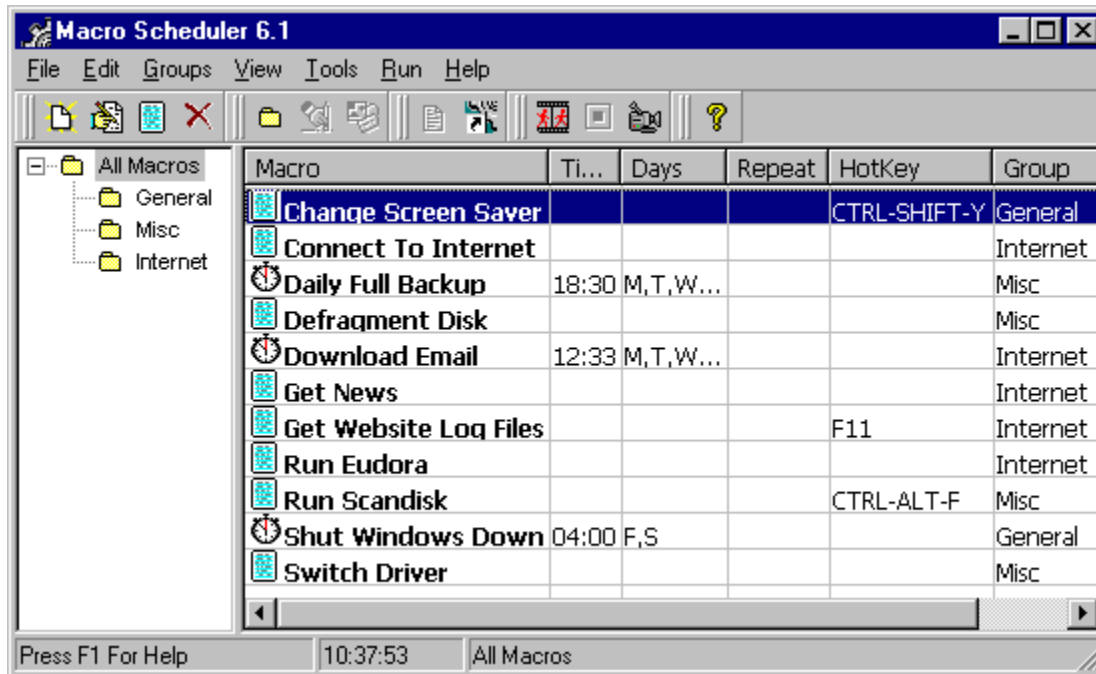


Getting Started

When Macro Scheduler is started for the first time the main form will appear. It will also place it's icon in the system tray next to the clock. When you minimize Macro Scheduler, it will minimize to the system tray. To restore it, double click on it's icon or right click on the icon to display the pop up menu and select 'Show'. In NT3.5x Macro Scheduler is simply minimized to an icon. Double click on the icon to access its functions.

If you like you can set Macro Scheduler to start minimized. To do this select Tools/Options and check 'Start Minimized'. See [Options](#).



This is the main control centre for Macro Scheduler. From here macros can be started, recorded, deleted and edited.

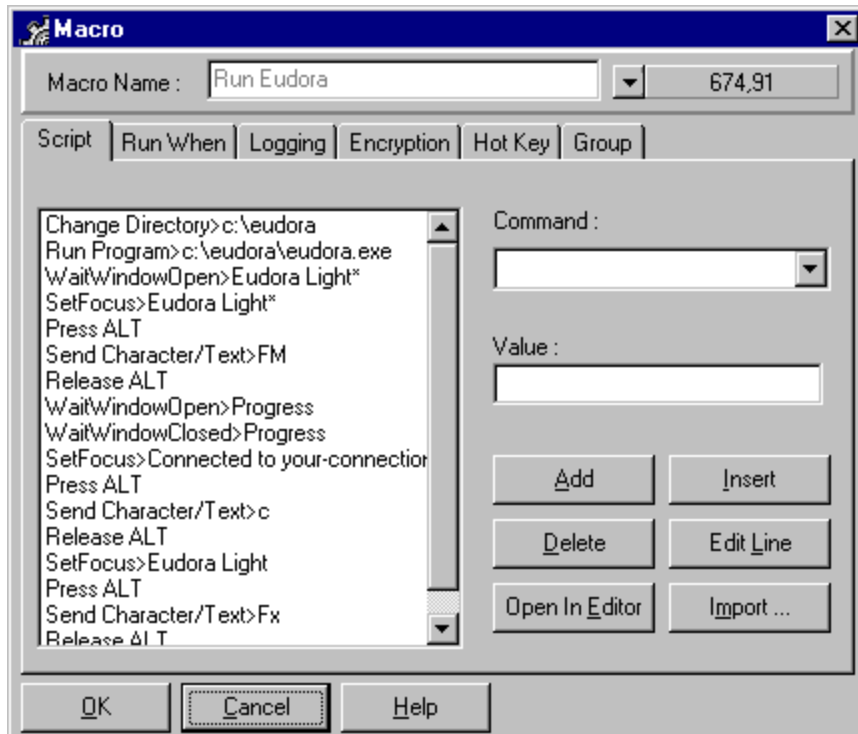
For further information on creating macros see [Creating Scripts](#) and [Recording Macros](#)

Or see [Menu Commands](#) or [Toolbars](#) for an explanation of the menu options and toolbar buttons.

Access to the functions is through the toolbar buttons, the system menu or by double clicking on the appropriate macro, or right clicking on it to display a pop up menu.

Creating Scripts

To create a new script select 'File/New Macro', or click the 'New' button on the [toolbar](#). To Edit an existing script, select it from the list by clicking on it and then select 'File/Macro Properties' or double click on it. Having done this you will be presented with the following window:



In this example an existing script was selected to be edited. If 'New' had been pressed then this form would appear blank.

There are two ways of building scripts from this window. One way is to use the drop down list of commands and the other is to use the built in editor. The latter method is recommended only for more experienced users and is great for maintaining long scripts. See [Using The Editor](#) for more information.

To build your script using the drop down list, simply select the desired script command, enter a value if one is required and then click 'Add' to add it to the end of the script. To insert a line, highlight the line you want to insert above and click 'Insert' after choosing the right command. You can delete a line from the script by selecting it and clicking on 'Delete'.

To help you find the required command, you can sort the command list by right clicking on the drop down box and selecting 'Sorted'.

If you choose a command that requires a parameter value, enter one in the box marked 'Value'. If you forget Macro Scheduler will tell you what you need to enter. For detailed help on a command, select it from the list and then press F1.

If you need to edit a line quickly, highlight the line and click 'Edit Line'. The value will then appear in the value box and the 'Edit Line' button will now say 'Update'. After editing the value click on 'Update' and the script will be updated.

The button marked 'Open In Editor...' opens the script in the [editor](#). The Import button allows you to load in

a script that has already been created. This is useful if you have a number of Macro Scheduler installations and want to make use of a script created on a different PC for example.

The numbers at the top right of this window show your mouse cursor position. Use these to determine the correct parameters when using the MouseMove command, much easier and quicker than guessing !! The button to the left of the numbers will reveal a drop down menu. On here you can toggle between absolute coordinates and relative coordinates. When set to relative, the numbers show you the relative coordinates to the window the cursor moves over. The other option allows you to attach a small tag to the mouse cursor that shows the coordinates and follows the cursor around. This means that if the script window becomes concealed by another application you can still see the cursor position. The last option of the drop down menu is used to add the pixel colour to the display as well as the cursor position. This is helpful for the [WaitPixelColor](#) command.

Once you have created your script press 'OK' to save it and return to the main window. Press 'Cancel' to leave without saving the changes.

MacroScript Commands

<u>Add</u>	<u>Ask</u>	<u>Ascii</u>
<u>CapsOff</u>	<u>CapsOn</u>	<u>Change Directory</u>
<u>CloseWindow</u>	<u>ConCat</u>	<u>CountFiles</u>
<u>CopyFile</u>	<u>CreateDir</u>	<u>DateStamp</u>
<u>Day</u>	<u>DayOfWeek</u>	<u>DeleteFile</u>
<u>DDEPoke</u>	<u>DDERequest</u>	<u>EditIniFile</u>
<u>ExecuteFile</u>	<u>FileDate</u>	<u>FileSize</u>
<u>FindWindowWithText</u>	<u>FTPDelFile</u>	<u>FTPGetDirList</u>
<u>FTPGetFile</u>	<u>FTPPutFile</u>	<u>FTPRenameFile</u>
<u>GetActiveWindow</u>	<u>GetCheckBox</u>	<u>GetClipboard</u>
<u>GetCursorPos</u>	<u>GetDate</u>	<u>GetFileList</u>
<u>GetPixelColor</u>	<u>GetRectChecksum</u>	<u>GetTime</u>
<u>GetWindowPos</u>	<u>Goto</u>	<u>Hour</u>
<u>HTTPRequest</u>	<u>If</u>	<u>IfFileChanged</u>
<u>IfFileExists</u>	<u>IfWindowOpen</u>	<u>Input</u>
<u>Label</u>	<u>LClick</u>	<u>LDbClick</u>
<u>LDown</u>	<u>Length</u>	<u>Let</u>
<u>LUp</u>	<u>Macro</u>	<u>MClick</u>
<u>MDbClick</u>	<u>MDown</u>	<u>Message</u>
<u>MessageModal</u>	<u>MidStr</u>	<u>Min</u>
<u>Month</u>	<u>MouseMove</u>	<u>MouseMoveRel</u>
<u>MouseOver</u>	<u>MoveFile</u>	<u>MoveWindow</u>
<u>MUp</u>	<u>NumOff</u>	<u>NumOn</u>
<u>PlayWav</u>	<u>Position</u>	<u>Press...</u>
<u>PushButton</u>	<u>PutClipboard</u>	<u>Random</u>
<u>RClick</u>	<u>RDbClick</u>	<u>RDown</u>
<u>ReadIniFile</u>	<u>ReadLn</u>	<u>RegistryDelKey</u>
<u>RegistryDelVal</u>	<u>RegistryReadKey</u>	<u>RegistryWriteKey</u>
<u>Release...</u>	<u>Remark</u>	<u>Repeat</u>
<u>ResizeWindow</u>	<u>Run Program</u>	<u>RUp</u>
<u>ScrollOff</u>	<u>ScrollOn</u>	<u>Sec</u>
<u>Send Character/Text</u>	<u>Separate</u>	<u>SetCheckBox</u>
<u>SetFocus</u>	<u>ShutdownWindows</u>	<u>SMTPSendMail</u>
<u>Sub</u>	<u>TimeStamp</u>	<u>Until</u>
<u>VBEND</u>	<u>VBEval</u>	<u>VBSTART</u>
<u>VBRun</u>	<u>Wait</u>	<u>WaitCursorChanged</u>
<u>WaitKeyDown</u>	<u>WaitPixelColor</u>	<u>WaitReady</u>
<u>WaitRectChanged</u>	<u>WaitWindowClosed</u>	<u>WaitWindowOpen</u>
<u>WindowAction</u>	<u>WriteLn</u>	<u>Year</u>

System Variables

OS_VER	Operating System
WIN_DIR	Windows Directory Path
SYS_DIR	Windows System Directory Path

The above variables store their values in upper case.

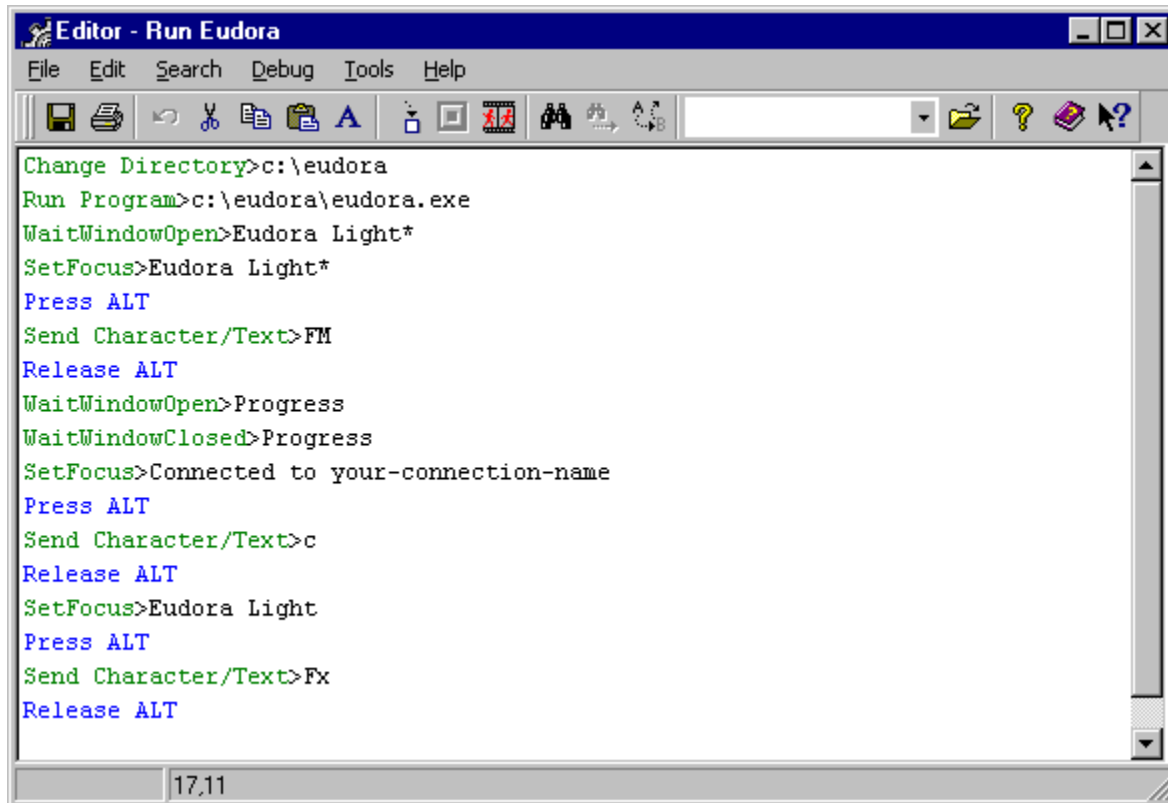
USER_NAME	Current Username
COMPUTER_NAME	Computer Name

WW_TIMEOUT	Timeout value from <u>WaitWindowOpen</u> / <u>Closed</u>
WW_RESULT	Result from <u>WaitWindowOpen</u> / <u>Closed</u>

WCC_RESULT	Result from WaitCursorChanged
DDE_TIMEOUT	Timeout value from DDERequest
RP_WINDOWMODE	Used to set window mode for Run Program
RP_RESULT	Used to store return code from Run Program
RP_DISPLAYERROR	Allows Run Program error messages to be turned off/on
MSG_STAYONTOP	Used to set message to stay on top in Message / MessageModal
MSG_CENTERED	Used to set message box to center
SK_DELAY	Millisecond delay to pause between sending characters in Send Character/Text
RND_SEED	Allows a seed to be set for the Random command
CF_OVERWRITE	Allows changing CopyFile to overwrite or rename on collision
FTP_STATUS	Used to switch off/on the FTP status window
FTP_PASSIVE	Used to toggle FTP passive mode
FTP_TIMEOUT	Timeout value for FTPGetFile / FTPPutFile / FTPGetDirList
FTP_RESULT	Result of FTP commands
SENDMAIL_STATUS	Used to switch off/on the SMTPSendMail status window
SMTP_RESULT	Result of SMTPSendMail command.
SMTP_AUTH	Used to set SMTP authentication on or off
SMTP_USERID	Used for SMTP authentication
SMTP_PASSWORD	Used for SMTP authentication
SMTP_RECEIPT	Used to enable SMTP return receipt
SMTP_PORT	Used to optionally set the port of the SMTP server
REG_INTASSTR	Used to set RegistryWriteKey to write integers as strings/integers
WF_TYPE	Used to set the type of window the windowing functions should affect
VBS_TIMEOUT	Used to set the timeout for VBScript code
CR	Carriage Return
LF	Line Feed
CRLF	Carriage Return, Line Feed Combination (to force a new line)

Using The Editor

While the method mentioned in [Creating Scripts](#) is very easy to use and avoids any errors occurring, it can become a bit laborious when creating very long and complicated scripts, especially if you need to perform cut and paste operations with large or repeated chunks of code. This is where the editor comes in.



To open your script in the editor simply edit the script in the usual way and then click on the 'Open In Editor...' button on the [script settings window](#). Alternatively you can now go direct to the Editor by selecting 'File/Edit Script' on the [main menu](#) or clicking the 'Edit Script' [toolbar](#) button.

As a guide there is also a drop down command list on the panel which will write out your chosen command in the editor. Select the desired command and then press Enter, or select 'Insert' from the popup menu which appears when you right click on the command list box. This is useful for avoiding spelling mistakes and getting the case wrong. The command list can be sorted by right clicking to reveal the popup menu and selecting 'Sort List'. It can also be set to show only your favorites (see [Command Locator](#)).

The usual editing functions are available from the toolbar buttons, menu bar, or right mouse button, to cut, copy and paste. These operations can also be achieved using the standard windows shortcut keys (CTRL-X, CTRL-C, CTRL-V).

The script can be printed by clicking on the printer button. If you're not sure what a button does just hover the mouse cursor over it to reveal the tool tip. The editor also has find and replace features available from the menu bar or the toolbar buttons. An "Indent Block" option on the Edit menu allows you to indent the highlighted lines of text.

For help with debugging, see [Using the Debugger](#).

When you have finished editing press the 'Save' button or select File/Save and close the editor by selecting File/Close, or by clicking on the standard close button.

To change the default font name and size use the font button represented with a capital A.

Using an External Editor

It is possible to assign an editor of your choice to Macro Scheduler, so that when you click on the edit script button on the main form, your script opens in your preferred editor. See [Options](#) for details on how to do this. You can also open script files as you would any file. Your script files have the extension .scp and you will find them in the Macro Scheduler directory.



Boxer 99 Text Editor

If you do wish to use an external editor, we strongly recommend Boxer 99, from Boxer Software. Boxer 99 is a powerful, full-featured text editor which will appeal to programmers, writers, engineers, students and others. Boxer features color syntax highlighting and printing, column blocking, undo and redo, regular expression searches and much, much more. An *extensive* array of configuration options are available, including the ability to define key assignments, screen colors and fonts, templates, and even to define external programs as tools.

