'X'

v3.3.14.2

AutoIt v3 Homepage

Introduction

AutoIt v3 is a freeware BASIC-like scripting language designed for automating the Windows GUI. It uses a combination of simulated keystrokes, mouse movement and window/control manipulation in order to automate tasks in a way not possible or reliable with other languages (e.g. VBScript and SendKeys).

AutoItX is a DLL version of AutoIt v3 that provides a subset of the features of AutoIt via an ActiveX/COM and DLL interface. This means that you can add AutoIt-like features to your favourite scripting and programming languages, e.g. VB, VBScript, Delphi, C, C++, Kixtart, and most other languages that support the use of DLLs.

As AutoItX provides a subset of the features of AutoIt v3 you should read the help file for AutoIt v3 and become familar with the basic concepts, including:

- The AutoIt Window Info Tool
- Windows
- Controls

Help pages on the above topics are duplicated in this help file for reference.

The original version of AutoIt came with two controls: AutoItX (a COM/ActiveX control) and AutoItDLL (a DLL control). In this new version both the COM and DLL versions have been combined into the single AutoItX control which provides both methods of access.

How you use AutoItX depends on the host language you want to use. If you are using something that supports COM object access (like VBScript) then using AutoItX as a COM control is recommended. If you want to use AutoItX from a language such as C then using it as a DLL is simplest.

Throughout this help file examples for COM/ActiveX usage are given using **VBScript**, and standard DLL usage using **C/C++**. These are expected to be the most common ways of using AutoItX (and the two that I am most familiar with). Users of other languages should check out the AutoItX section on the AutoIt Forum for help and advice should you need it.

Software License

AutoIt

Author : Jonathan Bennett and the AutoIt Team
WWW : https://www.autoitscript.com/site/autoit/
Email : support at autoitscript dot com

END-USER LICENSE AGREEMENT FOR THIS SOFTWARE

This End-User License Agreement ("EULA") is a legal agreement between you (either an individual or a single entity) and the mentioned author of this Software for the software product identified above, which includes computer software and may include associated media, printed materials, and "online" or electronic documentation ("SOFTWARE PRODUCT"). By installing, copying, or otherwise using the SOFTWARE PRODUCT, you agree to be bound by the terms of this EULA. If you do not agree to the terms of this EULA, do not install or use the SOFTWARE PRODUCT.

SOFTWARE PRODUCT LICENSE

The SOFTWARE PRODUCT is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The SOFTWARE PRODUCT is licensed, not sold.

The definition of SOFTWARE PRODUCT does not includes any files generated by the SOFTWARE PRODUCT, such as compiled script files in the form of standalone executables.

1. GRANT OF LICENSE

This EULA grants you the following rights:

Installation and Use. You may install and use an unlimited number of copies of the SOFTWARE PRODUCT.

Reproduction and Distribution. You may reproduce and distribute an unlimited number of copies of the SOFTWARE PRODUCT either in whole or in part; each copy should include all copyright and trademark notices, and shall be

accompanied by a copy of this EULA. Copies of the SOFTWARE PRODUCT may be distributed as a standalone product or included with your own product.

Commercial Use. You may use the SOFTWARE PRODUCT for commercial purposes. You may sell for profit and freely distribute scripts and/or compiled scripts that were created with the SOFTWARE PRODUCT.

Reverse engineering. You may not reverse engineer or disassemble the SOFTWARE PRODUCT.

2. COPYRIGHT

All title and copyrights in and to the SOFTWARE PRODUCT (including but not limited to any images, photographs, animations, video, audio, music, text, and "applets" incorporated into the SOFTWARE PRODUCT), the accompanying printed materials, and any copies of the SOFTWARE PRODUCT are owned by the Author of this Software. The SOFTWARE PRODUCT is protected by copyright laws and international treaty provisions. Therefore, you must treat the SOFTWARE PRODUCT like any other copyrighted material.

MISCELLANEOUS

If you acquired this product in the United Kingdom, this EULA is governed by the laws of the United Kingdom. If this product was acquired outside the United Kingdom, then local law may apply.

Should you have any questions concerning this EULA, or if you desire to contact the author of this Software for any reason, please contact him/her at the email address mentioned at the top of this EULA.

LIMITED WARRANTY

1. NO WARRANTIES

The Author of this Software expressly disclaims any warranty for the SOFTWARE PRODUCT. The SOFTWARE PRODUCT and any related documentation is provided "as is" without warranty of any kind, either express or implied, including, without limitation, the implied warranties or merchantability, fitness for a particular purpose, or non-infringement. The entire risk arising out of use or performance of the SOFTWARE PRODUCT remains with you.

2. NO LIABILITY FOR DAMAGES

In no event shall the author of this Software be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use this product, even if the Author of this Software has been advised of the possibility of such damages. Because some states/jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

[END OF LICENSE]

