - Fills the recordset by retrieving in the table the records matching the sql query