Thanks to Boxer's pre-defined Macro Scheduler syntax information, Macro Scheduler scripts are automatically syntax highlighted. You can also use Boxer's pre-defined Template Set for Macro Scheduler, which includes Templates for the most common scripting commands. When used in this way, Boxer and Macro Scheduler complement each other's considerable capabilities.

You will find an evaluation copy of Boxer 99 on your Macro Scheduler CD-ROM. Or, if you received Macro Scheduler via the Internet, you can download Boxer 99, and get more information, from <http://www.boxersoftware.com>

Special Offer

We can foresee that some users of Macro Scheduler may wish to use Boxer 99, so we've arranged special pricing with Boxer Software for our customers who wish to order this product. Licensed users of Macro Scheduler are entitled to purchase Boxer 99 for just \$41.30. This is 30% off the regular price, saving you \$14.70. MJT Net Ltd has arranged to serve as an agent for Boxer Software in accepting and delivering these orders. You can place your order for Boxer 99 directly with us when you order Macro Scheduler, or at some later time.

Notes

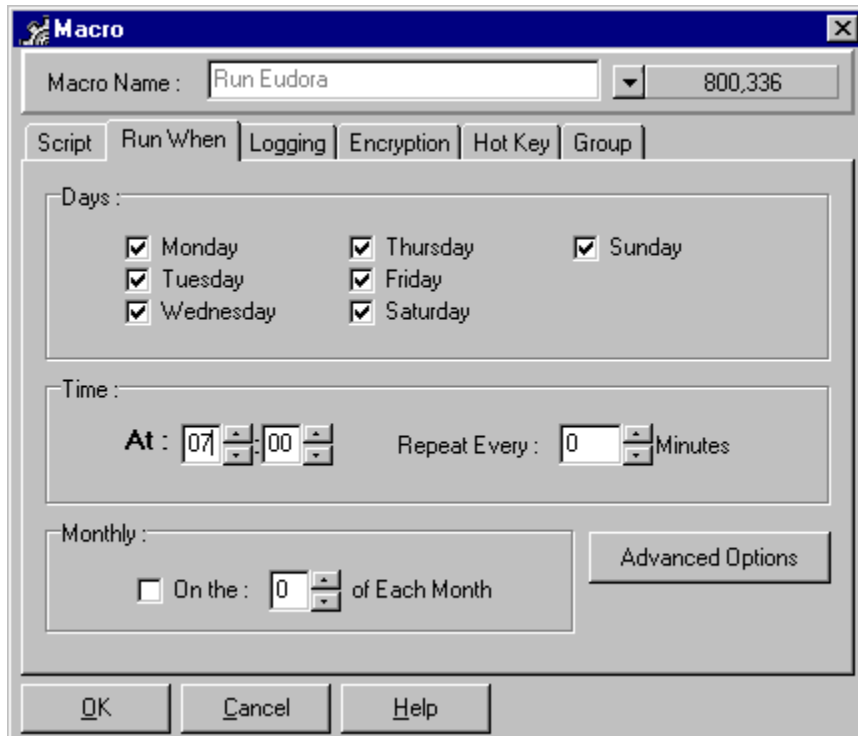
- The Boxer discount offer is limited to single-user licenses. If you have purchased a Multi-User License for Macro Scheduler, and also wish to purchase multiple copies of Boxer, please contact Boxer Software directly for their multi-user pricing.
- In order to take advantage of this special offer, you must mention it when ordering by phone, email, fax or mail. This offer is not available via our online ordering methods.

- Technical support for Boxer will be provided by Boxer Software.

Scheduling Scripts

Once you have created your macro you will probably want to execute it. Macros can be run at any time from the main window, from Windows shortcuts, from the command line or to a specified schedule.

To set up a schedule, select the appropriate script and choose to edit it to invoke the [Macro Properties Form](#). Then select the tab marked 'Run When' to display the following options.



The screenshot shows the 'Macro' window with the 'Run When' tab selected. The 'Macro Name' is 'Run Eudora' and the ID is '800,336'. The 'Days' section has checkboxes for all days of the week (Monday through Sunday), all of which are checked. The 'Time' section has 'At' set to 07:00 and 'Repeat Every' set to 0 minutes. The 'Monthly' section has a checkbox for 'On the' followed by a spinner set to 0, and the text 'of Each Month'. An 'Advanced Options' button is located to the right of the 'Monthly' section. At the bottom are 'OK', 'Cancel', and 'Help' buttons.

Mark off the days on which you want the macro to run and enter a time. The time must be entered in 24 hour notation. If you want the macro to be repeated enter an appropriate value in the 'repeat every' box. If you don't want it repeated simply leave this set to zero. The repeat every box allows you to specify upto 1440 minutes (a full day).

To make the macro run on a monthly basis check the monthly box and enter the day of the month on which you want the macro invoked.

If you choose to run a macro on the 5th of the month and also check the Friday box, the macro will run every Friday AND on the 5th of the month regardless of what day the 5th is.

In the 'Advanced' scheduling settings you can specify what should happen when Macro Scheduler restarts. This can be useful if you have Macro Scheduler start every time you reboot your computer. For instance, you can set Macro Scheduler to continue to repeat when the computer restarts. You can also determine what should happen if a schedule was missed while the computer was turned off, and you can specify a window title so that the macro starts when that window appears. See [Advanced Scheduling Options](#).

If you have created a schedule for a macro, but wish to temporarily stop scheduling without changing the schedule details, you can do so from the pop up menu of the script list on the main Macro Scheduler window.

A Note About Screensavers

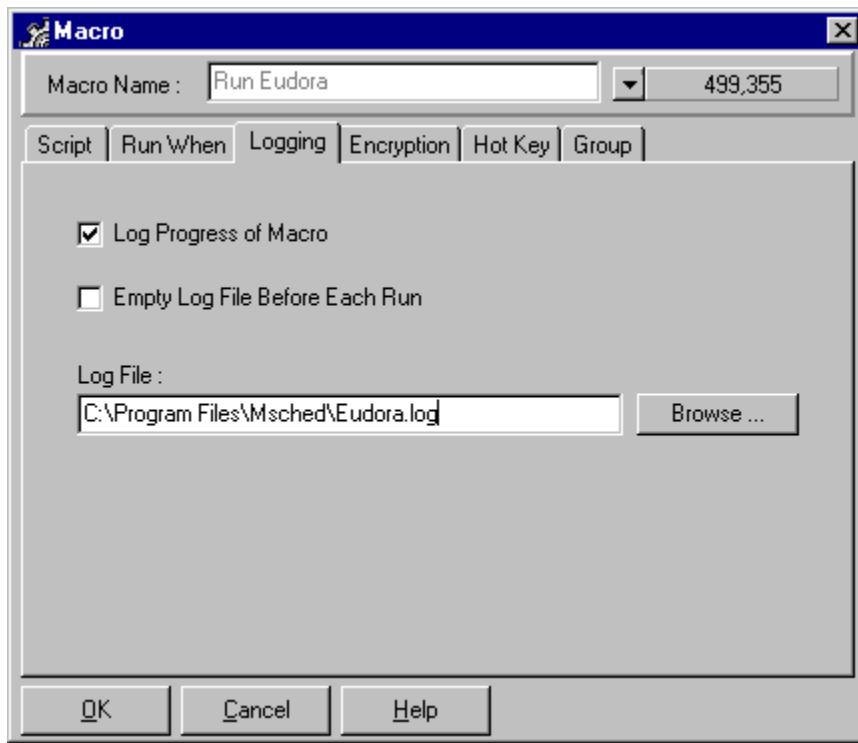
Screensavers usually stop successful detection of other windows. Consequently if a script that needs to setfocus or wait for windows to appear is run while a screensaver is active it may not work correctly.

To get round this Macro Scheduler temporarily disables screensaving just before it runs a script and re-enables screensaving when the script completes. It also attempts to determine if a screensaver is currently active and if so closes it down. However, there are many different implementations of screensavers which operate in different ways, making their detection and close down a rather unreliable process. To try to ensure that Macro Scheduler is successful in closing an active screensaver it briefly moves the mouse back and forth before running the script.

This should work in most cases. However, if you find that Macro Scheduler is unable to stop your screensaver, then the most reliable method of making sure that a scheduled macro runs properly is to simply disable screensaving altogether.

Logging

To set up a log file for a script select the tab marked 'Logging' on the [macro settings window](#).



To enable logging check 'Log Progress of Macro'.

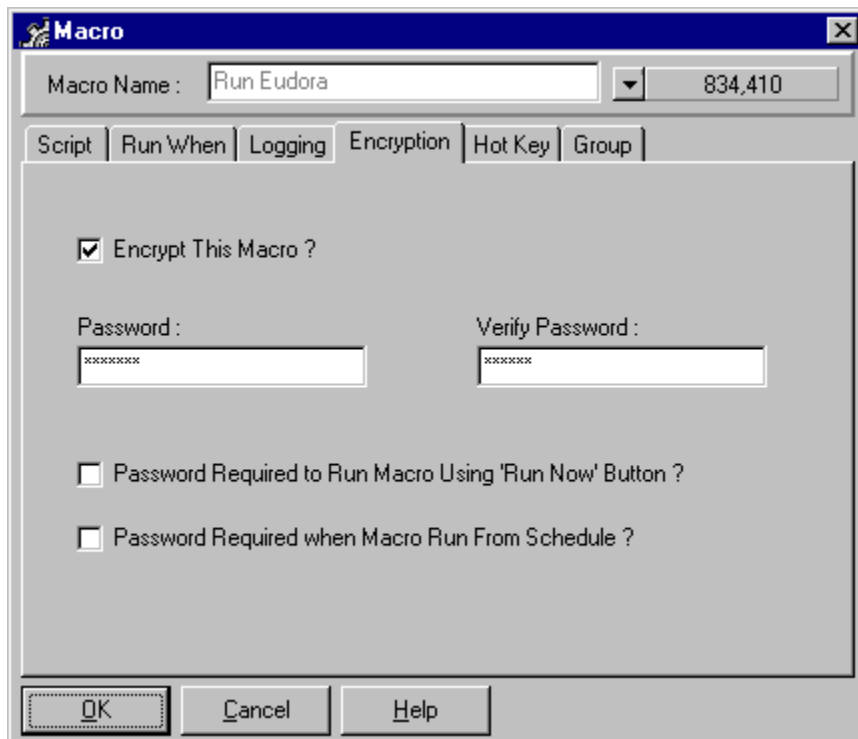
You can have Macro Scheduler purge the log file before each run by ticking the second box.

Enter a file name for the log file or select an existing one by using the browse button. If you like you can use one file for more than one macro.

Encryption

The majority of us probably won't ever have to use the encryption facility. However, if you need to use Macro Scheduler to automate a process which involves sending passwords to other applications or to send other sensitive information, then you would want to ensure that only the right people can edit the script and see the secrets.

Macro Scheduler allows you to set a password for a script which must then be used to edit it. The script file itself is scrambled so that if it is edited in any way it wouldn't make any sense.



Simply tick the box and provide a password.

The password must be entered twice to ensure it is entered correctly.

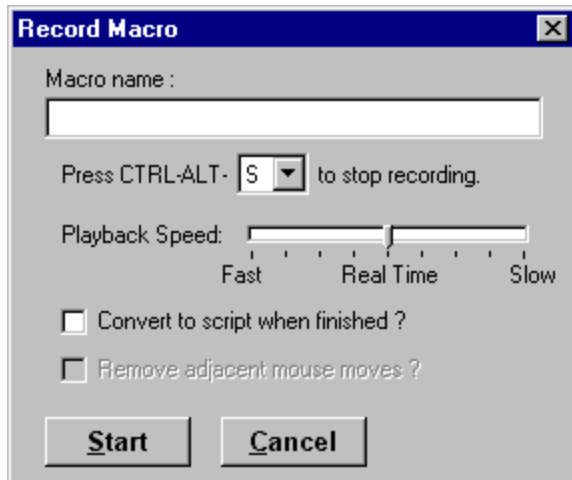
Next time you try to edit the macro you will be asked for the password.

Using the last two options you can specify whether the password should be entered when the macro is run. The first of these two options will ensure that when a macro is started from the [main window](#), [command line](#), or [Macro](#) command, it will only run if the correct password is entered. The second option will further secure the macro for scheduled events, so that a scheduled macro will need the password to be entered before it will start.

To disable encryption on a macro that has been encrypted, edit it and then uncheck this box.

Recording Macros

To record a macro select Run/Record, or press the 'Record' [toolbar](#) button (looks like a video camera) button on the [main window](#). You will be prompted for a name for the macro and recording will commence when Start is pressed.



By default CTRL-ALT-S will terminate the recording. You can select an alternative key from the drop down list box if required. SHIFT-ESC will also stop the recording in the same way that it stops playback.

New in version 4.2 was the ability to have the recorded macro translated into an ordinary script which can then be edited in the usual way. It is possible to choose whether or not to translate the macro when recording is finished. If you choose not to have it translated at this stage you can do it later by attempting to edit the macro. If you select to have the macro translated you can also decide whether or not to remove adjacent mouse moves (see below).

Carry out the tasks you want to be captured and finally press CTRL-ALT-S (or chosen key) or click on 'Stop' to end the recording.

The new macro will appear in the macro list and can be executed by clicking the 'Run Now' button.

To schedule a recorded macro you need to create a new script and use the Macro> command to call the macro from the script. This script can then be scheduled in the usual way. See [Script Commands](#). Alternatively if you translated the macro to a script you will be able to schedule it as usual.

Recorded macros consist of very low level system commands. Each key press and mouse click is treated separately. Consequently when you edit a translated macro you will find that each Send Character/Text> command has only one letter after it which represents the key that was pressed. Furthermore, where you might choose to use a LClick if you were writing a script manually, the program will use one LDown followed by one LUp command, possibly with a wait statement between the two. There will be many wait statements. This is necessary for the macro to reflect as accurately as possible what took place during the record. The only time you will find that the program has simplified things is when interpreting mouse moves.

If you chose to remove adjacent mouse moves, the only mouse move given will always be the last one before a different command. This speeds up playback and keeps the script file much shorter (Moving your mouse slowly across the screen can create hundreds of mouse move messages, but in a script only the last one is needed). However, there might be cases when it is preferable to leave all mouse moves intact, which is why the option is there to allow you not to remove adjacent mousemove commands if

required.

Macros that are not translated may be played back more precisely than those translated, since each recorded Windows message can be replayed just as it was sent.

Playing Scripts and Macros

To play a macro or script without scheduling it use the 'Run Now' [toolbar](#) button on the [main window](#). You can also press CTRL-R, or choose Run/Run Now from the [main menu](#). Another way to start a macro is to right click on it and choose 'Run Now' from the pop-up menu.

After starting a macro the stop button on the tool bar will become enabled. You can press this to stop the macro at any time. CTRL-B or Run/Break will also stop a macro.

To stop macros no matter what program or window is active, press SHIFT-ESC.

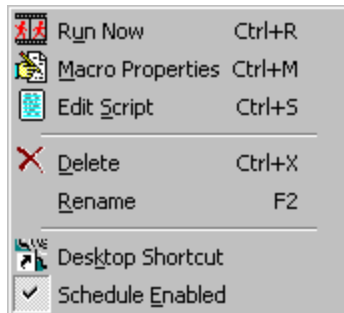
The menu that pops up from the icon in the task bar also has an option called 'Break'. This works like the stop button and allows you to cancel the execution of a script. This option is available even when a script is executed automatically by the scheduler.

You can run recorded macros within ordinary scripts by using the Macro> command. See [Script Commands](#).

Macros can also be assigned to desktop shortcuts or run from the command line. See [Creating Desktop Shortcuts](#) and [Command Line Option](#).

Creating Desktop Shortcuts

You can tell Macro Scheduler to create a shortcut for a macro by selecting the appropriate macro from the main window and then clicking the right mouse button to display a pop up menu. Alternatively you can select Tools/Desktop Shortcut from the [main menu](#), or press the Desktop Shortcut [toolbar](#) button.



Select the second from last option and a shortcut will be placed on your desktop. To run the macro you then only need to double click your desktop icon. Once it is on your desktop you can, if you prefer, move it elsewhere in the usual way using explorer etc.

Command Line Options

It is possible to run macros from the command line using the following syntax:

msched macroname

e.g. to run the Defragment Disk example script you would type:

msched Defragment Disk

This is useful for creating shortcuts and running Macro Scheduler scripts from other programs or from macros created in other applications such as Word or Excel.

However, if you want to create a shortcut, you can get Macro Scheduler do it for you. [Click here for details](#).

When running a Macro from the command line in this way, you can also pass parameter values into the script :

msched Example Script /filename=testfile.txt /path=c:\outpath\

The above example runs a script called 'Example Script' and passes two variable values in filename and path. These variables can be used in the usual way within the script, e.g. :

```
Change Directory>path
ifFileExists>filename,ok
Goto>end
Label>ok
Message>File Exists !!
Label>end
```

It is also possible to run any .scp file by passing the full path and filename to Macro Scheduler like this :

msched c:\scripts\my script.scp

In this instance, the script 'my script' does not have to exist in the Macro Scheduler macro list. If you wish you can associate .scp files with Macro Scheduler so that Macro Scheduler scripts can be run by double clicking on them in explorer.

To switch off system tray support, so that the Macro Scheduler icon does not show in the System Tray add the parameter /NOSYSTRAY at the end of the command line :

msched /NOSYSTRAY

License Agreement

Macro Scheduler and MacroScript are Copyright © 1997-2002, MJT Net Ltd

All rights for this software are reserved by MJT Net Ltd.

You are not allowed to modify or reverse engineer the contents of the program file.

You can use the evaluation version of Macro Scheduler free of charge only for 30 days to evaluate the product. After that, you must purchase the full version by paying the license fee, or stop using it.

This software cannot be resold, leased or rented, including, but not limited to, distributing Macro Scheduler as part of commercial products, or in support of commercial services, without expressed written permission from MJT Net Ltd. Reseller agreements are available from MJT Net Ltd.

There is no warranty or claim of fitness or reliability. The program is distributed AS IS, and MJT Net Ltd shall not be held liable for any loss of data, down time, loss of revenue or any other direct or indirect damage or claims caused by this software.

Copyright © 1997-2002 MJT Net Ltd, all rights reserved

Registration

Why Register ?