AutoIt Window Information Tool

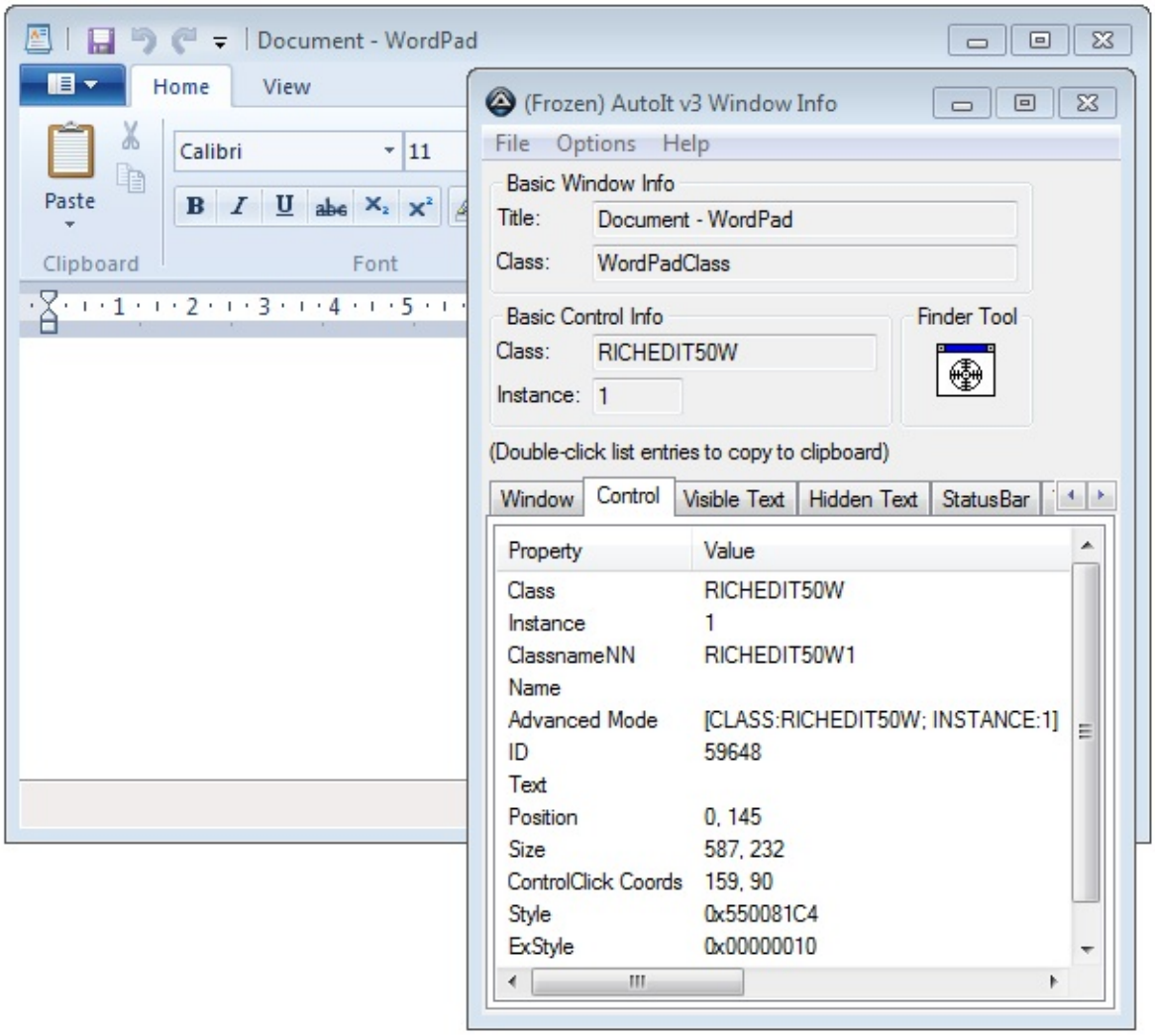
AutoIt v3 comes with a standalone tool called the **AutoIt Window Info Tool** (Program Files\AutoIt3\Au3Info.exe). Au3Info allows you to get information from a specified window that can be used to effectively automate it. Information that can be obtained includes:

- Window titles
- Text on the window (visible and hidden)
- Window size and position
- Contents of the status bar
- Position of the mouse pointer
- Color of the pixels underneath the mouse pointer
- Details of the *Control* underneath the mouse pointer

To use Au3Info just run it (from the command line or Start menu). Au3Info will remain the top most window at all times so that you can read it. Once active move to the window you are interested in and activate it - the contents of Au3Info will change to show the information that is available. With the help of Au3Info you should be automating in no time!

When Au3Info is running you may want to copy text directly from it using **CTRL-C** and then paste it into your script to avoid spelling/case errors. For the tabs that have information in a list view (like the control information shown below) just **double-click** on an entry to copy it to the clipboard. This can be difficult when you want to capture pixel/mouse information as it keeps changing! To help with this you can "freeze" the output of Au3Info by pressing **CTRL-ALT-F**. Press the keys again to "unfreeze".

Here is an example of Au3Info in use with the Windows "WordPad" editor:


Document - WordPad
Home
View
Paste
Clipboard
Calibri
11
Font
(Frozen) AutoIt v3 Window Info
File
Options
Help
Basic Window Info
Title:
Document - WordPad
Class:
WordPadClass
Basic Control Info
Class:
RICHEDIT50W
Instance:
1
Finder Tool
(Double-click list entries to copy to clipboard)
Window
Control
Visible Text
Hidden Text
StatusBar
Property
Value
Class
RICHEDIT50W
Instance
1
ClassnameNN
RICHEDIT50W1
Name
Advanced Mode
[CLASS:RICHEDIT50W; INSTANCE:1]
ID
59648
Text
Position
0, 145
Size
587, 232
ControlClick Coords
159, 90
Style
0x550081C4
ExStyle
0x00000010

Window Titles and Text (Basic)

When automating, most windows can be uniquely identified by their **title** or a combination of their **title & text**. And by using AutoIt Window Info Tool (or even by sight) this information is easy to obtain. The titles of most windows are fairly obvious, for example **Untitled - Notepad** is the title of the notepad.exe editor and in many cases this is enough to automate.

Note: If a blank string "" is given for both *title* and *text* then the currently *Active* window will be used (this is not true in some of the more advanced WinTitleMatchModes)!

Window titles and text are **case sensitive**. You must match the case and punctuation exactly. To avoid problems select the title/text in the Window Info Tool and use **ctrl-c** to copy it and then paste it directly into your script.

Most of AutoIt's window functions have space for a title and text entry, here is the WinWaitActive function. This function pauses execution of the script ***until*** the specified window appears and is active.

```
WinWaitActive "title", ["text"], [timeout]
```

title is the only required parameter for this function, both the ***text*** and ***timeout*** are optional. In some functions the ***text*** parameter is not optional, if you do not wish to specify any text then just use "" (a blank string). A blank string, or nothing at all, in the ***text*** tells AutoIt that any text is valid.

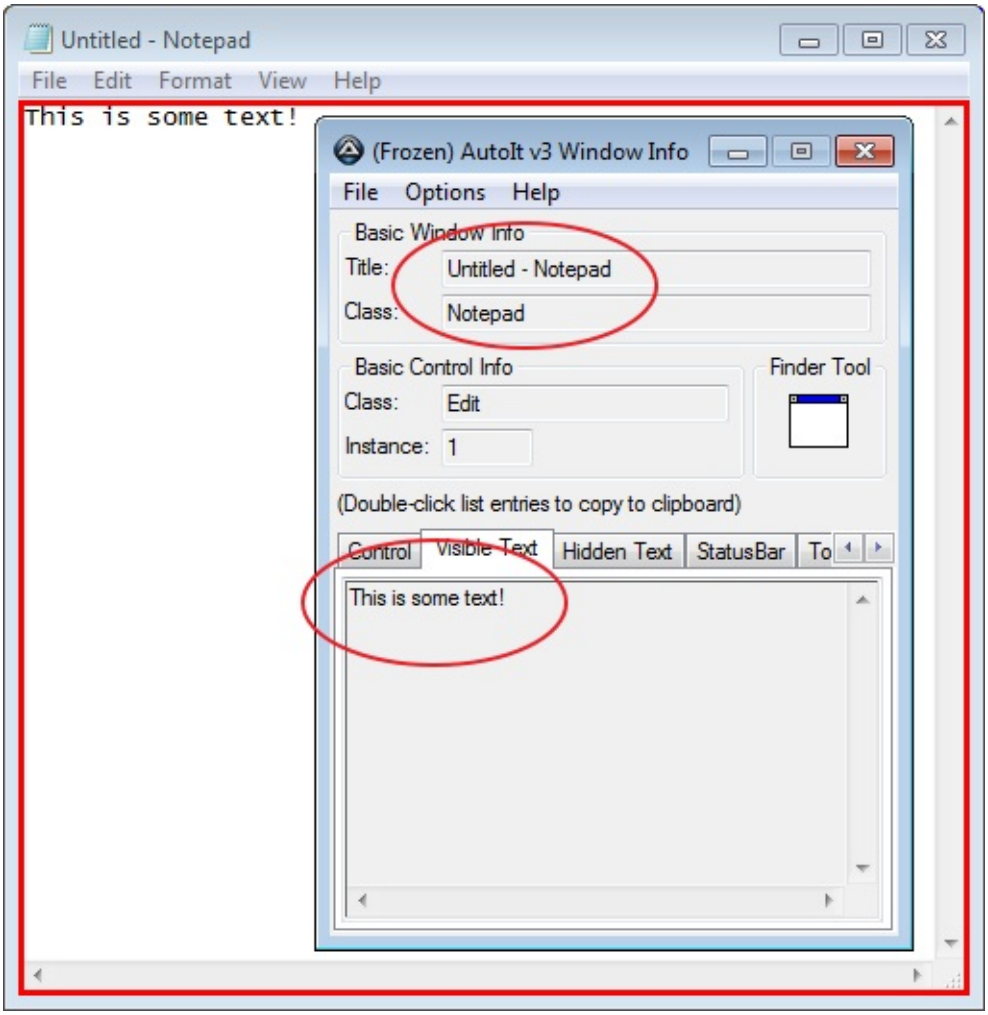
To use the above function with any notepad window both these methods will work:

```
oAutoIt.WinWaitActive "Untitled - Notepad"
```