The [License Agreement](#) specifies that you can use Macro Scheduler freely for evaluation purposes only for 30 days. After that time you must purchase a fully licensed copy.

As well as legalising your copy of Macro Scheduler, registering will bring you the following benefits :

- No reminder notices on startup.
- No time limits
- Technical Support.
- Discounts on Upgrades and New Versions.
- Notices about major upgrades.
- Access to the Macro Scheduler email discussion list, and usergroup website.
- Boxer 99 discount. MJT Net Ltd has arranged a special discount for its customers on Boxer 99, a powerful, full-featured text editor with Macro Scheduler syntax highlighting and templates.

You also have the choice between purchasing a full version of Macro Scheduler, or the VBScript edition which includes support for Microsoft VBScript.

[Pricing and ordering information for Australia and New Zealand](#)

[Pricing and ordering information for UK, USA and rest of world](#)

Support

If you are a registered user you can get support by emailing us at :

support@mjtnet.com

(Registered users are given priority)

Please also send bug reports, comments and suggestions to this address.

For hints and tips have a look at the Scripts & Tips page at :

<http://www.mjtnet.com/scripts.htm>

Keep an eye on the MJT Net Ltd web site for new product announcements and information:

<http://www.mjtnet.com/>

Hot Keys

Each script can be assigned a hot key to allow the script to be launched from a keyboard shortcut.

To assign a hot key select the tab marked 'HotKey' from the script settings window and select the appropriate keys from the drop down lists.

Press OK to save the settings. No matter what program you are working with, as long as Macro Scheduler is running, the chosen key combination will now launch your macro.

Add

Add>Value,Number

Adds Number to Value.

Interpreted as $\text{Value} = \text{Value} + \text{Number}$

Value must be a variable containing either a numeric or date value. Number can be either a literal number or a variable containing a numeric value.

For date values this function will add the number of days, represented in Number to the given date value.

See also : [Sub](#)

Example

```
Let>Counter=5
```

```
Add>Counter,2
```

i.e. $\text{Counter} = \text{Counter} + 2$

In this example the numeric variable, Counter, is given a new value of 7.

CapsOff

Switches caps lock off. If Caps lock is already off, no action is taken. If Caps lock is on, it is switched off.

Abbreviation : [COF](#)

See also : [CapsOn](#)

CapsOn

Switches caps lock on. If Caps lock is already on, no action is taken. If Caps lock is off, it is switched on.

Abbreviation : [CAP](#)

See also : [CapsOff](#)

Change Directory

Change Directory>path

Changes the current directory to the directory specified in path. path can be a literal string or a variable.

Abbreviation : [Cha](#)

See also : [CreateDir](#)

Example

[Change Directory](#)>c:\program files\my directory\

ConCat

ConCat>string1,string2

Concatenates string1 with string2. String1 must be a variable containing a string. String2 can be a literal string or a variable. The result is that string1 has string2 appended to it.

Abbreviation : **Con**

Example

```
Let>path=c:\temp\  
ConCat>path,myfile.txt
```

In this example path becomes 'c:\temp\myfile.txt'

CopyFile

CopyFile>sourcefile,destinationfile

Copies the file (or files), sourcefile, to destinationfile

sourcefile, and destinationfile may be variables. Wildcards can be used.

By default if destinationfile already exists it will not be overwritten and the new file will be renamed appropriately. It is possible to change the behaviour of the CopyFile command to overwrite instead by setting CF_OVERWRITE to 1. The default value of CF_OVERWRITE is 0.

Abbreviation : [Cop](#)

See also : [MoveFile](#), [DeleteFile](#), [IfFileExists](#)

Example

```
CopyFile>c:\temp\myfile.txt,c:\my documents\myfile.old
```

Or with variables:

```
Let>filename=c:\temp\myfile.txt
```

```
Let>newfilename=c:\temp\myfile.new
```

```
CopyFile>filename,newfilename
```

CreateDir

CreateDir>directory

Creates a new directory. The directory may be a full path.

Abbreviation : [Cre](#)

See also : [Change Directory](#)

Example

```
CreateDir>c:\my documents\test
```

or

```
Let>dir=c:\my documents\test
```

```
CreateDir>dir
```

EditIniFile

EditIniFile>infile,section,entry,newvalue

Edits a section entry in an ini file. infile can be a full path.

All parameters can be variables containing strings.

Abbreviation : [Edi](#)

See also : [ReadIniFile](#)

Example

[EditIniFile](#)>c:\program files\myini.ini,settings,user,fred

ExecuteFile

ExecuteFile>file_to_execute

Executes a file using the application associated with the given file's filetype.

file_to_execute can include a full path.

Abbreviation : [Exe](#)

See also : [Run Program](#)

Example

[ExecuteFile](#)>report.doc

or

[Let](#)>filename=c:\my documents\accounts.xls

[ExecuteFile](#)>filename

GetDate

GetDate>result

Returns the current date in the specified variable. The format of the date depends on the regional settings of the system.

Abbreviation : [GDT](#)

See also : [GetTime](#), [Year](#), [Month](#), [Day](#), [FileDate](#)

Example

```
GetDate>date  
Let>msg=The Date Is :  
ConCat>msg,date  
Message>msg
```

GetTime

GetTime>result

Returns the current time in the specified variable. The format of the time depends on the regional settings of the system.

Abbreviation : [GTM](#)

See also : [GetDate](#), [Year](#), [Month](#), [Day](#), [FileDate](#)

Example

```
GetTime>time  
Let>msg=The Time Is :  
ConCat>msg,time  
Message>msg
```


Goto

Goto>Label_Name

Causes execution to continue at the specified label, missing any commands in between. If the label does not exist an error message will be displayed.

Label_Name can be or include a variable name.

Goto in conjunction with Label, can be used to create infinite loops. Use the If.. commands to cause conditional branching. To break out of infinite loops press Stop, or choose the Break option from the taskbar pop up menu.

See also : [Label](#)

Example

```
Label>Start
..
..
Goto>SecondBit
..
..
Label>SecondBit
..
```

If

If>expression,true_label_name[,false_label_name]

Causes execution to continue at the specified label, if expression is true, missing any commands in between. If the label does not exist an error message will be displayed. If a second false label is specified, execution will jump to that label if the expression resolves false.

The following operators can be used in the conditional statement :

- = Equals
- > Greater than
- < Less than
- <> Not Equal

Values in the expression can be numeric or string values, or variables containing such values.

See also : [Label](#), [Goto](#), [IfFileChanged](#), [IfFileExists](#), [IfWindowOpen](#),

Example

```
Label>Start
..
..
..
If>a<b,Start
```

IfFileChanged

IfFileChanged>filename,range,label_name

Causes execution to jump to the specified label if the given file's date is in the range of days specified.

Abbreviation : [IFC](#)

See also : [Label](#), [Goto](#), [IfWindowOpen](#), [IfFileExists](#), [If](#)

Example

To see if test.txt is less than 30 days old :

```
IfFileChanged>test.txt,<30,end
```

```
..
```

```
..
```

```
Label>end
```

IfFileExists

IfFileExists>filename,label_name

Branches to the specified label if the file specified in filename exists.

Abbreviation : [IFE](#)

See also : [Label](#), [Goto](#), [IfWindowOpen](#), [IfFileChanged](#), [If](#)

Example

```
IfFileExists>myfile.txt,end
```

```
..
```

```
..
```

```
Label>end
```

IfWindowOpen

IfWindowOpen>window_title,label_name

Checks to see if the specified window is open. If so, it causes the script to continue from the specified label without running any other lines of code in between.

The window_title may contain the * symbol at the end to indicate a wildcard.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will first attempt to find the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and selects the first one it finds whose title contains the entered text. This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

It is possible to limit the type of windows this command effects using the WF_TYPE variable:

Let>WF_TYPE=0 - No Child Windows

Let>WF_TYPE=1 - ALL Windows (Default)

Let>WF_TYPE=2 - Visible Windows Only

Abbreviation : [IfW](#)

See also : [Label](#), [Goto](#), [IfFileExists](#), [IfFileChanged](#), [If](#)

Example

IfWindowOpen>Notepad - [Untitled],donotepad

..

..

Label>donotepad

or, with a wildcard :

IfWindowOpen>notepad*,donotepad

..

..

Label>donotepad

Input

Input>variable,prompt[,default_value]

Displays a dialog box to request information from the user. The dialog box displays the prompt specifies in prompt and accepts input into variable. Optionally, a default value can be specified.

The Input box now also has a file browse button, making it useful for accepting filenames from the user.

If the Input dialog is cancelled (cancel is pressed) Input returns an empty string.

prompt can be a variable, containing the prompt to display.

Abbreviation : [Inp](#)

See also : [Ask](#), [Message](#), [MessageModal](#)

Example

[Input](#)>name,Please enter your name ...

Label

Label>Label_Name

Marks a point in the script to allow execution to be passed to that point by the Goto, and If.. commands.

Goto in conjunction with Label, can be used to create infinite loops. Use the If.. commands to cause conditional branching. To break out of infinite loops press Stop, or choose the Break option from the taskbar pop up menu.

See also : [Goto](#), [If](#), [IfWindowOpen](#), [IfFileExists](#), [IfFileChanged](#)

Example

```
Label>Start
..
..
Goto>SecondBit
..
..
Label>SecondBit
..
```

LClick

Simulates a left button mouse click at the current point on the screen.

The following commands will produce the same result :

LDown

LUp

Abbreviation : [LCI](#)

See also : [LDown](#), [LUp](#), [LDbIClick](#), [RClick](#), [RDown](#), [RUp](#), [RDbIClick](#), [MDown](#), [MUp](#), [MDbIClick](#), [MClick](#), [MouseMove](#), [MouseMoveRel](#)

LDblClick

Simulates a left mouse button double click at the current point on the screen.

The following will achieve the same result :

LClick
LClick

or

LDown
LUp
LDown
LUp

Abbreviation : [LDb](#)

See also : [LDown](#), [LUp](#), [LClick](#), [RClick](#), [RDown](#), [RUp](#), [RDbClick](#), [MDown](#), [MUp](#), [MDblClick](#), [MClick](#), [MouseMove](#), [MouseMoveRel](#)

LDown

Simulates a press of the left mouse button. This is like pressing the mouse button down but not releasing it. It is half of a click.

Issuing this command and then using MouseMove would implement dragging. Use LUp to complete the operation.

Abbreviation : [LDo](#)

See also : [LClick](#), [LUp](#), [LDbClick](#), [RClick](#), [RDown](#), [RUp](#), [RDbClick](#), [MDown](#), [MUp](#), [MDbClick](#), [MClick](#), [MouseMove](#), [MouseMoveRel](#)

Let

Let>variable_name=value

Let is used to assign a value to a variable.

Where parameters are passed to commands, variables can also be passed. Variables can also be embedded within a parameter by enclosing the variable name within % symbols.

Let can also be used to perform basic calculations, and to concatenate strings. NB. If you want to assign to a variable a string that already contains a + sign, add an extra + sign to avoid the two strings being concatenated.

Examples

```
Let>name=freddy
```

```
Let>a=5
```

```
Let>path=c:\Program Files\
```

```
Run Program>%path%myapp.exe
```

```
Let>k=k+1
```

```
Let>A=60-4
```

```
Let>A=5*3
```

```
Let>F=75/32
```

```
Let>Name=John+ Smith
```

Name would now equal John Smith

```
Let>Name=John++Smith
```

Name would equal 'John+Smith'

LUp

Releases the left mouse button. It is the latter half of a click.

See LDown.

See also : [LDown](#), [LClick](#), [LDbClick](#), [RClick](#), [RDown](#), [RUp](#), [RDbClick](#), [MDown](#), [MUp](#), [MDbClick](#), [MClick](#), [MouseMove](#), [MouseMoveRel](#)

Macro

Macro>macro_name [/variable=value|variable [/variable=value|variable] ...]

Executes another macro. The specified macro must exist in the current installation of Macro Scheduler. It must be a valid macro that appears in the macro list on Macro Scheduler's main form.

To pass values to the macro specify each one after a / character. The variable name given should exist in the script to be run. The value to assign to that variable is specified after the = character.

Abbreviation : [Mac](#)

Examples

[Macro](#)>Defragment Disk

[Macro](#)>MyMoveFile /source=c:\temp\myfile.bat /destination=c:\temp\myfile.bak

Message

Message>message_text

Displays a message box containing the text specified in message_text. In order that execution of the script can continue, these message boxes are not modal. This means they can be used to display information even if the script is not being run interactively. For modal message boxes, use [MessageModal](#).

To make a message box stay on top, set the variable MSG_STAYONTOP to 1. Likewise, setting MSG_CENTERED to 1 will ensure that the message box is always shown centrally on the screen.

Abbreviation : [MSG](#)

See also : [MessageModal](#), [Input](#), [Ask](#)

Example

```
Message>Hello World!
```

or with variables ..

```
Let>mymsg=Hello World!
```

```
Message>mymsg
```

To force a new line in a message box, use the CRLF system variable :

```
Message>Hello World %CRLF% %CRLF%End Message.
```

This would display :

Hello World

End Message.

To make the message box stay on top (above all other windows) :

```
Let>MSG_STAYONTOP=1
```

```
Message>Hello World
```

MidStr

MidStr>string,start,length,result

Returns a substring of specified length from a given position in a string. result is a variable in which to store the returned string. Any parameter can be a variable containing the appropriate values.

Abbreviation : [Mid](#)

See also : [Position](#), [ConCat](#), [Length](#)

Example

In the following example, the variable somevalue becomes equal to 'Happy' :

```
MidStr>Happy Birthday,1,5,somevalue
```

```
Message>somevalue
```

MouseMove

MouseMove>X,Y

Moves the mouse cursor to screen position X,Y. 0,0 is the upper left hand corner of the screen. The maximum limits are determined by your screen resolution settings. Variables containing the coordinates can be used in the command.

To help determine a particular point on the screen, the macro window has a cursor monitor which updates as you move the cursor. See [Creating Scripts](#).

Abbreviation : [Mou](#)

See also : [MouseMoveRel](#), [LClick](#), [LDown](#), [LUp](#), [LDbClick](#), [RClick](#), [RDown](#), [RUp](#), [RDbClick](#)

Example

If position 504,252 is within the area taken up by a button, the following script would cause that button to be clicked :

[MouseMove](#)>504,252

[LClick](#)

MoveFile

MoveFile>sourcefile,destinationfile

Moves the file (or files), sourcefile, to the file (or files) destinationfile.

sourcefile, and destination_path may be variables. Wildcards can be used.

Abbreviation : [Mov](#)

See also : [CopyFile](#), [DeleteFile](#), [IfFileExists](#)

Example

```
MoveFile>c:\temp\myfile.txt,c:\temp\myfile.bak
```

Or with variables:

```
Let>filename=c:\temp\myfile.txt
```

```
Let>newfilename=c:\temp\myfile.bak
```

```
MoveFile>filename,newfilename
```

NumOff

Switches Num lock off. If Num lock is already off, no action is taken. If Num lock is on, it is switched off.

Abbreviation : [NOF](#)

See also : [NumOn](#)

NumOn

Switches Num lock on. If Num lock is already on, no action is taken. If Num lock is off, it is switched on.

Abbreviation : [NON](#)

See also : [NumOff](#)

Position

Position>substring,string,start,result

Returns the starting position of a substring in a string. The search commences at the position specified in start. If found the starting position of the substring is returned in the result variable. If no match is found this value will be zero.

Abbreviation : [Pos](#)

See also : [Length](#), [MidStr](#), [ConCat](#)

Example

In this example, StartPos will contain the value 4 :

```
Position>Smith,Mr Smith,1,StartPos
```

Press ...

All commands starting with Press, facilitate the sending of non-character keys. The following is a complete list of all commands, with explanations where necessary :

Press Backspace

Press Tab

Press Enter

Press Esc

Press F1

Press F2

Press F3

Press F4

Press F5

Press F6

Press F7

Press F8

Press F9

Press F10

Press F11

Press F12

Press F13

Press F14

Press F15

Press F16

Press F17

Press F18

Press F19

Press F20

Press F21

Press F22

Press F23

Press F24

Press Home

Press End

Press Up

Press Down

Press Left

Press Right

Press Page Up

Press Page Down

Press Ins

Press Del

Press Shift

Release Shift

Press CTRL

Release CTRL

Press ALT

Release ALT

Press ALTGR

Release ALTGR

Press CAPS

has the effect of toggling caps lock.

Press Num Lock

Press Scroll Lock

Press NP0

0 on Number Pad

Press NP1	etc
Press NP2	
Press NP3	
Press NP4	
Press NP5	
Press NP6	
Press NP7	
Press NP8	
Press NP9	
Press NP Add	Num pad operator keys
Press NP Subtract	etc
Press NP Multiply	
Press NP Divide	
Press NP Decimal	
Press NP Enter	
Press LWinKey	}
Press RWinKey	} Windows 95 Keys
Press MenuKey	}

RClick

Simulates a right button mouse click at the current point on the screen.

The following commands will produce the same result :

RDown

RUp

Abbreviation : [RCI](#)

See also : [LDown](#), [LUp](#), [LDbIClick](#), [LClick](#), [RDown](#), [RUp](#), [RDbIClick](#), [MDown](#), [MUp](#), [MDbIClick](#), [MClick](#), [MouseMove](#), [MouseMoveRel](#)

RDbIClick

Simulates a right mouse button double click at the current point on the screen.

The following will achieve the same result :

RClick
RClick

or

RDown
RLUp
RDown
RUp

Abbreviation : [RDb](#)

See also : [LClick](#), [LDown](#), [LUp](#), [LDbIClick](#), [RClick](#), [RDown](#), [RUp](#), [MDown](#), [MUp](#), [MDbIClick](#), [MClick](#), [MouseMove](#), [MouseMoveRel](#)

RDown

Simulates a press of the right mouse button. This is like pressing the mouse button down but not releasing it. It is half of a click.

Abbreviation : [RDo](#)

See also : [LClick](#), [LDown](#), [LUp](#), [LDbClick](#), [RClick](#), [RDbClick](#), [RUp](#), [MDown](#), [MUp](#), [MDbClick](#), [MClick](#), [MouseMove](#), [MouseMoveRel](#)

ReadIniFile

ReadIniFile>infile,section,entry,result

Reads a value from an entry in an ini file and places the result in a variable.

All parameters can be variables containing strings.

Abbreviation : [Rea](#)

See also : [EditIniFile](#)

Example

This example reads a username from an ini file and displays it in a message :

```
ReadIniFile>c:\program files\myini.ini,settings,user,username
Let>msg=The Username is
ConCat>msg,username
Message>msg
```

Release ...

Some keys that can be pressed must also be released. This facilitates holding down a key while another is pressed, such as with the ALT key for instance.

For example, to exit a program you would press ALT and F together to activate the File menu, followed by the X key to select the Exit option. To simulate this in a script you would Press ALT, then send the text FX and finally Release ALT. This would appear in the script window as :

```
Press ALT  
Send Character/Text>FX  
Release ALT
```

The following release key commands exist :

```
Release ALT  
Release ALTGR  
Release CTRL  
Release Shift
```

Remark

Remark>Some Comment

The remark statement is ignored by the interpreter. It exists simply to allow comments to be placed in the code. In fact, any text that is not a recognised command can be used for this purpose.

Run Program

Run Program>path

Executes a specified file. Files that can be executed are .exe, .bat, and .com files.

By setting the RP_WINDOWMODE variable programs can be executed minimized, maximised, hidden or normal. RP_WINDOWMODE can be one of the following :

- 0: Hidden
- 1: Normal (default)
- 2: Minimized
- 3: Maximized

If you set RP_WINDOWMODE it is used by all subsequent Run Program commands, so remember to set it back if you don't want the same window mode to be used each time.

By default a message is displayed if this command encounters an error when running the specified program. Error messages from this command can be suppressed by setting RP_DISPLAYERROR to 0.

The result of the Run Program command is stored in the variable RP_RESULT. A value greater than 31 indicates success. The following values represent errors :

- 0 - The system was out of memory, or the executable file was corrupt, or relocations were invalid.
- 2 - The file was not found.
- 3 - The path was not found.
- 5 - An attempt was made to dynamically link to a task, or there was a sharing or network protection error.
- 6 - The library required separate data segments for each task.
- 10 - The Windows version was incorrect.
- 11 - The executable file was invalid. It was either not a Windows-based application or there was an error in the .EXE image.
- 12 - The application was designed for OS/2.
- 13 - The application was designed for MS-DOS 4.0.
- 14 - The type of executable file was unknown.
- 15 - An attempt was made to load a real-mode application (developed for an earlier version of Windows).
- 16 - An attempt was made to load a second instance of an executable file containing multiple data segments that were not marked "read-only."
- 17 - Attempt in large-frame EMS mode to load a second instance of an application that links to certain non-shareable DLLs already in use.
- 18 - Attempt in real mode to load an application marked for protected mode only.

Abbreviation : [Run](#)

See also : [ExecuteFile](#)

Example

To open Notepad :

```
Run Program>notepad.exe
```

A path may be specified if necessary :

```
Run Program>c:\my programs\eudora\eudora.exe
```

To start notepad minimized :

```
Let>RP_WINDOWMODE=2
```

```
Run Program>notepad.exe
```

RUp

Releases the right mouse button. It is the latter half of a click.

See RDown.

See also : [LClick](#), [LDown](#), [LUp](#), [LDbIClick](#), [RClick](#), [RDbIClick](#), [RDown](#), [MDown](#), [MUp](#), [MDbIClick](#), [MClick](#), [MouseMove](#), [MouseMoveRel](#)

ScrollOff

Switches Scroll lock off. If Scroll lock is already off, no action is taken. If Scroll lock is on, it is switched off.

Abbreviation : [SOF](#)

See also : [ScrollOn](#)

ScrollOn

Switches Scroll lock on. If Scroll lock is already on, no action is taken. If Scroll lock is off, it is switched on.

Abbreviation : [SON](#)

See also : [ScrollOff](#)

Send Character/Text (Send)

Send Character/Text>text_to_send

This command sends the specified text to the window that currently has the focus. See [SetFocus](#).

It is possible to slow down the speed at which each character in the string is sent by using the SK_DELAY variable. Set this to the number of milliseconds to pause between each individual character. e.g.:
Let>SK_DELAY=10

Abbreviation : [Sen \(Send\)](#)

See Also : [Ascii](#)

Examples

[Send Character/Text](#)>Here Is Some Text ...

The following example simulates pressing ALT-FX, a standard key combination for closing a program :

[Press ALT](#)

[Send Character/Text](#)>fx

[Release ALT](#)

Variables can be used :

[Let](#)>SomeText=Hello World

[Send Character/Text](#)>SomeText

SetFocus

SetFocus>window_title

Sets focus to the specified window. The window_title may contain the * symbol at the end to indicate a wildcard.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to setfocus to the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and sets focus to the first one it finds whose title contains the entered text. This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

To set focus to child windows, such as MDI children, first use SetFocus to focus the application, and then issue a second SetFocus for the child window. In most cases it is necessary to add the * symbol for child windows.

It is possible to limit the type of windows this command effects using the WF_TYPE variable:

- Let>WF_TYPE=0 - No Child Windows
- Let>WF_TYPE=1 - ALL Windows (Default)
- Let>WF_TYPE=2 - Visible Windows Only

Abbreviation : [Set](#)

Example

SetFocus>notepad*

Child Window Example:

SetFocus>Opera*

SetFocus>google*

Sub

Sub>Value,Number

Subtracts a number from a value.

Interpreted as $\text{Value} = \text{Value} - \text{Number}$

Value must be a variable containing a numeric or date value. Number can be either a literal number or a variable containing a numeric value.

For date values this function will subtract the number of days, represented in Number from the given date value.

See also : [Add](#), [Let](#)

Example

```
Let>Counter=5  
Sub>Counter,2
```

i.e. $\text{Counter} = \text{Counter} - 2$

In this example the numeric variable, Counter, is given a new value of 3.

```
Let>Counter=Counter-1
```

Will now do the same thing.

Wait

Wait>seconds_to_wait

This command makes Macro Scheduler pause for the specified number of seconds.

See also : [WaitWindowOpen](#), [WaitWindowClosed](#), [WaitPixelColor](#), [WaitCursorChanged](#)

Example

To wait 5 seconds :

```
Wait>5
```

This will work too :

```
Let>WaitTime=5
```

```
Wait>WaitTime
```

WaitWindowClosed

WaitWindowClosed>window_title

Waits for a specified window to close. Execution of the script will not continue until the window with the specified title text is no longer present or a specified timeout value is exceeded. The window title may contain the * symbol at the end to indicate a wildcard.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will first attempt to find the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and selects the first one it finds whose title contains the entered text. This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

The system variable WW_TIMEOUT can be used to set the number of seconds after which this command should timeout. If set to zero (the default) the timeout will not occur and the command will continue indefinitely. If WW_TIMEOUT is used, WW_RESULT will indicate whether or not the command ended successfully. If it timed out WW_RESULT will be set to FALSE. If the window it was waiting for appeared within the timeout setting, the WW_RESULT value will be set to TRUE.

It is possible to limit the type of windows this command effects using the WF_TYPE variable:

- Let>WF_TYPE=0 - No Child Windows
- Let>WF_TYPE=1 - ALL Windows (Default)
- Let>WF_TYPE=2 - Visible Windows Only

Abbreviation : [WWC](#)

See also : [Wait](#), [WaitWindowOpen](#)

Examples

[WaitWindowClosed](#)>Progress

[Let](#)>WW_TIMEOUT=10

[WaitWindowClosed](#)>Progress

[If](#)>WW_RESULT=FALSE,Endit

..

..

[Label](#)>Endit

WaitWindowOpen

WaitWindowOpen>window_title

Waits for a specified window to open/appear. Execution of the script will not continue until a window with the specified title text appears or a specified timeout value is exceeded. The window title may contain the * symbol at the end to indicate a wildcard.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will first attempt to find the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and selects the first one it finds whose title contains the entered text. This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

The system variable WW_TIMEOUT can be used to set the number of seconds after which this command should timeout. If set to zero (the default) the timeout will not occur and the command will continue indefinitely. If WW_TIMEOUT is used, WW_RESULT will indicate whether or not the command ended successfully. If it timed out WW_RESULT will be set to FALSE. If the window it was waiting for appeared within the timeout setting, the WW_RESULT value will be set to TRUE.

It is possible to limit the type of windows this command effects using the WF_TYPE variable:

```
Let>WF_TYPE=0 - No Child Windows
Let>WF_TYPE=1 - ALL Windows (Default)
Let>WF_TYPE=2 - Visible Windows Only
```

Abbreviation : [WWO](#)

See also : [Wait](#), [WaitWindowClosed](#)

Examples

```
Run Program>c:\program files\msoffice\winword.exe
WaitWindowOpen>microsoft word*
```

```
Let>WW_TIMEOUT=30
WaitWindowOpen>microsoft word*
If>WW_RESULT=FALSE,Endit
..
..
Label>Endit
Message>Error starting Word!
```

MouseMoveRel

MouseMoveRel>X,Y

Moves the mouse cursor to the position X,Y relative to the upper left corner of the window currently in focus. 0,0 will be the upper left hand corner of the active window. Variables containing the coordinates can be used in the command.

The advantage of this command over the Move command, is that this will not fail when the window changes its position or resizes, or if the screen resolution is changed.

To help determine a particular point on the screen, the macro window has a cursor monitor which updates as you move the cursor. See [Creating Scripts](#). To determine a point relative to a specific window, try moving that window so that its upper left corner is in the upper left corner of the screen (position 0,0). Maximising the app would achieve the same result. Then you can use the values displayed in the cursor position monitor of Macro Scheduler.

Abbreviation : [MMR](#)

See also : [MouseMove](#), [LClick](#), [LDown](#), [LUp](#), [LDoubleClick](#), [RClick](#), [RDown](#), [RUp](#), [RDoubleClick](#)

Example

If position 40,50 is a point relative to the current window, on a button, the following script would cause that button to be clicked :

```
MouseMoveRel>40,50  
LClick
```

DeleteFile

DeleteFile>filename

Deletes the file/files in filename.

filename can be a variable. Wildcards can be used.

Abbreviation : [Del](#)

See also : [CopyFile](#), [MoveFile](#), [IfFileExists](#)

Example

[DeleteFile](#)>c:\temp*.*

GetClipBoard

GetClipBoard>result_variable

Retrieves the contents of the clipboard as text and places it in the specified variable.

Abbreviation : [GCB](#)

See also : [PutClipBoard](#)

Example

[GetClipBoard](#)>WhatsInTheClipBoard

PutClipboard

PutClipboard>SomeValue

Places the specified text onto the clipboard. A variable may be used, or a literal value.

Abbreviation : [Put](#)

See also : [GetClipboard](#)

Example

```
PutClipboard>Hello World !!
```

GetWindowPos

GetWindowPos>window_title,X,Y

Locates the window specified in window_title and retrieves its upper left screen coordinates. X and Y are variables in which to store the coordinates. window_title may be a variable or literal.

The window_title may contain the * symbol at the end to indicate a wildcard.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to setfocus to the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and sets focus to the first one it finds whose title contains the entered text. This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

Abbreviation : [GWP](#)

See also : [GetActiveWindow](#), [CloseWindow](#)

Example

GetWindowPos>My Computer,X,Y

The following example achieves the same result as the MouseMoveRel command, moving to the point 10,10 relative to Notepad :

GetWindowPos>notepad*,npX,npY

Add>npX,10

Add>npY,10

MouseMove>npX,npY

FindWindowWithText

FindWindowWithText>text_to_find,setfocus_flag,result_variable,check_edits_flag

This function attempts to locate a window containing somewhere within it the text specified in text_to_find. setfocus_flag can be set to 1 or 0. If set to 1, the function will activate the window it finds containing the specified text. If a window is found, result_variable will contain the found window's title text. If no window is found it will contain "NOT FOUND".

The check_edits_flag can be set to 1 or 0. When set to 1, Macro Scheduler will attempt to compare text contained within edit boxes, including single and multi line edit boxes.

NB. Not all text can be detected successfully. Text on buttons, labels belonging to other controls, window titles, and pre-set text in edit boxes can usually be detected successfully. It may not be possible to find stand-alone label text or text entered by the user. Also, text in list boxes and combo boxes etc cannot be detected. To find text entered by the user it may be necessary to set the check_edits_flag on.

This function is useful when an application has two windows with the same name, and allows the correct one to be located and focused.

In some Windows 95 installations, when the check_edits_flag is switched on, this function has caused a general protection fault in module user.exe. However, it has been found to work with no problems on Windows NT.

Abbreviation : [Fin](#)

Example

FindWindowWithText>Continue,1,windowname,0

Day

Day>result

Returns the current day number of the month in the specified variable.

See also : [Month](#), [Year](#), [GetDate](#), [GetTime](#)

Example

Day>the_day

Month>the_month

Year>the_year

Message>The date is : %the_day% - %the_month% - %the_year%

Year

Year>result

Returns the current year in the specified variable.

See also : [Month](#), [Day](#), [GetDate](#), [GetTime](#)

Example

```
Day>the_day  
Month>the_month  
Year>the_year  
Message>The date is : %the_day% - %the_month% - %the_year%
```

Month

Month>result

Returns the current month number in the specified variable.

Abbreviation : [Mon](#)

See also : [Day](#), [Year](#), [GetDate](#), [GetTime](#)

Example

[Day](#)>the_day

[Month](#)>the_month

[Year](#)>the_year

[Message](#)>The date is : %the_day% - %the_month% - %the_year%

CloseWindow

CloseWindow>window_title

Closes the specified window. The window_title may contain the * symbol at the end to indicate a wildcard.

If the specified window is the main window of an application, then that application will begin to close down. Any processing that the application does on exit will be carried out as usual.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to close the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and selects the first one it finds whose title contains the entered text. This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

It is possible to limit the type of windows this command effects using the WF_TYPE variable:

Let>WF_TYPE=0 - No Child Windows

Let>WF_TYPE=1 - ALL Windows (Default)

Let>WF_TYPE=2 - Visible Windows Only

Abbreviation : [Clo](#)

See also : [GetWindowPos](#), [GetActiveWindow](#)

Example

[CloseWindow](#)>notepad*

GetActiveWindow

GetActiveWindow>window_title,X,Y

Retrieves information about the current active window. The window title, and top left coordinates of the active window are stored in window_title, X and Y.

Abbreviation : [GAW](#)

See also : [GetWindowPos](#), [CloseWindow](#)

MessageModal

MessageModal>message_text

Displays a message box containing the text specified in message_text. With this command the Message box is modal. This means that the script will not continue until OK is pressed. For non modal message boxes use [Message](#).

To make a message box stay on top, set the variable MSG_STAYONTOP to 1. Likewise, setting MSG_CENTERED to 1 will ensure that the message box is always shown centrally on the screen.

Abbreviation : [MDL](#)

See also : [Message](#), [Input](#), [Ask](#)

Example

```
MessageModal>Hello World!
```

or with variables ..

```
Let>mymsg=Hello World!
```

```
MessageModal>mymsg
```

FTPGetFile

FTPGetFile>Server,Username,Password,port,Local_File,Host_File,Mode

This command will connect to the specified FTP server and retrieve a file.

The file retrieved from the server is specified in Host_File, which must contain the full remote path. The file is saved on the local machine under the name specified in Local_File.

Mode can be either A or I, where A represents ASCII transfer mode and I represents binary.

When the FTP commands are active a small floating window is displayed at the top left of the screen showing the status of the FTP session. To stop the status window appearing set FTP_STATUS to 0 before issuing an FTP command. Set it back to 1 to show the window. The default value for FTP_STATUS is 1.

If passive mode is required, set FTP_PASSIVE to 1. Set to zero to switch passive mode off.

The result of the FTP operation is stored in FTP_RESULT.

If you are connecting via a proxy server, enter the name or IP address of the proxy server in 'Server', and for 'Username' specify username@remote_server.

By default the timeout for the FTP commands is set to 15 seconds. To change this set the FTP_TIMEOUT variable.

Abbreviation : [FGF](#)

See also : [FTPPutFile](#), [FTPGetDirList](#), [FTPDelFile](#), [FTPRenameFile](#)

Example

FTPGetFile>ftp.domain.com,anonymous,user@domain.com,21,c:\temp\myfile.txt,/pub/readme.txt,A

FTPPutFile

FTPPutFile>Server,Username,Password,port,Local_File,Host_File,Mode

This command will connect to the specified FTP server and upload a file.

The file to be uploaded is specified in Local_File, which can contain a full path. The file saved on the remote host is specified in Host_File, which must contain the full remote path.

Mode can be either A or I, where A represents ASCII transfer mode and I represents binary.

When the FTP commands are active a small floating window is displayed at the top left of the screen showing the status of the FTP session. To stop the status window appearing set FTP_STATUS to 0 before issuing an FTP command. Set it back to 1 to show the window. The default value for FTP_STATUS is 1.

If passive mode is required, set FTP_PASSIVE to 1. Set to zero to switch passive mode off.

The result of the FTP operation is stored in FTP_RESULT.

If you are connecting via a proxy server, enter the name or IP address of the proxy server in 'Server', and for 'Username' specify username@remote_server.

By default the timeout for the FTP commands is set to 15 seconds. To change this set the FTP_TIMEOUT variable.

Abbreviation : [FPF](#)

See also [FTPGetFile](#), [FTPGetDirList](#), [FTPDelFile](#), [FTPRenameFile](#)

Example

FTPPutFile>ftp.domain.com,anonymous,user@domain.com,21,c:\temp\readme.txt,/pub/readme.txt,A

PlayWav

PlayWav>wav_file

Plays a .WAV sound file.

Abbreviation : [Pla](#)

Example

[PlayWav](#)>c:\windows\media\chimes.wav

CountFiles

CountFiles>file_spec,result,subdir_flag

Returns the number of files matching file_spec in the result variable.

file_spec can include a directory path, and should include a mask. e.g. c:\temp*.*

To recurse sub directories set subdir_flag to 1.

Abbreviation : [Cou](#)

Examples

```
CountFiles>c:\my documents\*.doc,DocCount,0
Message>Number of .doc files : %DocCount%
```

The following example will count all the files anywhere on the c: drive.

```
CountFiles>c:\*.*;TotalFiles,1
Message>Total Number Of Files On PC : %TotalFiles%
```

FileSize

FileSize>filename,result

Returns the size of filename in the result variable.

The size is returned in bytes.

filename can include a full path.