and

```
oAutoIt.WinWaitActive "Untitled - Notepad", ""
```

If the same notepad window had "This is a line of text" typed into the window, the Window Info Tool would show:



Note that the Window Info Tool has ***seen*** the title and text of the notepad window. Whatever the Window Info Tool can see is what AutoIt can see. Now we have enough information to identify this exact window even if there are lots of other notepad windows open. In this case we use:

```
oAutoIt.WinWaitActive "Untitled - Notepad", "This is s
```

Window Text

The window text consists of all the text that AutoIt can "see". This will usually be things like the contents of edit controls (as above with "This is a line of text") but will also include other text like:

- Button text like &Yes, &No, &Next (the & indicates an underlined letter)
- Dialog text like "Are you sure you wish to continue?"
- Control text
- Misc text - sometimes you don't know what it is :)

The important thing is that you can use the text along with the title to uniquely identify a window to work with.

When you specify the ***text*** parameter in a window function it is treated as a substring. So for the example above if you used the text "is some " you would get a match.

What has been described is the *default* mode that AutoIt operates in, there are a number of more advanced modes to use when things are not as simple as this.

Note: Hidden window can be matched by "title" only if "text" is empty ("").

Window Titles and Text (Advanced)

AutoIt operates in one of three "Window matching" modes. The modes are set with the AutoItSetOption function using the WinTitleMatchMode option.

Mode 1 (default)

Matches partial titles from the start.

In this mode the a window titled **Untitled - Notepad** would be matched by "Untitled - Notepad", "Untitled", "Un", etc.

eg.

```
    oAutoIt.WinWait "Untitled"
```

Mode 2

Matches any substring in the title.

In this mode a window titled **Untitled - Notepad** would be matched by "Untitled - Notepad", "Untitled", "Notepad", "pad", etc.

eg.

```
    oAutoIt.WinWait "Notepad"
```

Mode 3

Exact title match.

In this mode a window titled **Untitled - Notepad** would only be matched by "Untitled - Notepad"

Mode 4 (Kept for backward compatibility)

Advanced mode

Must be replaced with Advanced Window Descriptions which does not need any mode to be set.

Mode -1 to -4

Force lower case match according to other type of match.

Advanced Window Descriptions

A special description can be used as the window **title** parameter. This description can be used to identify a window by the following *properties*:

- **TITLE -** Window title
- **CLASS -** The internal window classname
- **REGEXPTITLE -** Window title using a regular expression
- **REGEXPCLASS -** Window classname using a regular expression
- **LAST -** Last window used in a previous AutoIt command
- **ACTIVE -** Currently active window
- **X** \ **Y** \ **W** \ **H** - The position and size of a window
- **INSTANCE -** The 1-based instance when all given properties match
- **HANDLE** - The handle address as returned by a method like WinGetHandle

One or more properties are used in the *title* parameter of a window command in the format:

```
[PROPERTY1:Value1; PROPERTY2:Value2]
```

Note : if a Value must contain a ";" it must be doubled.

e.g. Wait a window of classname "Notepad"

```
oAutoIt.WinWaitActive "[CLASS:Notepad]", ""
```

e.g. Close the currently active window

```
oAutoIt.WinClose "[ACTIVE]", ""
```

e.g. Wait for the 2nd instance of a window with title "My Window" **and** classname "My Class"

```
oAutoIt.WinWaitActive "[TITLE:My Window; CLASS:My Clas
```

Note : if a Value must contain a ";" it must be doubled.

Controls

One of the best new features with AutoIt v3 is the ability to work directly with certain types of Window *Controls*. Almost everything you see on a window is a control of some kind: buttons, listboxes, edit fields, static text are all controls. In fact Notepad is just one big "Edit" control! Because AutoIt works directly with a control they provide a more reliable way to automate than just sending keystrokes.

***Note**: AutoIt only works with standard Microsoft controls - some applications write their own custom controls which may look like a standard MS control but may resist automation. Experiment!*

Using the AutoIt Window Info Tool you can move your mouse around the window you are interested in and you will be given information of the control that is currently under your mouse. Information on controls is given in a number of ways, these are:

- Control ID
- ClassNameNN
- Text

Whenever you see a Control...() command expecting a **controlID** parameter (most of them) you can use any **one** of these methods. The method you choose will vary by personal preference and the information you are able to retrieve from the AutoIt Window Info Tool. In general, the best method to use is the **Control ID**, with the other methods being used when a Control ID is not available or is not unique (usually with static text each piece of text actually has the same Control ID so you will need to use one of the other methods to work with those controls).

Control ID

The Control ID is the internal numeric identifier that windows gives to each control. It is generally the best method of indentifying controls. In addition to the AutoIt Window Info Tool, other applications such as screenreaders for the blind and Microsoft tools/APIs may allow you to get this Control ID.

ClassNameNN

Each Microsoft standard control has a "classname" such as "button" or "edit". In AutoIt this is combined with the "instance" of that control to give the "ClassNameNN" method. If you have a simple dialog with a few buttons on it they will generally be referred to as "Button1", "Button2", "Button3", etc.

This method is useful when the Control ID is not applicable (for static text or custom controls).