Abbreviation : [FSZ](#)

See also : [FileDate](#)

Example

```
FileSize>c:\program files\myfile.exe,MyFileSize
```

FileDate

FileDate>filename,result

Returns the file date of filename in the result variable.

The date returned is in the format YYYYMMDD

filename can include a full path.

Abbreviation : [FDT](#)

See also : [FileSize](#)

Example

```
FileDate>c:\program files\myfile.exe,MyFileDate
```


Length

Length>string,result

Returns the length of the given string

Abbreviation : [Len](#)

See also : [MidStr](#), [Position](#), [ConCat](#)

Example

[Length](#)>Hello World,Len

Ask

Ask>prompt,result_variable

Displays a Yes, No dialog box with the specified prompt. If the user presses 'Yes', result_variable is set to 'YES', else result_variable becomes 'NO'.

The result is returned in upper case.

prompt can be a variable containing the prompt to display.

See also : [Input](#), [Message](#), [MessageModal](#)

Example

Ask>Do you want to continue ?,continue

Using Variables

As well as system variables, which are pre-set, variables can be assigned by script commands which return variables, and by the Let command which allows you to create variables at any time.

The Let command is used to assign a value to a variable like so :

```
Let>Name=Freddy Mercury
```

The first time a variable is assigned a value, that variable is created. Subsequently, it's value is modified.

Variables can then be passed into other functions as one of the parameters. For example, assuming the above Let command has taken place, the following command will display a message box containing the words 'Freddy Mercury' :

```
Message>Name
```

If Name did not exist as a variable, the message box would simply display the word 'Name'.

Commands always check to see if a parameter passed to it is a variable. If a variable with that name exists, the value assigned to that variable is used, otherwise just that literal value is used.

All commands which accept a value can also accept a variable in place of that value.

For example, the following command would send the text 'Freddy Mercury' to the current window :

```
Send Character/Text>Name
```

In some cases a variable must be embedded within a string. This can be achieved using the % operator. In the following example a message is displayed saying 'Hello Freddy Mercury' :

```
Message>Hello %Name%
```

The system variable CRLF can be used to force a new line in a message box. Therefore, a list can be built up like so :

```
Message>1 : first line %CRLF%2 : second line %CRLF%3 : third line
```

It is possible to ask the user for a value, using the Input box :

```
Input>path,Please enter directory to create :
```

This would prompt the user to enter a directory name to create, which would then be assigned to the variable path. We could then use it in the CreateDir command as follows :

```
CreateDir>path
```

The If command can be used to check the value of a variable :

```
If>Age>15,adult
```

```
..
```

```
..
```

```
Label>adult
```

System Variables

OS_VER	Operating System
WIN_DIR	Windows Directory Path
SYS_DIR	Windows System Directory Path

The above variables store their values in upper case.

USER_NAME	Current Username
COMPUTER_NAME	Computer Name

WW_TIMEOUT	Timeout value from WaitWindowOpen / Closed
WW_RESULT	Result from WaitWindowOpen / Closed
WCC_RESULT	Result from WaitCursorChanged
DDE_TIMEOUT	Timeout value from DDERequest
RP_WINDOWMODE	Used to set window mode for Run Program
RP_RESULT	Used to store return code from Run Program
RP_DISPLAYERROR	Allows Run Program error messages to be turned off/on
MSG_STAYONTOP	Used to set message to stay on top in Message / MessageModal
MSG_CENTERED	Used to set message box to center
SK_DELAY	Millisecond delay to pause between sending characters in Send Character/Text
RND_SEED	Allows a seed to be set for the Random command
CF_OVERWRITE	Allows changing CopyFile to overwrite or rename on collision
FTP_STATUS	Used to switch off/on the FTP status window
FTP_PASSIVE	Used to toggle FTP passive mode
FTP_TIMEOUT	Timeout value for FTPGetFile / FTPPutFile / FTPGetDirList
FTP_RESULT	Result of FTP commands.
SENDMAIL_STATUS	Used to switch off/on the SMTPSendMail status window
SMTP_RESULT	Result of SMTPSendMail command.
SMTP_AUTH	Used to set SMTP authentication on or off
SMTP_USERID	Used for SMTP authentication
SMTP_PASSWORD	Used for SMTP authentication
SMTP_RECEIPT	Used to enable SMTP return receipt
SMTP_PORT	Used to optionally set the port of the SMTP server
REG_INTASSTR	Used to set RegistryWriteKey to write integers as strings/integers
WF_TYPE	Used to set the type of window the windowing functions should affect
VBS_TIMEOUT	Used to set the timeout for VBScript code

CR	Carriage Return
LF	Line Feed
CRLF	Carriage Return, Line Feed Combination (to force a new line)

GetCursorPos

GetCursorPos>X,Y

Returns the X and Y coordinates of the current cursor position. X and Y are variables in which to store the coordinates.

Abbreviation : [GCP](#)

See also : [GetWindowPos](#)

Example

[GetCursorPos](#)>X,Y

[Message](#)>Current Position : %X%,%Y%

Sec

Sec>result

Returns the seconds portion of the current time in the specified variable.

See also : [Min](#), [Hour](#), [GetDate](#), [GetTime](#)

Example

Sec>Seconds

Min>Minutes

Hour>Hour

Message>The time is : %Hour% : %Minutes% : %Seconds%

Min

Min>result

Returns the minutes portion of the current time in the specified variable.

See also : [Sec](#), [Hour](#), [GetDate](#), [GetTime](#)

Example

Sec>Seconds

Min>Minutes

Hour>Hour

Message>The time is : %Hour% : %Minutes% : %Seconds%

Hour

Hour>result

Returns the seconds portion of the current time in the specified variable.

See also : [Min](#), [Sec](#), [GetDate](#), [GetTime](#)

Example

Sec>Seconds

Min>Minutes

Hour>Hour

Message>The time is : %Hour% : %Minutes% : %Seconds%

PushButton

PushButton>window_title,button_caption

Attempts to 'click' the specified button of the specified window.

window_title can contain an asterisk (*) as with all other window functions. For buttons that have a hot key associated with them, and represented on the button by an underscored letter, pass a & character before that letter. e.g.: for a button called 'Close', send &Close.

This command works by attempting to send the BM_CLICK message to the button when it finds a button in the specified window with the given caption.

This will only work with objects of the standard 'Button' class. It may not work for all buttons on all windows. For instance, it doesn't work for buttons on html documents in Netscape Navigator 4.05 or Internet Explorer 3.02, as they are not real buttons. For these use the [MouseOver](#) command.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to locate the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and stops at the first one it finds whose title contains the entered text. This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

Abbreviation : [PUS](#)

See also : [MouseOver](#)

Example

```
Run Program>rundll32.exe shell32.dll,Control_RunDLL TimeDate.cpl
```

```
WaitWindowOpen>Date/Time Properties
```

```
...
```

```
PushButton>Date/Time*,OK
```

ReadLn

ReadLn>file_name,line_number,result

Reads the specified line from the given text file. If line_number is past the end of the file, result will contain **##EOF##**. If the file does not exist, result is set to **##NOFILE##**. If an error occurred, result is set to **##ERR##** followed by the error code.

Abbreviation : [RLN](#)

See also : [WriteLn](#)

Example

This example reads each line from the text file and displays it in a Message window.

```
Let>k=1
Label>start
ReadLn>c:\temp\test.txt,k,line
If>line==##EOF##,finish
Message>line
Let>k=k+1
Goto>start
Label>finish
```

WriteLn

WriteLn>file_name,result,line

Writes text to a text file. If the file does not exist it is created and the given line is written to it. If it does exist, the line is written to the end of the file. If the command was successful, result is set to zero. A non zero result means an error occurred.

Abbreviation : [WLN](#)

See also : [ReadLn](#)

Example

[GetDate](#)>date

[WriteLn](#)>c:\temp\test.txt,result,%date% - Starting Macro

DDEPoke

DDEPoke>Server,Topic,Item,Data

Pokes data to the given DDE server. The server's DDE topic and item must be specified.

Abbreviation : [DPK](#)

See also : [DDERequest](#)

Example

[DDEPoke](#)>MyServer,System,Item1,Testing 123

DDERequest

DDERequest>Server,Topic,Item,Result,Timeout

Requests data from a DDE server. The data returned is stored in the Result variable. If the conversation does not complete within the Timeout value specified, Result will contain 'DDE_TIMEDOUT'. The Timeout value is in seconds.

Abbreviation : [DRQ](#)

See also : [DDEPoke](#)

Example

The following example gets the URL and window title from Netscape.

```
DDERequest>Netscape,WWW_GetWindowInfo,0xFFFFFFFF,ret,10
Message>ret
```

The DDERequest command can also be used to open a web page in Netscape :

```
DDERequest>Netscape,WWW_OpenUrl,www.mjtnet.com,ret,0
```

DayOfWeek

DayOfWeek>result

Returns the current week day number, starting with Sunday as day 1, and ending on Saturday with day 7.

Abbreviation : [DOW](#)

See also : [Day](#), [Month](#), [Year](#), [GetDate](#), [GetTime](#)

Example

This example displays the current day as a proper day name. It also shows how to use variables in a Goto command.

```
DayOfWeek>result
```

```
Goto>Day%result%
```

```
Label>Day1
```

```
Let>DayString=Sunday
```

```
Goto>Continue
```

```
Label>Day2
```

```
Let>DayString=Monday
```

```
Goto>Continue
```

```
Label>Day3
```

```
Let>DayString=Tuesday
```

```
Goto>Continue
```

```
Label>Day4
```

```
Let>DayString=Wednesday
```

```
Goto>Continue
```

```
Label>Day5
```

```
Let>DayString=Thursday
```

```
Goto>Continue
```

```
Label>Day6
```

```
Let>DayString=Friday
```

```
Goto>Continue
```

```
Label>Day7
```

```
Let>DayString=Saturday
```

```
Goto>Continue
```

```
Label>Continue
```

```
MessageModal>DayString
```

WaitCursorChanged

WaitCursorChanged>Timeout

This command causes Macro Scheduler to wait until the cursor of the foreground window changes. If it doesn't change within the number of seconds specified in Timeout, the command stops waiting and the variable WCC_RESULT is set to FALSE. WCC_RESULT is TRUE if the command terminated because the foreground window cursor changed within the specified time. If Timeout is set to 0, the command will wait indefinitely.

This command is useful for waiting for applications to become idle. For example, it can be used after initiating some operation in an application that invokes the hourglass cursor, so that you can wait for the application to become idle again.

Abbreviation : [WCC](#)

See also : [Wait](#), [WaitWindowOpen](#), [WaitWindowClosed](#), [WaitPixelColor](#)

Example

```
Change Directory>c:\program files\agent\data
Run Program>"c:\program files\agent\agent.exe"
WaitCursorChanged>500
SetFocus>Agent*
```

Menu Commands

File

- [New Macro](#)
- [Macro Properties](#)
- [Edit Script](#)
- [Hide](#)
- [Exit](#)

Edit

- [Delete](#)
- [Rename](#)
- [Select All](#)

Groups

- [New Group](#)
- [Delete Group](#)
- [Group Properties](#)

View

- [Sort](#)
- [Font](#)
- [Grid Lines](#)
- [Large Buttons](#)
- [Toolbars](#)

Tools

- [View Log File](#)
- [Desktop Shortcut](#)
- [View System Windows](#)
- [Options](#)

Run

- [Run Now](#)
- [Record](#)
- [Break](#)

Help

- [Contents](#)
- [Register](#)
- [About](#)

New Macro

Displays the [Macro Properties](#) window for creating a new macro.

Macro Properties

Displays the [Macro Properties](#) window for the currently selected macro.

Edit Script

This opens the current macro in the [editor](#), bypassing the [properties](#) window.

Hide

Hides the main form (minimizes to the system tray). Not available in NT3.5x

Exit

Closes Macro Scheduler.

Delete

After prompting for confirmation, deletes the selected macro.

Rename

Displays the rename dialog to allow the selected macro to be renamed.

Sort

Presents a further menu containing the names of the macro list columns. Selecting one sorts the macro list by that column.

You can also sort the macro list by clicking on a column header.

Font

Displays the standard font dialog to allow the macro list font to be set.

GridLines

Toggles the grid lines in the macro list on and off.

Toolbars

Allows the toolbars to be toggled on and off.

View Log File

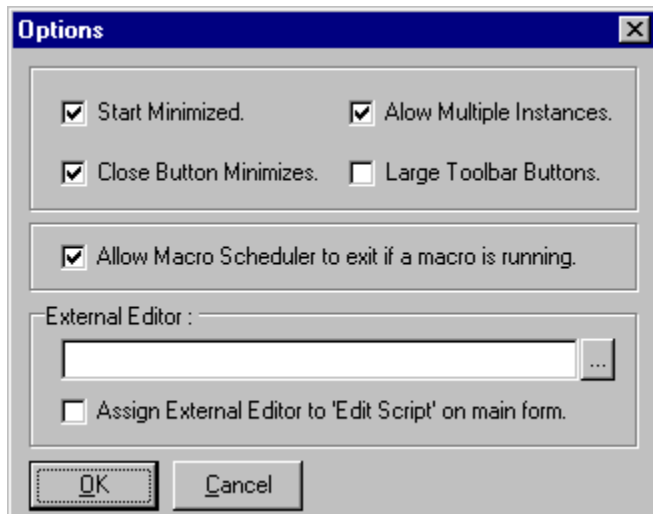
If the selected macro has a log file assigned to it, this option will display that log file in Notepad.

Desktop Shortcut

Creates a desktop shortcut for the selected macro. See [Desktop Shortcuts](#).

Options

Displays the options dialog.



To make Macro Scheduler startup minimized, check the first box - 'Start Minimized'.

If you want the close button to minimize Macro Scheduler instead of closing it, check the box marked 'Close Button Minimizes'

By default it is possible to have more than once instance of Macro Scheduler running at the same time. However, if you need to ensure that only one instance can run at a time, uncheck the third option 'Allow Multiple Instances'.

Check 'Large Toolbar Buttons' for larger tool bar buttons with text.

By default the option, 'Allow Macro Scheduler to exit if a macro is running', is selected. This means that if a macro is running and Macro Scheduler should be closed, the macro will end and Macro Scheduler will terminate. You may prefer to make sure Macro Scheduler can't close if a macro is running. However, if you want Windows to be able to close down even if a macro is running, you would need to leave this option switched on.

External Editor

The final part of this form allows you to assign an external editor to use for editing scripts, instead of (or aswell as) the built in Macro Scheduler editor. If you have a preferred text editor or require more specific or extensive features only available in your preferred editing program, you can use this option to link Macro Scheduler to your editor so that when opening a script for editing it opens in the external program.



Boxer 99 Text Editor

If you do wish to use an external editor, we strongly recommend Boxer 99, from Boxer Software, which provides Macro Scheduler syntax highlighting and script templates. For more information, please see

the Editor topic.

Run Now

Runs the selected macro. See [Playing Scripts & Macros](#).

Record

Displays the record dialog. See [Recording Macros](#).

Break

Stops running macros. SHIFT-ESC will also stop macros without Macro Scheduler having to be active. There is also a break option on the system tray menu.

Contents

Displays the help file contents.

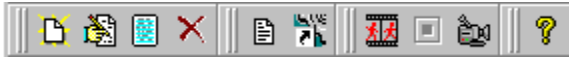
Register

Displays the registration dialog box. See [Registration](#).

About

Displays the About dialog box. This is useful to determine which version of Macro Scheduler you are using, who it is registered to and for a quick link to our web site.

Toolbars



Buttons from Left to Right are :

[New Macro](#)
[Macro Properties](#)
[Edit Script](#)
[Delete Macro](#)

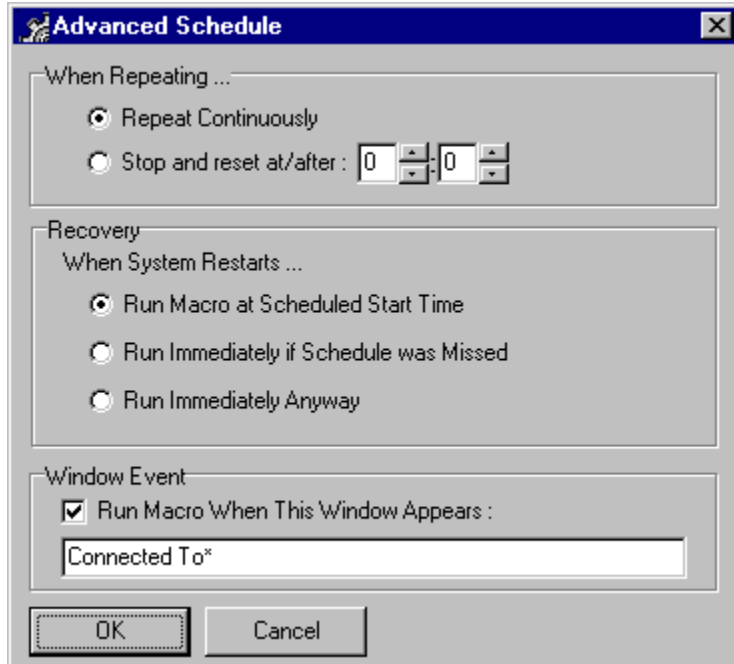
[New Group](#)
[Delete Group](#)
[Group Properties](#)

[View LogFile](#)
[Desktop Shortcut](#)

[Run Macro](#)
[Stop Macro \(Break\)](#)
[Record Macro](#)

[Help](#)

Advanced Scheduling Options



This window is accessed from the scheduling options of the macro properties window.

When Repeating

These options are used when you have set a repeat interval on the main schedule. The default option is repeat continuously. What this means is that even after the system restarts the schedule will continue to repeat (as long as the current day is a day chosen in the main schedule). After restarting, Macro Scheduler will set the next schedule to the current time plus the repeat interval.

The other option will tell Macro Scheduler not to keep repeating, but only to repeat until the specified time. At that point the original schedule time is reset and the macro stops repeating.

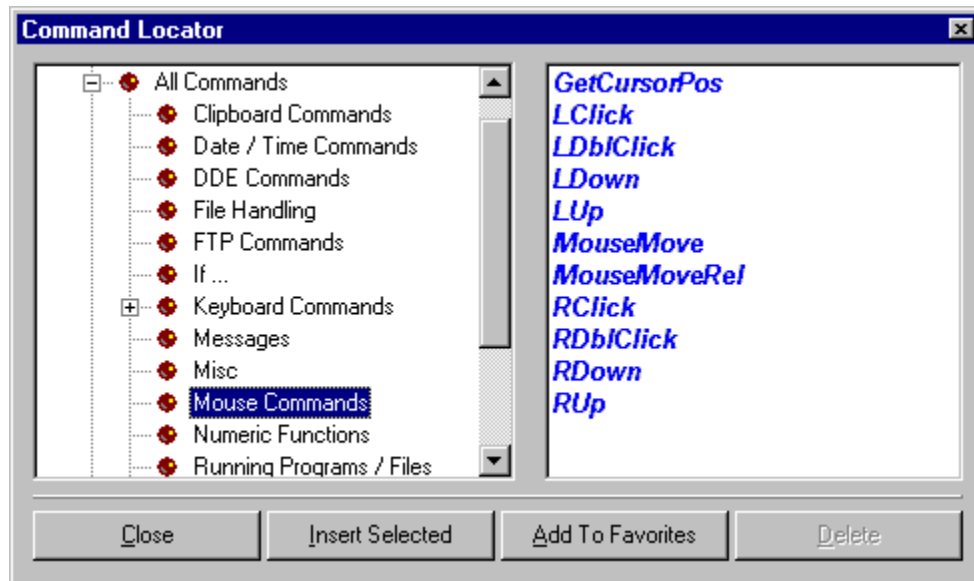
Recovery

Here there are three options. The default is to do nothing except run the macro at the scheduled start time, as normal. If the second option is set, then when Macro Scheduler restarts it checks to see if a macro should have run on that day. If a macro was supposed to run on that day and at a time prior to the system restarting, the macro will be run immediately. The third option will simply allow you to run the macro on startup regardless.

Window Event

Set this to have the macro respond and run when a specific window appears. Check the box and enter the window title. The window title can end in an asterisk as with the window commands in Macro Scheduler. By ending the window title with an asterisk, the macro will run if a window whose title contains the window name specified appears. Without the asterisk it will only run if a window whose title exactly matches the specified window appears (including case). See [SetFocus](#) for more detail on the window commands.

Command Locator



The command locator is available from the [Editor](#). It organises all the script commands into convenient categories, and also allows commands to be added and removed from a Favorites category.

Select a category to see the list of commands in the right hand pane. To insert the selected command into the editor, double click the command, or select it and click 'Insert', or right click and select 'Insert' from the pop-up menu.

Click 'Add to Favorites' to add the selected command to the favorites folder. With the favorites folder active you can remove an item from Favorites by clicking Delete.

Macro Scheduler

Version 6.2 - With MacroScript

Copyright © 1997-2002, MJT Net Ltd

Welcome to Macro Scheduler!

Some Useful Starting Points :

[Getting Started](#)

[Toolbar](#)

[Menu Commands](#)

[Command Reference](#)

[License Agreement](#)

[Purchasing / Registration](#)

[History](#)

How to contact us :

Info : info@mjtnet.com

Support : support@mjtnet.com

Sales : sales@mjtnet.com

Phone : +44 (0)7976 691 276

Fax : +44 (0)870 055 8176

Web : <http://www.mjtnet.com/>

MJT Net Ltd, 4A Oxford Street, Tynemouth
Tyne & Wear, NE30 4PR, UK

History

This is version 6.2.3 of Macro Scheduler

Version 6.2.3 21/03/2002

Removed limit on number of variables per script.
Added VBS_TIMEOUT for VBScript code.
Improved resource usage issues.
Fixed missed repeating schedule issue.

Version 6.2.2 03/01/2002

Added WF_TYPE switch to switch off CHILD/INVISIBLE window search:
Let>WF_TYPE=0 - No Child Windows
Let>WF_TYPE=1 - ALL Windows (Default)
Let>WF_TYPE=2 - Visible Windows Only
Modified Input Box function to ensure it always appears on top.
Addressed some general schedule issues.
Addressed some string to integer conversion issues.

Version 6.2.1.4 05/12/2001

Fixes/Improvements:

Fixed GetWindowPos - problem if X coordinate already an assigned variable.
Fixed problems when sending Shift in conjunction with other extended keys (occurred on some Win2k Installations, but worked fine on other OSs)
Fixed wrong hotkey being displayed on editing macro if hotkey previously selected was system key and a function key larger than F9.
Improved status display for running macros.
More graceful/helpful when attempt to edit and close running macros (macros that cannot be saved).
Improved all Window Functions to identify child windows (Including MDI children)

Added:

Now possible to set focus to MDI Child windows.
Added Tools/View System Windows - option to display all system windows.
Added HTTPRequest command.
Added FTPDelFile command.
Added FTPRenameFile command.

Version 6.2.0.3 05/06/2001

Fixes:

Fixed problem with GetRectChecksum value not remaining static.
Fixed problem caused by entering a window name containing a colon in the window event setting.

Added:

Added SMTP_PORT variable for optionally setting port for SMTPSendMail command

Version 6.2.0.2 14/05/2001

Fixes:

Fixed problem with SMTPSendMail sending attachments
Removed spurious message that sometimes popped up on editing macro

Version 6.2.0.1 24/04/2001

Fixes:

Fixed bug where macros in subfolders could get duplicated in main directory and not updated in certain circumstances when saving from the internal script editor.
Fixed problem introduced in 6.2 stopping macros being executed from system tray menu.
Fixed problem where wrong macro could be executed from the system tray menu.
Fixed problem with hot key assignation - wrong hot key showing and potential integer error.
Fixed spelling mistake on Options dialog.
Fixed spelling mistakes in help file.

Version 6.2 18/04/2001

New Features:

Additional hotkey combinations.
Added SMTP login authentication support to SMTPSendMail command.
Added FTP passive mode option.
Added FTPGetDirList command to retrieve directory listing.
Added ability to retrieve result of FTP operations.
Scroll position in macro list maintained after editing macro.
Added option to disallow multiple instances.

Fixes/Improvements:

Fixed accessibility problems - menus now support screen readers for blind/partially sighted.
Fixed bottom of tray menu disappearing off screen.
Fixed error if startup macros not in main directory.

Changes:

Removed old style buttons and associated options.
Changed encryption for encrypted macros.

Version 6.1.0.4 28/11/2000

Fixes:

Fixed problem with Macro Scheduler sometimes preventing system close down
Fixed exception occurring when using an embedded undeclared variable
Fixed 'Allow Macro Scheduler to exit if a macro is running' option
Added commands missing from 'Script Commands' section of help file

Version 6.1 19/09/2000

New Commands :

GetFileList
MClick
MDoubleClick
MDown
MUp
Repeat
Until
Separate
ShutDownWindows
WaitKeyDown

New Features :

Added numpad keys to hotkey selection

Fixes/Improvements :

Rebuilt SMTPSendMail function
Added support for wildcard attachments in SMTPSendMail
Removed unnecessary polling of macros.dat
Modified variables to allow embedding variables in variables (to make array type variables)
Fixed external editor problem when opening macros in directories other than default directory
Fixed invalid desktop shortcut issue on Windows 2000
Fixed help file entry for SMTPSendMail command
Fixed problem with /NOSYSTRAY command line option when used with a macro on command line
Fixed problem with tray menu not showing on first show
Stopped menu items being draggable
Reviewed and updated FTP component
Fixed problem with Press/Release ALTGR

Tested successfully under Windows 2000 and Windows ME.

Version 6.0 31/01/2000

New Commands :

GetCheckBox command.
SetCheckBox Command.
WaitReady
SMTPSendMail

New Features :

Macro Groups
Debugger
Graphical Menu Bars
Variable playback speed for recorded macros

Fixes/Improvements :

Macro Multi-select

Added default to OK button on password form.
Fixed checking/unchecking of sort menu options.
Added graphical system tray pop up menu.
Stopped recorded macros showing events caused by stopping record.
Fixed convert recorded macro to change icon to script in list when converted.
Fixed access violation occurring when pressing old style edit button when no macro selected.
Stopped delete macro menu option being enabled when no macro highlighted.
Added missing commands to macro properties macro list drop down.
Added CTRL-S keyboard shortcut to Save option in editor.
Fixed problems with sorting from menu options.
Macro Scheduler now remembers which column was sorted and in which direction, and shows sort on startup.
Fixed access violation when attempting to edit script when no script is selected. Disabled edit script buttons while no script selected.
Fixed right click Macro Group menu showing only for selected group.
Removed duplicate IfWindowOpen from Command Locator.
Modified macro record to hide Macro Scheduler window just prior to recording starting.
Fixed problem with macro playback not resetting stop and play buttons and menu options.
Changed method by which repeating macro's schedules are changed on startup.
Instead of scheduling interval mins from now, advance schedule from original start time by interval until schedule time greater than now ... therefore steps in accordance with interval.
Fixed keyboard shortcuts that didn't work in editor.
Fixed repositioning of toolbar buttons when resizing from large to small buttons.
Fixed replacement of small buttons when changing from large button view.
Fixed problems passing variables to other macros.
Fixed crash that occurs on some systems when showing pixel color in cursor position monitor.
Fixed problem with timer stopping after manually running encrypted macro.
Added option to force RegistryWriteKey to write integer values as strings.
RegistryReadKey now reads binary data values.
Fixed problem with Input stopping subsequent Ask and Message in command line scripts.
Added support for national decimal placeholder (regional settings) in numeric functions.
Fixed date error that occurred with some system date configurations when scheduling repeat/stop and reset macros.
Macro list view now updates when a schedule time is reset by repeat/stop and reset macros.
Fixed problem with FileDate reassigning values to existing variables.

Version 5.1 20/08/99

New Commands :

Ascii
DateStamp
GetPixelColor
GetRectCheckSum
LRError
LRStatus
MoveWindow
RegistryDelKey
RegistryDelVal
RegistryReadKey
RegistryWriteKey
Rendezvous
ResizeWindow
TransactionEnd
TransactionStart

WaitRectChanged
WindowAction

Modified/Improved Commands :

If - extra parm, and not equal operator
Send Character/Text - SK_DELAY for slowing send text down
FTP commands - switch off/on status window
All mouse click commands - handles reversed mouse buttons
Random - RND_SEED to allow setting seed
CopyFile - CF_OVERWRITE parameter
Fixed FileSize command
Fixed FileCount with no path problem.
Fixed WaitPixelChange timeout variable problem.

General Fixes/Improvements :

Focus edit area when editor opened
Restore editor when opening editor if minimised
Fixed problem with translated macros failing to repeat
Fixed invalid integer error if repeat interval empty
Added message if no log file associated when view log file selected
Fixed sorting issues (order and header graphic) when using menu options
New send text engine - supports all characters
Improved sending of variables between macros and on command line
- parameters containing '/' chars not split up.
Fixed 'index out of bounds' error associated with window event schedule option

Version 5.04 - 10/03/99

Fixes :

Fixed Find/Replace dialog so as not to be resizeable.
FTP exception when running from command line fixed
Fixed problem with Change Directory if dir does not exist.
Fixed editor save/dirty flag
Fixed syntax highlighting problems
Fixed missing chevrons on some commands with drop down inserter
Fixed problem with assigning values to existing variables.

Version 5.03 - 02/03/99

Fixes :

Fixed singled % problem in command strings
Close down problem with 'allow exit when macro running' option fixed.
Fixed MouseOver context sensitive help
Save from editor updates macro immediately so that changes take effect even while editor open.
Fixed problem with Edit button on macro properties dialog

Additions/Changes :

WaitPixelColor command
TimeStamp command

Random command
Added milliseconds to logging times
Command line option to turn system tray support off
Improved SetFocus code for use with Win98
Improved script command context sensitive help handling
Added command help to Command Locator
Improved editor
Added external editor option.

Version 5.02 - 19/11/98

Fixes :

Fixed problem with 'Allow exit if macro running' option and scheduled macro.
Fixed Edit Line/Update Problem.
Fixed ReadLn link in Help File.

Additions/Changes :

If editor or builder behind main form, it will now be focused if edit script / or macro properties selected.

Version 5.0 - 21/10/98

Fixes :

Windows shutdown problem fixed.
System tray menu sticking fixed.
Fixed Context Sensitive Help for Add command.
Fixed help file for FTP command examples.
Fixed SetFocus command for Win98.
Fixed Problem with FTP commands not working after a failure.
Fixed exceptions on termination if FindWindowWithText macro running.
Fixed schedule/hotkey scripts stopping manually run macros.

Additions/Changes :

Modified interface
More info on main form
Improved editor
Command Locator / Favorites list.
System Tray Macro Menu
Startup Options
Window Event in Advanced Scheduling
Window Mode and return values for Run Program command
Message boxes can be set for stay on top, stay centered
Input box with file browser and default value.
Script commands no longer case sensitive
Script commands can be left indented (i.e. leading spaces no longer a problem).
Press NP Enter command
Global hotkey SHIFT-ESC stops macros running.
Alt keys and Tab indexing improved on macro form.
Remembers if command list sorted.
Let command now does addition, subtraction, multiplication, division and concatenation.
Goto command now accepts variables.

CapsOn command abbreviation is now CAP.

FTP Status Window.

New Commands :

- DayOfWeek
- DDEPoke
- DDERequest
- GetCursorPos
- Sec
- Min
- Hour
- PushButton
- ReadLn
- WriteLn
- WaitCursorChanged
- MouseOver

New System Variables :

- USER_NAME
- COMPUTER_NAME

Microsoft Visual Basic Scripting Edition support (VBScript Edition only)

Version 4.3.6 - 18/04/98

Fixes :

Monthly settings display initialisation fixed.

Use of keyboard to highlight multiple lines in editor.

Uses Arial font in macro list if Tahoma unavailable.

Additions/Changes :

System variables added : OS_VER, WIN_DIR, SYS_DIR, LF, CR, CRLF

More colours in script editor

Macros can now be renamed

Macro Scheduler .scp files can be run from command line and can therefore be associated.

Improved encryption settings - macros can now also be protected when run.

PlayWav command to play .wav files

MessageModal command to allow modal messages.

CountFiles command to count files in a directory (and optionally subdirectories).

Built in FTP commands - FTPGetFile, FTPPutFile.

FileSize command to return size of given file in bytes.

FileDate command to return date of file (YYYYMMDD).

FindWindowWithText command improved to avoid use of clipboard.

Ask command to confirm choice from user.

Length command to get length of string.

Cut, Copy, Paste popup menu added to Editor.

Edited/New macro now highlighted in list on return to main window.

Macro is updated if closing macro window while editor still open.

Smaller executable!

Version 4.3.5 - 07/03/98

Fixes :

Stopped Run Now button and icon resetting when macro run from another macro.

Stopped Wait.. commands taking 100% cpu on Win95.

Fixed recovery of repeating schedules.

Additions/Changes :

Modified Add & Sub commands to add/subtract days to/from dates.
Three new date commands : Day, Month, Year
Variables passed in Macro command can now reference other previously allocated variables.
Added CloseWindow command to close windows and applications.
Added GetActiveWindow command to return title and X,Y coords of active window.
Added support for more hotkeys (including function keys).
New option in advanced scheduling to set macros to always run on startup.
Added embedded variable assignation using %variable% format.
Windows 95 special keys added.

Version 4.3.4 - 31/01/98

Fixes :

Exception errors on NT3.51 when running scripts (introduced in version 4.3.3) fixed.
Memory leaks with wildcarded window operations plugged.
Wait times of less than one second now accurate.
Fixed sticky message window when script run from desktop shortcut.
Fixed problem with double click and F1 on command in editor bringing up wrong help topic.
Problems with Macro Scheduler stopping system from shutting down should be fixed for good now.
Stopped startup schedule checks from running when launching a script/macro from command line.
Addressed problems with unusual system shortdate formats.

Additions/Changes :

Added icons to macro list for more information.
When translating macros to scripts, the removal of adjacent mouse move commands is now optional.
The cursor position monitor can be switched between relative and absolute.
Added ability to show cursor position at mouse cursor so that position can be seen even when macro window is concealed.
Added timeout and success checking to WaitWindowClosed/Open commands.
More hotkey combinations are now available.
Added the ability to pass variables to scripts with the Macro> command.

Version 4.3.3 - 10/01/98

Fixes :

Fixed bug where schedule would only work if system time separator was a colon.

Additions/Changes :

Variable support added to Change Directory and MouseMove commands.
MouseMoveRel command. Works like MouseMove, but moves relative to active window.
GetWindowPos command to retrieve upper left coords of given window.
GetClipboard function to retrieve contents of the clipboard.
PutClipboard to insert values into the clipboard.
MoveFile, CopyFile, DeleteFile now handle wildcards.
FindWindowWithText - locate and setfocus to window containing specified text somewhere within it.
Window handling error messages improved to indicate the window that was being sought.

System tray icon now flashes when a script is running.
Context sensitive help added to forms, macro command list and macro editor.

Version 4.3.2 - 12/12/97

Fixes :

All international characters now supported.
Macro name illegal character error now trapped to avoid file errors.
Add and Sub routines - format of whole numbers altered (zero decimals removed).

Additions :

Ability to sort command list.
MidStr command to extract substrings from strings.
Position command to return position of one string in another.
Command reference added to help file.

Version 4.3.11 - 18/11/97

Fixes :

Fixed startup date problem which occurred on NT platforms.

Version 4.3.1 - 01/11/97

Additions :

New commands for handling Num Lock, Scroll Lock and Number pad keys.
GetDate and GetTime commands.
Command to read parameters from INI files.
Ability to pass parameters on command line.
Advanced scheduling options for system recovery and repeat stop time.
Ability to more quickly disable/enable a macro's schedule.
Macro command now allows running of scripts as well as recorded macros.

Version 4.3b - 11/10/97

Fixes :

Stopped Hotkey setting getting lost after scheduled macro run.
Month date schedule problem fixed.

Version 4.3 - 28/09/97

Fixes :

Update button no longer updates wrong line if another line selected after pressing Edit Line.
Stop button now available when script is scheduled, rather than just the break option.

Additions :

Now works on NT3.51

Mouse cursor position monitor added to settings form.

Default font settings for editor can be changed.

Macro translator ignores unnecessary duplicate control key presses.

Repeat every x minutes now allows up to 1440 minutes.

Variables can now be declared using new Let> command.

All existing commands which take a parameter can accept a declared variable.

Input box command to get a value from the user.

ConCat command to concatenate string values.

Add command to add two numeric values.

Sub command to subtract a numeric value from another.

If command to perform conditional branching based on boolean operation.