Text

You will notice that Au3Info gives you the text it can see on a control, for example on a **Next** button you might see the text **&Next** - the & indicates that the next letter is underlined and is common when working with menus and controls. You can use this text to identify a control in place of the "ClassNameNN" if you like - however if many controls have the same text you will run into problems.

Look under the contents for **Method Reference \ Window Management \ Controls** for a list of the functions that work with controls.

Using the COM Interface

Before you can use the COM interface to AutoItX it needs to be "registered" (This is done automatically when you install the full version of AutoIt but you may need to do it manually if you are using AutoItX separately).

To register the COM interface:

1. Open a command prompt
2. Change directory (using CD) to the directory that contains **AutoItX3.dll**
3. Type **regsvr32.exe AutoItX3.dll** and press enter

The name of the AutoItX control is **AutoItX3.Control**

Here is an example of calling a the Run method of the control from VBScript:

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Control")
oAutoIt.Run("notepad.exe")
```

Method Reference

Below is a complete list of the methods available in AutoItX. Click on a method name for a detailed description.

Method	Description
AutoItSetOption	Changes the operation of various AutoIt functions/parameters
ClipGet	Retrieves text from the clipboard
ClipPut	Writes text to the clipboard
ControlClick	Sends a mouse click command to a given control
ControlCommand	Sends a command to a control
ControlDisable	Disables or "grays-out" a control
ControlEnable	Enables a "grayed-out" control
ControlFocus	Sets input focus to a given control on a window
ControlGetFocus	Returns the ControlRef# of the control that has keyboard focus within a specified window
ControlGetHandle	Retrieves the internal handle of a control
ControlGetPosHeight	Retrieves the position and size of a control relative to it's window
ControlGetPosWidth	Retrieves the position and size of a control relative to it's window
ControlGetPosX	Retrieves the position and size of a control relative to it's window
ControlGetPosY	Retrieves the position and size of a control relative to it's window
ControlGetText	Retrieves text from a control
ControlHide	Hides a control
ControlListView	Sends a command to a ListView32 control

ControlMove	Moves a control within a window
ControlSend	Sends a string of characters to a control
ControlSetText	Sets text of a control
ControlShow	Shows a control that was hidden
ControlTreeView	Sends a command to a TreeView32 control
DriveMapAdd	Maps a network drive
DriveMapDel	Disconnects a network drive
DriveMapGet	Retreives the details of a mapped drive
IsAdmin	Checks if the current user has administrator privileges
MouseClick	Perform a mouse click operation
MouseClickDrag	Perform a mouse click and drag operation
MouseDown	Perform a mouse down event at the current mouse position
MouseGetCursor	Returns a cursor ID Number of the current Mouse Cursor
MouseGetPosX	Retrieves the current X position of the mouse cursor
MouseGetPosY	Retrieves the current Y position of the mouse cursor
MouseMove	Moves the mouse pointer
MouseUp	Perform a mouse up event at the current mouse position
MouseWheel	Moves the mouse wheel up or down. XP ONLY
PixelChecksum	Generates a checksum for a region of pixels
PixelGetColor	Returns a pixel color according to x,y pixel coordinates
PixelSearch	Searches a rectangle of pixels for the pixel color provided
ProcessClose	Terminates a named process
ProcessExists	Checks to see if a specified process exists

ProcessSetPriority	Changes the priority of a process
ProcessWait	Pauses script execution until a given process exists
ProcessWaitClose	Pauses script execution until a given process does not exist
Run	Runs an external program
RunAs	Runs an external program
RunAsWait	Runs an external program
RunWait	Runs an external program and pauses script execution until the program finishes
Send	Sends simulated keystrokes to the active window
Shutdown	Shuts down the system
Sleep	Pause script execution
StatusbarGetText	Retrieves the text from a standard status bar control
ToolTip	Creates a tooltip anywhere on the screen
WinActivate	Activates (gives focus to) a window
WinActive	Checks to see if a specified window exists and is currently active
WinClose	Closes a window
WinExists	Checks to see if a specified window exists
WinGetCaretPosX	Returns the coordinates of the caret in the foreground window
WinGetCaretPosY	Returns the coordinates of the caret in the foreground window
WinGetClassList	Retrieves the classes from a window
WinGetClientSizeHeight	Retrieves the size of a given window's client area
WinGetClientSizeWidth	Retrieves the size of a given window's client area
WinGetHandle	Retrieves the internal handle of a window
WinGetPosHeight	Retrieves the position and size of a given window
WinGetPosWidth	Retrieves the position and size of a given window

WinGetPosX	Retrieves the position and size of a given window
WinGetPosY	Retrieves the position and size of a given window
WinGetProcess	Retrieves the Process ID (PID) associated with a window
WinGetState	Retrieves the state of a given window
WinGetText	Retrieves the text from a window
WinGetTitle	Retrieves the full title from a window
WinKill	Forces a window to close
WinList	Retrieves a list of windows
WinMenuSelectItem	Invokes a menu item of a window
WinMinimizeAll	Minimizes all windows
WinMinimizeAllUndo	Undoes a previous WinMinimizeAll function
WinMove	Moves and/or resizes a window
WinSetOnTop	Change a window's "Always On Top" attribute
WinSetState	Shows, hides, minimizes, maximizes, or restores a window
WinSetTitle	Changes the title of a window
WinSetTrans	Sets the transparency of a window
WinWait	Pauses execution of the script until the requested window exists
WinWaitActive	Pauses execution of the script until the requested window is active
WinWaitClose	Pauses execution of the script until the requested window does not exist
WinWaitNotActive	Pauses execution of the script until the requested window is not active

Environment methods Reference

Below is a complete list of the methods available in AutoItX. Click on a method name for a detailed description.

Method	Description
ClipGet	Retrieves text from the clipboard
ClipPut	Writes text to the clipboard

ClipGet

Retrieves text from the clipboard

```
ClipGet
```

Return Value

Success:	Returns a string containing the text on the clipboard.
Failure:	Sets oAutoIt.error to 1 if clipboard is empty or contains a non-text entry.

Related

ClipPut

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

text = oAutoIt.ClipGet()
WScript.Echo "Clipboard contains:" & text
```

ClipPut

Writes text to the clipboard

```
ClipPut "value"
```

Parameters

value	The text to write to the clipboard.

Return Value

Success: Returns 1.

Failure: Returns 0.

Remarks

Any existing clipboard contents are overwritten.