CreateDir command to create directories.

Version 4.2.3 - 23/09/97

Fixes :

Correct handling of caps lock state in macro recording and translation, to ensure characters are sent in intended case.

Additions :

Three new commands: Press CAPS, CapsOn, CapsOff

Version 4.2.2 - 20/09/97

Fixes :

'Send Character/Text' now handles extended keys that use the right hand alt key such as the German '\ character.

The recorded macro translator also now handles these keys.

Version 4.2.1 - 16/09/97

Fixes :

System keys released automatically when scripts run from hot key.

ExecuteFile command hanging fixed.

Additions :

2 New Commands (IfFileExists,IfFileChanged)

Version 4.2 - 08/09/97

Additions :

Recorded macros can optionally be translated to scripts and edited.

Ability to assign hot keys to macros.

Commands for controlling the mouse.

Version 4.1.2 - 31/08/97

Fixes :

Memory leaks investigated and patched.

Additions :

Added the ability to stop macro recording using a user defined hot key to avoid the need to set focus back to the Macro Scheduler window.

Addition of ALT GR key to key list.

Version 4.1.1 - 14/08/97

Fixes :

Fixed bug causing exception error when script containing a message statement is run from the command line.

Version 4.1 - 06/08/97

Changes :

Altered window handling commands to only attempt substring window title matches when an asterisk is specified at the end of the text.

Version 4.0 - 26/07/97

Fixes :

Fixed bug with logging.

Fixed NT4.0 close down problem.

Fixed problem with looping causing errors if more than one script running at once.

Additions/Changes :

Execute File method to launch files based on their associations.

EditIniFile command to edit INI files.

Added Edit Line Button on script creation window.

Macro list now sorted alphabetically.

Browse button for Run Program command.

Changed window handling commands to work with substrings.

added Help button.

Added ability to stop a macro during execution.

Added automatic creation of desktop shortcuts.

Re-coded wait command to multi-task better and not lock up so tightly.

'Jazzed' up interface a little.

Added more error checking.

Version 3.0 - 01/07/97

Additions :

Logging capability.
Encryption facility.
Free format editor.
Import method.
CopyFile, MoveFile, DeleteFile commands.
Message Command.
Enhanced error checking.

Version 2.0 - 08/06/97

Added macro recording capability.
Changed repeat every from hours to minutes.
Added monthly run option.
Fixed command line bug.

Large Buttons

Toggles the toolbar buttons between regular small image only buttons, and larger buttons with text as well as images.

MouseOver

MouseOver>window_title,button/object_caption

Attempts to position the mouse cursor over the specified button (or object) of the specified window.

window_title can contain an asterisk (*) as with all other window functions. For buttons that have a hot key associated with them, and represented on the button by an underscored letter, pass a & character before that letter. e.g.: for a button called 'Close', send &Close.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to locate the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and stops at the first one it finds whose title contains the entered text. This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

Abbreviation : [MVR](#)

Example

```
Run Program>rundll32.exe shell32.dll,Control_RunDLL TimeDate.cpl
WaitWindowOpen>Date/Time Properties
...
MouseOver>Date/Time*,OK
LClick
```


VBEND

Only Available in VBScript Edition

VBEND

Marks the end of a block of VBScript code. Macro Scheduler reads from VBSTART to VBEND and stores the VBScript code between the two markers for later execution by the VBRun, or VBEval commands.

See also : [VBEval](#), [VBRun](#), [VBSTART](#)

Example

VBSTART

```
Function MultiplyNums (d,a)
```

```
    MultiplyNums = d * a
```

```
End Function
```

```
Sub DisplayMessage (msg)
```

```
    MsgBox msg
```

```
End Sub
```

```
Function GetName
```

```
    GetName = InputBox("Enter Your Name : ")
```

```
End Function
```

VBEND

```
Let>a=5
```

```
VBEval>MultiplyNums(%a%,2),answer
```

```
MessageModal>answer
```

```
VBRun>DisplayMessage,Hello World
```

```
VBEval>GetName,name
```

```
MessageModal>Your Name is : %name%
```

VBEval

Only Available in VBScript Edition

VBEval>Function([parms]),result

Evaluates a VBScript expression, or function in the proceeding VBScript code block. Parameters can be passed to the function. The result of the function is stored in the result variable which is a regular Macro Scheduler variable.

This command sometimes causes confusion. As it is evaluating a VBScript expression the syntax used in the first part of the command - the VBScript expression - should be valid VBScript syntax. To pass Macro Scheduler variables, embed them with the % symbol. If passing a Macro Scheduler variable as a string, remember that VBScript expects strings with quote marks around them. See examples below.

It is possible to set the VBScript code timeout using the VBS_TIMEOUT variable. By default VBScript code will never timeout. To set a timeout, set VBS_TIMEOUT to a value in milliseconds.

Abbreviation : [VBE](#)

See also : [VBRun](#)

Example

VBSTART

```
Function MultiplyNums (d,a)
```

```
    MultiplyNums = d * a
```

```
End Function
```

```
Sub DisplayMessage (msg)
```

```
    MsgBox msg
```

```
End Sub
```

```
Function GetName
```

```
    GetName = InputBox("Enter Your Name : ")
```

```
End Function
```

```
Function PointlessStringExample(somestring)
```

```
    DisplayMessage(somestring)
```

```
    StringExample = somestring
```

```
End Function
```

VBEND

```
Let>a=5
```

```
VBEval>MultiplyNums(%a%,2),answer
```

```
MessageModal>answer
```

```
VBRun>DisplayMessage,Hello World
```

```
VBEval>GetName,name
```

MessageModal>Your Name is : %name%

Let>text=Hello World

VBEval>PointlessStringExample("%text%"),sametext

VBSTART

Only Available in VBScript Edition

VBSTART

Marks the start of a block of VBScript code. Macro Scheduler reads from VBSTART to VBEND and stores the VBScript code between the two markers for later execution by the VBRun, or VBEval commands.

See also : [VBEval](#), [VBRun](#), [VBEND](#)

Example

VBSTART

```
Function MultiplyNums (d,a)
```

```
    MultiplyNums = d * a
```

```
End Function
```

```
Sub DisplayMessage (msg)
```

```
    MsgBox msg
```

```
End Sub
```

```
Function GetName
```

```
    GetName = InputBox("Enter Your Name : ")
```

```
End Function
```

VBEND

```
Let>a=5
```

```
VBEval>MultiplyNums(%a%,2),answer
```

```
MessageModal>answer
```

```
VBRun>DisplayMessage,Hello World
```

```
VBEval>GetName,name
```

```
MessageModal>Your Name is : %name%
```

VBRun

Only Available in VBScript Edition

VBRun>Subroutine[,Parm1,Parm2,...]

Executes the specified VBScript subroutine in the proceeding VBScript code block. Parameters can be passed to the subroutine. Macro Scheduler allows up to 25 parameters to be passed. The number of parameters passed must equal the number expected by the subroutine, otherwise a run time error will occur.

Macro Scheduler variables may be used in any part of the command.

It is possible to set the VBScript code timeout using the VBS_TIMEOUT variable. By default VBScript code will never timeout. To set a timeout, set VBS_TIMEOUT to a value in milliseconds.

Abbreviation : [VBR](#)

See also : [VBEval](#)

Example

VBSTART

Function MultiplyNums (d,a)

 MultiplyNums = d * a

End Function

Sub DisplayMessage (msg)

 MsgBox msg

End Sub

Function GetName

 GetName = InputBox("Enter Your Name : ")

End Function

VBEND

Let>a=5

VBEval>MultiplyNums(%a%,2),answer

MessageModal>answer

VBRun>DisplayMessage,Hello World

VBEval>GetName,name

MessageModal>Your Name is : %name%

WaitPixelColor

WaitPixelColor>ColorCode,X,Y,Timeout

This command causes Macro Scheduler to wait until the pixel colour at the specified pixel coordinates changes to the colour specified in ColorCode. If it doesn't change to that colour within the number of seconds specified in Timeout, the command stops waiting and the variable WPC_RESULT is set to FALSE. WPC_RESULT is TRUE if the command terminated because the colour changed to the specified colour within the specified time. If Timeout is set to 0, the command will wait indefinitely.

To determine the correct colour code to use, click the drop down menu button next to the cursor position monitor on the macro properties form. From the drop down menu select 'Pixel Color' and the pixel colour of the current mouse cursor position will be added to the display along with the X and Y coordinates. Now you can determine the correct colour code of any pixel on the screen.

Abbreviation : [WPC](#)

See also : [Wait](#), [WaitWindowOpen](#), [WaitWindowClosed](#), [WaitCursorChanged](#), [GetPixelColor](#), [GetRectChecksum](#), [WaitRectChanged](#)

Example

```
WaitPixelColor>16777215,652,355,10
```

```
Message>WPC_RESULT
```

Random

Random>Range,Result

Returns a random number within the specified range where $0 \leq \text{Result} < \text{Range}$.

The seed is set automatically and is stored in the RND_SEED variable. It is possible to set the seed programmatically by modifying the value of RND_SEED.

Result is a variable in which the result is stored.

Abbreviation : RAN

Example

```
Random>6,DiceResult
```

```
Message>You threw a %DiceResult%
```

TimeStamp

TimeStamp>filename,comment

Outputs the time in milliseconds and the given comment to the given text file.

Abbreviation : [TIM](#)

See also : [WriteLn](#), [DateStamp](#)

Example

TimeStamp>c:\temp\mylogfile.txt,Macro Finished

The file entry would appear :

15:43:03:066 - Macro Finished

Ascii

Ascii>ASCII_Code[,ASCII_Code[,ASCII_Code[...]]]

Inserts the characters specified by the given ASCII codes into the current application. If sending more than one ASCII code at once, separate them with commas.

This function is useful for sending non-printable, low, or high ASCII value characters to an editor, which you might otherwise be unable to send using the [Send](#) function.

Abbreviation : [ASC](#)

See also : [Send Character/Text](#)

Example

The following example shows three methods of sending the same text value to Notepad.

```
SetFocus>Notepad*
Ascii>77
Ascii>65
Ascii>82
Ascii>67
Ascii>85
Ascii>83
Press Enter
Ascii>77,65,82,67,85,83
Press Enter
Send>MARCUS
```

DateStamp

DateStamp>filename,comment

Outputs the date and time in milliseconds and the given comment to the given text file.

Abbreviation : [DAT](#)

See also : [WriteLn](#), [TimeStamp](#)

Example

TimeStamp>c:\temp\mylogfile.txt,Macro Started

The file entry would appear :

1999-08-06:13:38:10:352 - Macro Started

GetPixelColor

GetPixelColor>X,Y,result

Returns the pixel color of the specified screen coordinate. Enter the coordinates in X and Y, and specify a variable to store the pixel color in.

Abbreviation : [GPC](#)

See also : [WaitPixelColor](#), [GetRectChecksum](#), [WaitRectChanged](#)

Example

`GetPixelColor>652,355,PC`

`Message>Pixel Color is : %PC%`

GetRectChecksum

GetRectChecksum>TLX,TLY,BRX,BRY,result_value

When passed the top left and bottom right screen coordinates of a rectangle, this function returns the checksum of that rectangle. The checksum is based on the color and position in the rectangle of each individual pixel. The checksum is returned in the variable result_value.

This command will allow for waiting for a specific part of the screen to become equal to a known graphic. Use the command when the graphic to be compared against is present to determine it's checksum. The example below shows a loop which will cause the script to wait until the rectangle contains the required graphic.

TLX - Top Left corner X coordinate

TLY - Top Left corner Y coordinate

BRX - Bottom Right corner X coordinate

BRY - Bottom Right corner Y coordinate

Abbreviation : [GRC](#)

See also : [WaitRectChanged](#), [WaitPixelColor](#), [GetPixelColor](#)

Example

Label>waitforit

GetRectChecksum>10,10,50,50,cs

If>cs=6.20066481623195E16,Done

Goto>waitforit

Label>Done

Message>There You Go

MoveWindow

MoveWindow>window_title,new_X,new_Y

Moves the window with the specified window title to the new coordinates. new_X,new_Y denotes the new upper left corner position. The window_title may contain the * symbol at the end to indicate a wildcard.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to move the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and selects the first one it finds whose title contains the entered text. This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

It is possible to limit the type of windows this command effects using the WF_TYPE variable:

Let>WF_TYPE=0 - No Child Windows

Let>WF_TYPE=1 - ALL Windows (Default)

Let>WF_TYPE=2 - Visible Windows Only

Abbreviation : [MVW](#)

See also : [ResizeWindow](#)

Example

[MoveWindow](#)>notepad*,5,5

RegistryDelKey

RegistryDelKey>root_key,key

Removes a key from the registry.

Use all registry functions with caution. If you are unfamiliar with the Windows Registry we recommend that you do not use these functions. Removing or modifying a registry entry that you did not create could cause your system to become unstable.

Abbreviation : [RDK](#)

See also : [RegistryDelVal](#), [RegistryReadKey](#), [RegistryWriteKey](#)

Example

[RegistryDelKey](#)>HKEY_CURRENT_USER,Software\MJTNET\Temp

RegistryDelVal

RegistryDelKey>root_key,key,Value

Removes a value from the registry.

Use all registry functions with caution. If you are unfamiliar with the Windows Registry we recommend that you do not use these functions. Removing or modifying a registry entry that you did not create could cause your system to become unstable.

Abbreviation : [RDV](#)

See also : [RegistryDelKey](#), [RegistryReadKey](#), [RegistryWriteKey](#)

Example

[RegistryDelVal](#)>HKEY_CURRENT_USER,Software\MJTNET\Temp,TestVal1

RegistryReadKey

RegistryReadKey>root_key,key,entry,result_variable

Reads the value of an entry from the registry. The value is stored in the given variable.

Use all registry functions with caution. If you are unfamiliar with the Windows Registry we recommend that you do not use these functions. Removing or modifying a registry entry that you did not create could cause your system to become unstable.

Abbreviation : [RRK](#)

See also : [RegistryDelKey](#), [RegistryDelVal](#), [RegistryWriteKey](#)

Example

```
RegistryReadKey>HKEY_CURRENT_USER,Control Panel\Colors,ActiveTitle,VActTitle
Message>VActTitle
```


RegistryWriteKey

RegistryWriteKey>root_key,key,entry,value

Creates or modifies a registry entry. If the key and entry do not exist they are created and the new value assigned to the entry. If the key and entry already exist, the value is changed to the value provided. The function will create integer entries if the value specified is an integer, or else the new value will be a string. To force an integer value to be written as a string set REG_INTASSTR to 1 before calling RegistryWriteKey.

Use all registry functions with caution. If you are unfamiliar with the Windows Registry we recommend that you do not use these functions. Removing or modifying a registry entry that you did not create could cause your system to become unstable.

Abbreviation : [RRK](#)

See also : [RegistryDelKey](#), [RegistryDelVal](#), [RegistryReadKey](#)

Example

[RegistryWriteKey](#)>HKEY_CURRENT_USER,MyStuff,MyName,Fred Bloggs

ResizeWindow

ResizeWindow>window_title,new_width,new_height

Resizes the window with the specified window with the new width and height dimensions. The window_title may contain the * symbol at the end to indicate a wildcard.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to resize the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and selects the first one it finds whose title contains the entered text. This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

It is possible to limit the type of windows this command effects using the WF_TYPE variable:

- Let>WF_TYPE=0 - No Child Windows
- Let>WF_TYPE=1 - ALL Windows (Default)
- Let>WF_TYPE=2 - Visible Windows Only

Abbreviation : [RSW](#)

See also : [MoveWindow](#)

Example

[ResizeWindow](#)>notepad*,500,300

WaitRectChanged

WaitRectChanged>TLX,TLY,BRX,BRY,Timeout

This command causes Macro Scheduler to wait until the image bound by the specified pixel coordinates changes. If it doesn't change within the number of seconds specified in Timeout, the command stops waiting and the variable WRC_RESULT is set to FALSE. WRC_RESULT is TRUE if the command terminated because the image changed within the specified time. If Timeout is set to 0, the command will wait indefinitely.

TLX - Top Left corner X coordinate

TLY - Top Left corner Y coordinate

BRX - Bottom Right corner X coordinate

BRY - Bottom Right corner Y coordinate

Abbreviation : [WRC](#)

See also : [Wait](#), [WaitWindowOpen](#), [WaitWindowClosed](#), [WaitCursorChanged](#), [WaitPixelColor](#), [GetRectChecksum](#)

Example

[WaitRectChanged](#)>10,10,100,100,10

[Message](#)>WPC_RESULT

WindowAction

WindowAction>Action,window_title

Use this function to restore, minimize, maximize or close a window.

Action can be any one of the following:

- 0: Restore the window
- 1: Maximize the window
- 2: Minimize the window
- 3: Close the window

Specify the window name in window_title. The window_title may contain the * symbol at the end to indicate a wildcard.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to select the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and selects the first one it finds whose title contains the entered text. This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

It is possible to limit the type of windows this command effects using the WF_TYPE variable:

- Let>WF_TYPE=0 - No Child Windows
- Let>WF_TYPE=1 - ALL Windows (Default)
- Let>WF_TYPE=2 - Visible Windows Only

Abbreviation : [WIN](#)

See also : [MoveWindow](#), [ResizeWindow](#), [CloseWindow](#)

Example

[WindowAction](#)>2,notepad*

Registration

Ordering Macro Scheduler in Australia from Aquarian Technologies

How do I register ?

You can contact Aquarian Technologies in any of the following ways:

Sales: 1800-240-774
Telephone: +61 (0)3 5476-2005
Fax: +61 (0)3 5476-2008

Internet Orders: sales@aquatee.com
WEB Site: www.aquatee.com

Aquarian Technologies
PO Box 820
Castlemaine, VIC, 3450
Australia.

[Click here to print out an order form.](#)

You can order from Aquarian Technologies by payment with cheque, credit card, postal money order or purchase order (approved customers only). The credit cards accepted are: MasterCard, Visa, AMEX and BankCard.

Are there any multi-user licenses ?

Yes, please contact us for pricing information on discounted site licences and multiple copies.

[Click here to print out an order form](#)

Registration

Ordering Macro Scheduler in UK, USA and Rest of World

How much does it cost ?

The cost of registering one copy of Macro Scheduler is :

40 USD or 25 GBP

One copy of Macro Scheduler VBScript Edition is :

75 USD or 47 GBP

NB: EC residents must add 17.5% VAT.