Related

ClipGet

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.ClipPut "I am copied to the clipboard"
```

File, Directory and Disk methods Reference

Below is a complete list of the methods available in AutoItX. Click on a method name for a detailed description.

Method	Description
DriveMapAdd	Maps a network drive
DriveMapDel	Disconnects a network drive
DriveMapGet	Retreives the details of a mapped drive

DriveMapAdd

Maps a network drive

```
DriveMapAdd "device", "remote share" [, flags [,
"user" [, "password"]]]
```

Parameters

device	The device to map, for example "O:" or "LPT1:". If you pass a blank string for this parameter a connection is made but not mapped to a specific drive. If you specify "*" an unused drive letter will be automatically selected.
remote share	The remote share to connect to in the form "\\server\share".
flags	**Optional:** A combination of the following: 0 = default 1 = Persistant mapping 8 = Show authentication dialog if required
user	**Optional:** The username to use to connect. In the form "username" or "domain\username".
password	**Optional:** The password to use to connect.

Return Value

Success:	Returns 1. (See Remarks)
Failure:	Returns 0 if a new mapping could not be created and sets oAutoIt.error (see below). (See Remarks)

Remarks

When the function fails (returns 0) oAutoIt.error contains extended information:

1 = Undefined / Other error
2 = Access to the remote share was denied
3 = The device is already assigned
4 = Invalid device name
5 = Invalid remote share
6 = Invalid password

Note: When using "*" for the device parameter the drive letter selected will be returned instead of 1 or 0, e.g. "U:". If there was an error using "*" then a blank string "" will be returned.

Related

DriveMapDel, DriveMapGet

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

' Map X drive to \\myserver\stuff using current use
r
oAutoIt.DriveMapAdd "X:", "\\myserver\stuff"

' Map X drive to \\myserver2\stuff2 using the user
"jon" from "domainx" with password "tickle"
oAutoIt.Drivemap "X:", "\\myserver2\stuff2", 0, "do
mainx\jon", "tickle"
```

DriveMapDel

Disconnects a network drive

```
DriveMapDel "device"
```

Parameters

drive	The device to disconnect, e.g. "O:" or "LPT1:".

Return Value

Success: Returns 1.

Failure: Returns 0 if the disconnection was unsuccessful.

Remarks

If a connection has no drive letter mapped you may use the connection name to disconnect, e.g. \\server\share

Related

DriveMapAdd, DriveMapGet

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

' Map X drive to \\myserver\stuff using current use
r
oAutoIt.DriveMapAdd "X:", "\\myserver\stuff"

' Disconnect
oAutoIt.DriveMapDel "X:"
```

DriveMapGet

Retreives the details of a mapped drive

```
DriveMapGet"device"
```

Parameters

device	The device (drive or printer) letter to query. Eg. "O:" or "LPT1:"

Return Value

Success:	Returns details of the mapping, e.g. \\server\share
Failure:	Returns a blank string "" and sets oAutoIt.error to 1.

Related

DriveMapAdd, DriveMapDel

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

' Map X drive to \\myserver\stuff using current use
r
oAutoIt.DriveMapAdd "X:", "\\myserver\stuff"

' Get details of the mapping
WScript.Echo "Drive X: is mapped to " & DriveMapGet
("X:")
```

Graphic methods Reference

Below is a complete list of the methods available in AutoItX. Click on a method name for a detailed description.

Method	Description
PixelChecksum	Generates a checksum for a region of pixels
PixelGetColor	Returns a pixel color according to x,y pixel coordinates
PixelSearch	Searches a rectangle of pixels for the pixel color provided

PixelChecksum

Generates a checksum for a region of pixels

```
PixelChecksum left, top, right, bottom [, step]
```

Parameters

left	left coordinate of rectangle.
top	top coordinate of rectangle.
right	right coordinate of rectangle.
bottom	bottom coordinate of rectangle.
step	**Optional:** Instead of checksumming each pixel use a value larger than 1 to skip pixels (for speed). E.g. A value of 2 will only check every other pixel. Default is 1.

Return Value

Returns the checksum value of the region.

Remarks

Performing a checksum of a region is very time consuming, so use the smallest region you are able to reduce CPU load. On some machines a checksum of the whole screen could take many seconds!

A checksum only allows you to see if "something" has changed in a region - it does not tell you exactly what has changed.

When using a step value greater than 1 you must bear in mind that the checksumming becomes less reliable for small changes as not every pixel is checked.

Related

PixelChecksum, PixelCoordMode (Option), PixelGetColor, PixelSearch

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

' Get initial checksum
checksum = oAutoIt.PixelChecksum(0,0, 50,50)
```

PixelGetColor

Returns a pixel color according to x,y pixel coordinates

```
PixelGetColor x , y
```

Parameters

x	x coordinate of pixel.
y	y coordinate of pixel.

Return Value

Success:	Returns **decimal** value of pixel's color.
Failure:	Returns -1 if invalid coordinates.

Related

MouseGetPosX, MouseGetPosY, PixelCoordMode (Option), PixelSearch

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
var = oAutoIt.PixelGetColor( 10 , 100 )
WScript.Echo "The color is" & var
```

PixelSearch

Searches a rectangle of pixels for the pixel color provided

```
PixelSearch left, top, right, bottom, color [,
shade-variation] [, step]]
```

Parameters

left	left coordinate of rectangle.
top	top coordinate of rectangle.
right	right coordinate of rectangle.
bottom	bottom coordinate of rectangle.
color	color value of pixel to find (in decimal or hex).
shade-variation	**Optional:** A number between 0 and 255 to indicate the allowed number of shades of variation of the red, green, and blue components of the color. Default is 0 (exact match).
step	**Optional:** Instead of searching each pixel use a value larger than 1 to skip pixels (for speed). E.g. A value of 2 will only check every other pixel. Default is 1.

Return Value

Success:	Returns a 2 element array containing the pixel's coordinates
Failure:	Sets oAutoIt.error to 1 if color is not found.

Remarks

The search is performed top-to-bottom, left-to-right, and the first match is returned.

Performing a search of a region can be time consuming, so use the smallest region you are able to reduce CPU load.

Related

PixelChecksum, PixelCoordMode (Option), PixelGetColor

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

value = oAutoIt.PixelSearch(0,0, 100, 100, 0)
If oAutoIt.error = 1 Then
  WScript.Echo "Color not found"
Else
  WScript.Echo "Color found at: " & value(0) & ","
& value(1)
End If
```

Keyboard Control methods Reference

Below is a complete list of the methods available in AutoItX. Click on a method name for a detailed description.

Method	Description
Send	Sends simulated keystrokes to the active window

Send

Sends simulated keystrokes to the active window