Are there any multi-user licenses ?

Yes, the following table outlines the current licensing options available and their costs :

License	Macro Scheduler 5.0		VBScript Edition	
	Price (USD)	Price (GBP)	Price (USD)	Price (GBP)
5	150.00	93.75	280.00	175.00
10	260.00	162.50	487.50	305.00
25	600.00	375.00	1125.00	700.00
50	1000.00	625.00	1875.00	1172.00
100	1400.00	875.00	2625.00	1640.00
500	4000.00	2500.00	7500.00	4687.00
1000	6000.00	3750.00	11250.00	7031.00

NB: EC residents must add 17.5% VAT.

How do I register ?

For full details on all ordering methods including ordering on-line please see <http://www.mjtnet.com> or contact us on +44 7976 691 276

When ordering on-line you will **immediately*** be sent an email which contains details of where to download the fully licensed version.

* This may not apply for other resellers/agents. Sales via our own website are carried out in real time and a download link will be emailed immediately after completing the order form.

Please use this form when ordering by fax or mail:

Name _____
Company _____
ABN _____
Address _____

City/Town _____
Country _____ Post Code _____
Telephone _____
EMail _____

☐ Cheque ☐ AMEX ☐ Money Order
☐ BankCard ☐ VISA ☐ MasterCard

Name on Card _____
(Only if different from Name above)

Credit Card Number _____

Expiration Date ____ / ____

Signature _____

____ Copies of Macro Scheduler™ @ \$AUD 81.95 = \$____
____ Copies of Macro Scheduler VBScript @ \$AUD 146.30 = \$____

[Note: Prices include GST & Express Shipping & Handling]

Where did you hear about Macro Scheduler ?

What new features would you like to see ?

Please make cheques payable to: Aquarian Technologies.

WaitReady

WaitReady>paint_events

WaitReady suspends script execution until the foreground window has finished processing mouse, keyboard, show window, and optionally, paint events. Issue 1 to include paint events, and 0 to exclude paint events. This command can therefore be used to wait until the active application is ready to receive keyboard and mouse events in most situations.

Abbreviation : [WRD](#)

See also : [Wait](#), [WaitWindowOpen](#), [WaitWindowClosed](#), [WaitCursorChanged](#), [WaitPixelColor](#)

Example

```
Run Program>"C:\Program Files\Microsoft Office\Office\WINWORD.EXE"
```

```
WaitWindowOpen>Microsoft Word*
```

```
WaitReady>0
```

```
Message>Word is now ready for input
```


GetCheckBox

GetCheckBox>window_title,object_caption,result

GetCheckBox determines whether or not the given check box, or radio button, is checked. result is set to 1 if checked, 0 if not checked, or -1 if the command failed to locate the given check box. Specify the window title of the window containing the check box/radio button, and the object's caption. The caption of a check box or radio button is the text appearing next to it. The caption must be specified accurately with attention paid to case. Where a letter is underlined indicating a shortcut key, enter the '&' character before it.

window_title can end with an asterisk to indicate a substring match. See [SetFocus](#) for a more detailed explanation of how this works.

Abbreviation : [CBX](#)

See also : [SetCheckBox](#)

Example

```
GetCheckBox>Internet Explorer Prop*,Never dial a &connection,res
```

```
If>res=1,checked,unchecked
```

```
Label>checked
```

```
Message>The checkbox is checked!
```

```
Goto>end
```

```
Label>unchecked
```

```
Message>The checkbox is not checked!
```

```
Label>end
```

SetCheckBox

SetCheckBox>window_title,object_caption,TRUE|FALSE

SetCheckBox is used to check or uncheck a given checkbox or radio button. Specify the window title of the window containing the check box/radio button, and the object's caption. The caption of a check box or radio button is the text appearing next to it. The caption must be specified accurately with attention paid to case. Where a letter is underlined indicating a shortcut key, enter the '&' character before it.

window_title can end with an asterisk to indicate a substring match. See [SetFocus](#) for a more detailed explanation of how this works.

Abbreviation : [SBX](#)

See also : [GetCheckBox](#)

Example

```
GetCheckBox>Internet Explorer Prop*,Never dial a &connection,res
If>res=1,checked,unchecked

Label>checked
SetCheckBox>Internet Explorer Prop*,Never dial a &connection,FALSE
Goto>end

Label>unchecked
SetCheckBox>Internet Explorer Prop*,Never dial a &connection,TRUE

Label>end
```

SMTPSendMail

SMTPSendMail>recipients,server,from_address,from_name,subject,body,attachments

Sends email via an SMTP server to the recipients entered.

recipients : Include one or more email addresses, separated by semicolons (;)

server : The name or IP address of your SMTP server.

from_address : Your email address, or the email address you want the message to come from.

from_name : Your real name, or a string to appear in the from name field.

subject : The subject line.

body : The body text of the message.

attachments : if sending files include each file separated by semicolons (;). If no attachments end the line with a comma.

By default when this command is in use a small status window will appear at the top left of the screen. You can stop this window being displayed by setting SENDMAIL_STATUS to 0.

If you are using an SMTP server that requires authentication, you can enable authentication by setting SMTP_AUTH to 1, and then setting SMTP_USERID and SMTP_PASSWORD to the username and password that you need to connect to the SMTP server. e.g:

```
Let>SMTP_AUTH=1
```

```
Let>SMTP_USERID=myuser
```

```
Let>SMTP_PASSWORD=frogslegs
```

To enable return receipt of the email, set SMTP_RECEIPT to 1.

By default the port used by SMTPSendMail is 25. If you need to change the port number set the value of SMTP_PORT.

The result of the sendmail operation is placed into the variable SMTP_RESULT. The format of this response depends on the SMTP server being communicated with. If successful the value will **contain** the result code 250. Otherwise it will contain the appropriate error message.

Abbreviation : [SMT](#)

Example

```
Let>SENDMAIL_STATUS=1
```

```
Let>subject=Test Message
```

```
Let>me=myname@myplace.com
```

```
Let>myname=Mr Smith
```

```
Let>recipients=mickey@disney.com;someone@somewhere.com
```

```
Input>body,Enter your message:
```

```
SMTPSendMail>recipients,post.mail.com,me,myname,subject,body,
```

```
Message>Result of SendMail: %SMTP_RESULT%
```

Select All

Selects all macros in the macro list.

New Group

Select this to create a new macro group. The new group is created beneath the currently selected group. Therefore, to make a new top level group, select 'All Macros' first.

On selecting this option the [Group Properties](#) dialog is displayed where the new group name and a macro path are selected.

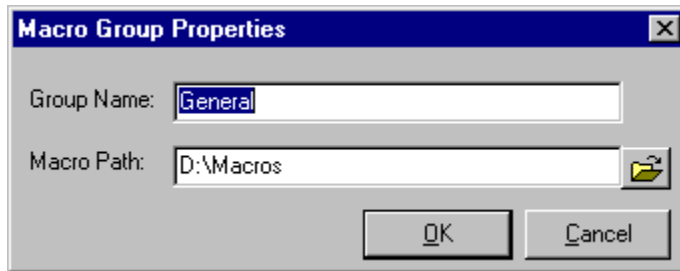
A macro path is simply a folder on a drive or network drive which is used to store the physical files associated with macros that you place in the associated group. By default the main Macro Scheduler directory is used.

Delete Group

Deletes the selected group. A group containing macros cannot be deleted. If an attempt is made to delete a group that has macros a message box will be displayed to this effect. Delete or move any macros before deleting the group.

Group Properties

Displays the group properties for the currently selected group. The name of the group and/or the macro path can be changed.



Using the Debugger

In the Editor there is a menu called 'Debug' with the following options

Step (F8)

This highlights the currently selected line, and then subsequently executes that line and moves to the next line in execution flow.

Stop Debug (CTRL-F2)

Stops debugging and resets everything.

Trace

This displays a small dialog where you can set an interval. Then the debugger auto-steps at the specified frequency.

Run

Runs the script without stepping, from the currently selected line.

Insert Breakpoint

Inserts a breakpoint after the selected line. Simply inserts ****BREAKPOINT**** on the next line. If running the script using 'Run' from the Debug menu, execution will pause when this Breakpoint line is reached and the script can then be stepped from this point.

Refocus Windows (check on or off)

Most of the time you want this checked on. Due to the nature of Macro Scheduler, and the fact that it is commonly used to automate other windows, most scripts will focus other apps and chop and change focus a lot. 'Refocus Windows' ensures that if a command causes focus to shift, focus will be set back to that window before executing each subsequent line so that (hopefully) that line will affect the correct app. Focus is returned to the editor after each line so that you can see what you're doing. Sometimes it is useful to turn this off ... if you're not doing any GUI scripting, or even at some point during a GUI script where you might have a loop, say, calculating some value but not actually requiring focus of anything. So you can switch it on and off during debug as required.

Show Watch List

Displays a list to the right hand side of the editor containing all current script variables and their values - updates as they change.

Notes

Remember that stepping through a script will slow it down, since you will be pausing at each line. Since many scripts that automate windows applications need to be time-sensitive - to make sure things don't happen before apps are ready etc - the process of debugging may give the impression that the script is fine, when it actually needs delays and waits to be built in!

GetFileList

GetFileList>filespec,result

GetFileList returns a list of the files found matching the specified filespec. Each filename is separated by a semicolon (;). For instance, to return the list of files in the temp directory specify c:\temp*.* as the filespec.

Abbreviation : [GFL](#)

Example

```
GetFileList>c:\temp\*.*;files
Separate>files;;file_names
MessageModal>Num Files: %file_names_count%

Let>k=0
Repeat>k
    Let>k=k+1
    Message>file_names_%k%
Until>k,file_names_count
```

Repeat

Repeat>variable

Use in conjunction with Until. Iterates the code from the Repeat statement to the Until statement until the specified variable equals the value specified in the Until statement.

See also : [Until](#)

Example

```
GetFileList>c:\temp\*.*,files
Separate>files,,file_names
MessageModal>Num Files: %file_names_count%

Let>k=0
Repeat>k
    Let>k=k+1
    Message>file_names_%k%
Until>k,file_names_count
```

Until

Until>variable,finalvalue

Use in conjunction with Repeat. Iterates the code from the Repeat statement to the Until statement until the specified variable equals finalvalue.

See also : [Repeat](#)

Example

```
GetFileList>c:\temp\*.*,files
Separate>files,;file_names
MessageModal>Num Files: %file_names_count%

Let>k=0
Repeat>k
    Let>k=k+1
    Message>file_names_%k%
Until>k,file_names_count
```

Separate

Separate>list,delimiter,returnvar

Separate takes a list, a delimiter and returns the elements of the list. The command returns a number of variables, one for each list element, each with the suffix "_n" where n is the index of the element in the list. The variable names are determined by the name given in returnvar. e.g. returnvar_1, returnvar_2, etc. Also returned is the number of elements in the list, in returnvar_count.

Abbreviation : **SEP**

Example

```
GetFileList>c:\temp\*.*,files
Separate>files,;,file_names
MessageModal>Num Files: %file_names_count%

Let>k=0
Repeat>k
    Let>k=k+1
    Message>file_names_%k%
Until>k,file_names_count
```

MClick

Simulates a middle button mouse click at the current point on the screen.

The following commands will produce the same result :

MDown

MUp

Abbreviation : [MCI](#)

See also : [LDown](#), [LUp](#), [LDbIClick](#), [LClick](#), [RDown](#), [RUp](#), [RDbIClick](#), [MDown](#), [MUp](#), [MDbIClick](#), [MouseMove](#), [MouseMoveRel](#)

MDblClick

Simulates a middle mouse button double click at the current point on the screen.

The following will achieve the same result :

MClick
MClick

or

MDown
MUp
MDown
MUp

Abbreviation : [MDb](#)

See also : [LClick](#), [LDown](#), [LUp](#), [LDbClick](#), [RClick](#), [RDown](#), [RUp](#), [MDown](#), [MUp](#), [MClick](#), [MouseMove](#), [MouseMoveRel](#)

MDown

Simulates a press of the middle mouse button. This is like pressing the mouse button down but not releasing it. It is half of a click.

Abbreviation : [MDo](#)

See also : [LClick](#), [LDown](#), [LUp](#), [LDbClick](#), [RClick](#), [RDbClick](#), [RUp](#), [MUp](#), [MDbClick](#), [MClick](#), [MouseMove](#), [MouseMoveRel](#)

MUp

Releases the middle mouse button. It is the latter half of a click.

See MDown.

See also : [LClick](#), [LDown](#), [LUp](#), [LDbClick](#), [RClick](#), [RDbClick](#), [RDown](#), [MDown](#), [MDbClick](#), [MClick](#), [MouseMove](#), [MouseMoveRel](#)

WaitKeyDown

WaitKeyDown>key_to_wait_for

WaitKeyDown pauses execution until the specified key is pressed. For an ordinary character key specify the character of the key. For other keys use the virtual key code preceded by VK:

WaitKeyDown>H - waits for the H key to be pressed

WaitKeyDown>VK101 - waits for virtual key code 101 (Numpad 5) to be pressed.

A list of virtual key codes can be found here: http://msdn.microsoft.com/library/psdk/winui/vkeys_529f.htm

Abbreviation : WKD

See also : [Wait](#), [WaitWindowOpen](#), [WaitWindowClosed](#), [WaitCursorChanged](#), [WaitPixelColor](#), [WaitReady](#)

ShutDownWindows

ShutDownWindows>shutdown_type

Use this command to shutdown or reboot the machine. Set shutdown_type appropriately as follows:

- 0: Shutdown
- 1: Reboot
- 2: Logoff
- 3: Forced Shutdown - forces tasks to terminate, avoids the Wait/Terminate dialogs

Example

```
//Reboot Server  
ShutDownWindows>1
```

FTPGetDirList

FTPGetDirList>Server,Username,Password,port,Local_File,Host_File_Spec,Type

This command will connect to the specified FTP server and download a directory listing for the file spec specified.

The directory listing is output to the local text file given in Local_File.

Type can be either D or L, where D obtains a full directory listing, including directories, subdirectories and their contents. L creates a simple file list, containing just filenames matching the file spec.

When the FTP commands are active a small floating window is displayed at the top left of the screen showing the status of the FTP session. To stop the status window appearing set FTP_STATUS to 0 before issuing an FTP command. Set it back to 1 to show the window. The default value for FTP_STATUS is 1.

The result of the FTP operation is stored in FTP_RESULT.

If you are connecting via a proxy server, enter the name or IP address of the proxy server in 'Server', and for 'Username' specify username@remote_server.

By default the timeout for the FTP commands is set to 15 seconds. To change this set the FTP_TIMEOUT variable.

Abbreviation : [FGD](#)

See also [FTPGetFile](#), [FTPPutFile](#), [FTPDelFile](#), [FTPRenameFile](#)

Example

The following command will retrieve a listing of the entire contents of the server.

```
FTPGetDirList>ftp.domain.com,anonymous,user@domain.com,21,c:\temp\readme.txt,*,D
```

FTPDelFile

FTPDelFile>Server,Username,Password,port,Host_File_Spec

This command will connect to the specified FTP server and delete the file spec specified.

When the FTP commands are active a small floating window is displayed at the top left of the screen showing the status of the FTP session. To stop the status window appearing set FTP_STATUS to 0 before issuing an FTP command. Set it back to 1 to show the window. The default value for FTP_STATUS is 1.

The result of the FTP operation is stored in FTP_RESULT.

If you are connecting via a proxy server, enter the name or IP address of the proxy server in 'Server', and for 'Username' specify username@remote_server.

By default the timeout for the FTP commands is set to 15 seconds. To change this set the FTP_TIMEOUT variable.

Abbreviation : [FDF](#)

See also [FTPGetFile](#), [FTPPutFile](#), [FTPRenameFile](#), [FTPGetDirList](#)

Example

FTPDelFile>ftp.domain.com,anonymous,user@domain.com,21,/pub/readme.txt

FTPRenameFile

FTPRenameFile>Server,Username,Password,port,Host_File_Spec,New_File_Name

This command will connect to the specified FTP server and rename the host file specified with New_File_Name.

When the FTP commands are active a small floating window is displayed at the top left of the screen showing the status of the FTP session. To stop the status window appearing set FTP_STATUS to 0 before issuing an FTP command. Set it back to 1 to show the window. The default value for FTP_STATUS is 1.

The result of the FTP operation is stored in FTP_RESULT.

If you are connecting via a proxy server, enter the name or IP address of the proxy server in 'Server', and for 'Username' specify username@remote_server.

By default the timeout for the FTP commands is set to 15 seconds. To change this set the FTP_TIMEOUT variable.

Abbreviation : [FDF](#)

See also [FTPGetFile](#), [FTPPutFile](#), [FTPDelFile](#), [FTPGetDirList](#)

Example

[FTPRenameFile](#)>ftp.domain.com,anonymous,user@domain.com,21,/pub/readme.txt,/pub/readme.new

HTTPRequest

**HTTPRequest>URL,[LocalFilename],Method,
[Post_Data],Result_Variable[,ProxyServer,ProxyPort]**

Retrieves a web document via the HTTP protocol using either GET or POST methods.

URL: URL of document to retrieve

LocalFileName: Optional (may be left blank) - local file to save response to.

Method: GET or POST

Post_Data: Data to Post to URL if using POST method. Use name=value pairs separated by '&'. See example below.

Result_Variable: Stores result of operation. If successful this will contain the HTML returned. Otherwise it will contain an error message.

ProxyServer: Optional - if using a proxy server set this to domain or IP address of proxy server.

ProxyPort: Optional - if using a proxy server, set to port number of proxy server.

Abbreviation : [HTT](#)

Example

The following line does a simple GET request and saves the resulting HTML to a variable called HTMLResponse:

```
HTTPRequest>http://www.mjtnet.com,,GET,,HTMLResponse
```

The following line does the same thing but also saves the output to a file:

```
HTTPRequest>http://www.mjtnet.com,d:\HTML\mjtnet.html,GET,,HTMLResponse
```

This demonstrates a POST operation, sending name=value pairs to the page:

```
Let>PostData=email=myemail@home.com&name=Joe Bloggs
```

```
HTTPRequest>http://www.someplace.com/someform.html,,POST,PostData,HTMLResponse
```

View System Windows

Displays a tree of all windows currently open in the system. The tree represents a hierarchy showing the handle, class name and title text for each window and their child windows.