```
Send "keys" [, flag]
```

Parameters

keys	The sequence of keys to send.
flag	**Optional:** Changes how "keys" is processed: flag = 0 (default), Text contains special characters like + and ! to indicate SHIFT and ALT key presses. flag = 1, keys are sent raw.

Return Value

None.

Remarks

See the Appendix for some tips.

The "Send" command syntax is similar to that of ScriptIt and the Visual Basic "SendKeys" command. Characters are sent as written with the exception of the following characters:

'!'
This tells AutoIt to send an ALT keystroke, therefore Send("This is text!a") would send the keys "This is text" and then press "ALT+a".

N.B. Some programs are very choosy about capital letters and ALT keys, i.e. "!A" is different to "!a". The first says ALT+SHIFT+A, the second is ALT+a. If in doubt, use lowercase!

'+'
This tells AutoIt to send a SHIFT keystroke, therefore Send("Hell+o") would send the text "HellO". Send("!+a") would send "ALT+SHIFT+a".

'^'
This tells AutoIt to send a CONTROL keystroke, therefore Send("^!a") would send "CTRL+ALT+a".

N.B. Some programs are very choosy about capital letters and CTRL keys, i.e. "^A" is different to "^a". The first says CTRL+SHIFT+A, the second is CTRL+a. If in doubt, use lowercase!

'#'
The hash now sends a Windows keystroke; therefore, Send("#r") would send Win+r which launches the Run dialog box.

You can set SendCapslockMode to make CAPS LOCK disabled at the start of a Send operation and restored upon completion.
However, if a user is holding down the Shift key when a Send function begins,

text may be sent in uppercase.
One workaround is to Send("{SHIFTDOWN}{SHIFTUP}") before the other Send operations.

Certain special keys can be sent and should be enclosed in braces:

N.B. Windows does not allow the simulation of the "CTRL-ALT-DEL" combination!

Send Command (if zero flag)	**Resulting Keypress**
{!}	!
{#}	#
{+}	+
{^}	^
{{}	{
{}}	}
{SPACE}	SPACE
{ENTER}	ENTER key on the main keyboard
{ALT}	ALT
{BACKSPACE} or {BS}	BACKSPACE
{DELETE} or {DEL}	DELETE
{UP}	Cursor up
{DOWN}	Cursor down
{LEFT}	Cursor left
{RIGHT}	Cursor right
{HOME}	HOME
{END}	END
{ESCAPE} or {ESC}	ESCAPE
{INSERT} or {INS}	INS

{PGUP}	PageUp
{PGDN}	PageDown
{F1} - {F12}	Function keys
{TAB}	TAB
{PRINTSCREEN}	Print Screen key
{LWIN}	Left Windows key
{RWIN}	Right Windows key
{NUMLOCK on}	NUMLOCK (on/off/toggle)
{CAPSLOCK off}	CAPSLOCK (on/off/toggle)
{SCROLLLOCK toggle}	SCROLLLOCK (on/off/toggle)
{BREAK}	for Ctrl+Break processing
{PAUSE}	PAUSE
{NUMPAD0} - {NUMPAD9}	Numpad digits
{NUMPADMULT}	Numpad Multiply
{NUMPADADD}	Numpad Add
{NUMPADSUB}	Numpad Subtract
{NUMPADDIV}	Numpad Divide
{NUMPADDOT}	Numpad period
{NUMPADENTER}	Enter key on the numpad
{APPSKEY}	Windows App key
{LALT}	Left ALT key
{RALT}	Right ALT key
{LCTRL}	Left CTRL key
{RCTRL}	Right CTRL key
{LSHIFT}	Left Shift key
{RSHIFT}	Right Shift key
{SLEEP}	Computer SLEEP key

{ALTDOWN}	Holds the ALT key down until {ALTUP} is sent
{SHIFTDOWN}	Holds the SHIFT key down until {SHIFTUP} is sent
{CTRLDOWN}	Holds the CTRL key down until {CTRLUP} is sent
{LWINDOWN}	Holds the left Windows key down until {LWINUP} is sent
{RWINDOWN}	Holds the right Windows key down until {RWINUP} is sent
{ASC nnnn}	Send the ALT+nnnn key combination
{BROWSER_BACK}	XP Only: Select the browser "back" button
{BROWSER_FORWARD}	XP Only: Select the browser "forward" button
{BROWSER_REFRESH}	XP Only: Select the browser "refresh" button
{BROWSER_STOP}	XP Only: Select the browser "stop" button
{BROWSER_SEARCH}	XP Only: Select the browser "search" button
{BROWSER_FAVORITES}	XP Only: Select the browser "favorites" button
{BROWSER_HOME}	XP Only: Launch the browser and go to the home page
{VOLUME_MUTE}	XP Only: Mute the volume
{VOLUME_DOWN}	XP Only: Reduce the volume
{VOLUME_UP}	XP Only: Increase the volume
{MEDIA_NEXT}	XP Only: Select next track in media player
{MEDIA_PREV}	XP Only: Select previous track in media player
{MEDIA_STOP}	XP Only: Stop media player
{MEDIA_PLAY_PAUSE}	XP Only: Play/pause media player
{LAUNCH_MAIL}	XP Only: Launch the email application
{LAUNCH_MEDIA}	XP Only: Launch media player
{LAUNCH_APP1}	XP Only: Launch user app1
{LAUNCH_APP2}	XP Only: Launch user app2

To send the ASCII value A (same as pressing ALT+065 on the numeric keypad)

```
Send("{ASC 65}")
```

Single keys can also be repeated, e.g.

```
Send("{DEL 4}")    ;Presses the DEL key 4 times
Send("{S 30}") ;Sends 30 'S' characters
Send("+{TAB 4}) ;Presses SHIFT+TAB 4 times
```

To hold a key down (generally only useful for games)

```
Send("{a down}")    ;Holds the A key down
Send("{a up}") ;Releases the A key
```

To set the state of the capslock, numlock and scrolllock keys

```
Send("{NumLock on}") ;Turns the NumLock key on
Send("{CapsLock off}") ;Turns the CapsLock key off
Send("{ScrollLock toggle}") ;Toggles the state of ScrollLock
```

If you with to use a variable for the count, try

```
$n = 4
Send("+{TAB " & $n & "}")
```

If you wish to send the ASCII value A four times, then try

```
$x = Chr(65)
Send("{" & $x & " 4}")
```

Most laptop computer keyboards have a special Fn key. This key cannot be simulated.

Note, by setting the flag parameter to 1 the above "special" processing will be disabled. This is useful when you want to send some text copied from a variable and you want the text sent exactly as written.

For example, open Folder Options (in the control panel) and try the following:

Send("{TAB}")	Navigate to next control (button, checkbox, etc)

Send("+{TAB}")	Navigate to previous control.
Send("^{TAB}")	Navigate to next WindowTab (on a Tabbed dialog window)
Send("^+{TAB}")	Navigate to previous WindowTab.
Send("{SPACE}")	Can be used to toggle a checkbox or click a button.
Send("{+}")	Usually checks a checkbox (if it's a "real" checkbox.)
Send("{-}")	Usually unchecks a checkbox.
Send("{NumPadMult}")	Recursively expands folders in a SysTreeView32.

Use Alt-key combos to access menu items. Also, open Notepad and try the following:
Send("!f") Send Alt+f, the access key for Notepad's file menu. Try other letters!

Send("{DOWN}")	Move down a menu.
Send("{UP}")	Move up a menu.
Send("{LEFT}")	Move leftward to new menu or expand a submenu.
Send("{RIGHT}")	Move rightward to new menu or collapse a submenu.

Related

ControlSend, SendAttachmode (Option), SendKeyDelay (Option), SendKeyDownDelay (Option)

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.Send "#r"
oAutoIt.WinWaitActive "Run"
oAutoIt.Send "notepad.exe{Enter}"
oAutoIt.WinWaitActive "Untitled -"
oAutoIt.Send "Today's time/date is {F5}"
```

Message Boxes and Dialogs methods Reference

Below is a complete list of the methods available in AutoItX. Click on a method name for a detailed description.

Method	Description
ToolTip	Creates a tooltip anywhere on the screen

ToolTip

Creates a tooltip anywhere on the screen

```
ToolTip "text" [, x, y]
```

Parameters

text	The text of the tooltip. (An empty string clears a displaying tooltip)
x, y	The x,y position of the tooltip.

Return Value

None.

Remarks

If the x and y coordinates are omitted the, tip is placed near the mouse cursor. If the coords would cause the tooltip to run off screen, it is repositioned to visible. Tooltip appears until it is cleared, until script terminates, or sometimes until it is clicked upon. You may use a linefeed character to create multi-line tooltips.

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

' This will create a tooltip in the upper left of t
he screen
oAutoIt.ToolTip "This is a tooltip", 0, 0
oAutoIt.Sleep 2000  ' Sleep to give tooltip time to
 display
```

Miscelaneous methods Reference

Below is a complete list of the methods available in AutoItX. Click on a method name for a detailed description.

Method	Description
AutoItSetOption	Changes the operation of various AutoIt functions/parameters
IsAdmin	Checks if the current user has administrator privileges

AutoItSetOption

Changes the operation of various AutoIt functions/parameters

```
AutoItSetOption "option", param
```

Parameters

option	The option to change. See Remarks.
param	The parameter (varies by option). See Remarks.

Return Value

Returns the value of the previous setting.

Remarks

You may use Opt as an alternative to AutoItSetOption.

AutoIt will halt with an error message if the requested option is unknown.
Options are as follows:

Option	**Param**
CaretCoordMode	Sets the way coords are used in the caret functions, either absolute coords or coords relative to the current active window: 0 = relative coords to the active window 1 = absolute screen coordinates (default) 2 = relative coords to the client area of the active window
MouseClickDelay	Alters the length of the brief pause in between mouse clicks. Time in milliseconds to pause (default=10).
MouseClickDownDelay	Alters the length a click is held down before release. Time in milliseconds to pause (default=10).
MouseClickDragDelay	Alters the length of the brief pause at the start and end of a mouse drag operation. Time in milliseconds to pause (default=250).
MouseCoordMode	Sets the way coords are used in the mouse functions, either absolute coords or coords relative to the current active window: 0 = relative coords to the active window 1 = absolute screen coordinates (default) 2 = relative coords to the client area of the active window
PixelCoordMode	Sets the way coords are used in the pixel functions, either absolute coords or coords relative to the current active window: 0 = relative coords to the active window 1 = absolute screen coordinates (default) 2 = relative coords to the client area of the active

	window
SendAttachMode	Specifies if AutoIt attaches input threads when using then Send() function. When not attaching (default mode=0) detecting the state of capslock/scrolllock and numlock can be unreliable under NT4. However, when you specify attach mode=1 the Send("{... down/up}") syntax will not work and there may be problems with sending keys to "hung" windows. ControlSend() ALWAYS attaches and is not affected by this mode. 0 = don't attach (default) 1 = attach
SendCapslockMode	Specifies if AutoIt should store the state of capslock before a Send function and restore it afterwards. 0 = don't store/restore 1 = store and restore (default)
SendKeyDelay	Alters the the length of the brief pause in between sent keystrokes. Time in milliseconds to pause (default=5). Sometimes a value of 0 does not work; use 1 instead.
SendKeyDownDelay	Alters the length of time a key is held down before released during a keystroke. For applications that take a while to register keypresses (and many games) you may need to raise this value from the default. Time in milliseconds to pause (default=1).
WinDetectHiddenText	Specifies if hidden window text can be "seen" by the window matching functions. 0 = Do not detect hidden text (default) 1 = Detect hidden text
WinSearchChildren	Allows the window search routines to search child windows as well as top-level windows. 0 = Only search top-level windows (default) 1 = Search top-level and child windows
	Alters the method that is used to match window text during search operations. 1 = Complete / Slow mode (default)

WinTextMatchMode	2 = Quick mode In quick mode AutoIt can usually only "see" dialog text, button text and the captions of some controls. In the default mode much more text can be seen (for instance the contents of the Notepad window). If you are having performance problems when performing many window searches then changing to the "quick" mode may help.
WinTitleMatchMode	Alters the method that is used to match window titles during search operations. 1 = Match the title from the start (default) 2 = Match any substring in the title 3 = Exact title match 4 = Advanced mode, see Window Titles & Text (Advanced)
WinWaitDelay	Alters how long a script should briefly pause after a successful window-related operation. Time in milliseconds to pause (default=250).

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oldvalue = oAutoIt.Opt("SendAttachMode", 0)
```

IsAdmin

Checks if the current user has administrator privileges

```
IsAdmin
```

Return Value

Success:	Returns 1 if the current user has administrator privileges.
Failure:	Returns 0 if user lacks admin privileges.

Remarks

This function will always return 1 under Window 95/98/Me.

Related

RunAs, RunAsWait

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Control")

If oAutoIt.IsAdmin Then WScript.Echo "Admin rights detected"
```

Mouse Control methods Reference

Below is a complete list of the methods available in AutoItX. Click on a method name for a detailed description.

Method	Description
MouseClick	Perform a mouse click operation
MouseClickDrag	Perform a mouse click and drag operation
MouseDown	Perform a mouse down event at the current mouse position
MouseGetCursor	Returns a cursor ID Number of the current Mouse Cursor
MouseGetPosX	Retrieves the current X position of the mouse cursor
MouseGetPosY	Retrieves the current Y position of the mouse cursor
MouseMove	Moves the mouse pointer
MouseUp	Perform a mouse up event at the current mouse position
MouseWheel	Moves the mouse wheel up or down. XP ONLY

MouseClick

Perform a mouse click operation

```
MouseClick "button" [, x, y [, clicks [, speed ]]]
```

Parameters

button	The button to click: "left", "right", "middle", "main", "menu", "primary", "secondary".
x, y	**Optional:** The x/y coordinates to move the mouse to. If no x and y coords are given, the current position is used.
clicks	**Optional:** The number of times to click the mouse. Default is 1.
speed	**Optional:** the speed to move the mouse in the range 1 (fastest) to 100 (slowest). A speed of 0 will move the mouse instantly. Default speed is 10.

Return Value

None.

Remarks

If the button is an empty string, the left button will be clicked.

If the button is not in the list, then oAutoIt.error will be set to 1.

If the user has swapped the left and right mouse buttons in the control panel, then the behaviour of the buttons is different. "Left" and "right" always click those buttons, whether the buttons are swapped or not. The "primary" or "main" button will be the main click, whether or not the buttons are swapped. The "secondary" or "menu" buttons will usually bring up the context menu, whether the buttons are swapped or not.

Button	**Normal**	**Swapped**
""	Left	Left
"left"	Left	Left
"middle"	Middle	Middle
"right"	Right	Right
"primary"	Left	Right
"main"	Left	Right
"secondary"	Right	Left
"menu"	Right	Left

Related

MouseClickDrag, MouseCoordMode (Option), MouseGetPosX, MouseGetPosY, MouseMove

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

' Double click at the current mouse pos
oAutoIt.MouseClick "left"
oAutoIt.MouseClick "left"

' Double click at 0,500
oAutoIt.MouseClick "left", 0, 500, 2
```

MouseClickDrag

Perform a mouse click and drag operation

```
MouseClickDrag "button", x1, y1, x2, y2 [, speed]
```

Parameters

button	The button to click: "left", "right", "middle", "main", "menu", "primary", "secondary".
x1, y1	The x/y coords to start the drag operation from.
x2, y2	The x/y coords to start the drag operation to.
speed	**Optional:** the speed to move the mouse in the range 1 (fastest) to 100 (slowest). A speed of 0 will move the mouse instantly. Default speed is 10.

Return Value

None.

Remarks

If the button is an empty string, the left button will be clicked.

If the button is not in the list, then oAutoIt.error will be set to 1.

If the user has swapped the left and right mouse buttons in the control panel, then the behaviour of the buttons is different. "Left" and "right" always click those buttons, whether the buttons are swapped or not. The "primary" or "main" button will be the main click, whether or not the buttons are swapped. The "secondary" or "menu" buttons will usually bring up the context menu, whether the buttons are swapped or not. See the table in MouseClick for more explaination

Related

MouseClick, MouseCoordMode (Option), MouseGetPosX, MouseGetPosY, MouseMove

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

' Left click drag from 0,200 to 600, 700
oAutoIt.MouseClickDrag "left", 0, 200, 600, 700
```

MouseDown

Perform a mouse down event at the current mouse position

```
MouseDown "button"
```

Parameters

button	The button to click: "left", "right", "middle", "main", "menu", "primary", "secondary".

Return Value

None.

Remarks

MouseClick.
Use responsibly: For every MouseDown there should eventually be a corresponding MouseUp event.

Related

MouseClick, MouseClickDrag, MouseCoordMode (Option), MouseGetPosX, MouseGetPosY, MouseMove, MouseUp

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

oAutoIt.MouseDown "left"
oAutoIt.Sleep 100
oAutoIt.MouseUp "left"
```

MouseGetCursor

Returns a cursor ID Number of the current Mouse Cursor

```
MouseGetCursor
```

Return Value

Returns a cursor ID Number:
0 = UNKNOWN (this includes pointing and grabbing hand icons)
1 = APPSTARTING
2 = ARROW
3 = CROSS
4 = HELP
5 = IBEAM
6 = ICON
7 = NO
8 = SIZE
9 = SIZEALL
10 = SIZENESW
11 = SIZENS
12 = SIZENWSE
13 = SIZEWE
14 = UPARROW
15 = WAIT

Related

MouseGetPosX, MouseGetPosY

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

oAutoIt.Sleep 2000  'allow time to move mouse befor
e reporting ID
cursor = oAutoIt.MouseGetCursor()
WScript.Echo "ID = " & cursor
```

MouseGetPosX

Retrieves the current X position of the mouse cursor

```
MouseGetPosX
```

Return Value

Returns the current X position of the mouse cursor.

Remarks

See MouseCoordMode for relative/absolute position settings. If relative positioning, numbers may be negative.

Related

MouseClick, MouseClickDrag, MouseCoordMode (Option), MouseGetPosY, MouseMove

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

posx = oAutoIt.MouseGetPosX()
posy  = oAutoIt.MouseGetPosX()

WScript.Echo "Mouse x,y:" & posx & "," & posy
```

MouseGetPosY

Retrieves the current Y position of the mouse cursor

```
MouseGetPosY
```

Return Value

Returns the current Y position of the mouse cursor.

Remarks

See MouseCoordMode for relative/absolute position settings. If relative positioning, numbers may be negative.

Related

MouseClick, MouseClickDrag, MouseCoordMode (Option), MouseGetPosX, MouseMove

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

posx = oAutoIt.MouseGetPosX()
posy  = oAutoIt.MouseGetPosX()

WScript.Echo "Mouse x,y:" & posx & "," & posy
```

MouseMove

Moves the mouse pointer

```
MouseMove x, y [, speed]
```

Parameters

x	The screen x coordinate to move the mouse to.
y	The screen y coordinate to move the mouse to.
speed	**Optional:** the speed to move the mouse in the range 1 (fastest) to 100 (slowest). A speed of 0 will move the mouse instantly. Default speed is 10.

Return Value

User mouse movement is hindered during a non-instantaneous MouseMove operation.
If MouseCoordMode is relative positioning, numbers may be negative.

Related

MouseClick, MouseClickDrag, MouseCoordMode (Option), MouseGetPosX, MouseGetPosY

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

oAutoIt.MouseMove 10, 100
oAutoIt.MouseMove 700, 700, 0
```

MouseUp

Perform a mouse up event at the current mouse position

```
MouseUp "button"
```

Parameters

button	The button to click: "left", "right", "middle", "main", "menu", "primary", "secondary".

Return Value

None.

Remarks

MouseClick.
Use responsibly: For every MouseDown there should eventually be a corresponding MouseUp event.

Related

MouseClick, MouseClickDrag, MouseCoordMode (Option), MouseDown, MouseGetPosX, MouseGetPosY, MouseMove

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

oAutoIt.MouseDown "left"
oAutoIt.Sleep 100
oAutoIt.MouseUp "left"
```

MouseWheel

Moves the mouse wheel up or down. XP ONLY

```
MouseWheel "direction" [, clicks]
```

Parameters

direction	"up" or "down"
clicks	**Optional:** The number of times to move the wheel. Default is 1.

Return Value

None.

Remarks

If the direction is not recognized, oAutoIt.error is set to 1.

Related

MouseClick, MouseClickDrag, MouseCoordMode (Option), MouseGetPosX, MouseGetPosY, MouseMove

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

' Move the wheel up 10 times
oAutoIt.MouseWheel "up", 10
```

Process methods Reference

Below is a complete list of the methods available in AutoItX. Click on a method name for a detailed description.

Method	Description
ProcessClose	Terminates a named process
ProcessExists	Checks to see if a specified process exists
ProcessSetPriority	Changes the priority of a process
ProcessWait	Pauses script execution until a given process exists
ProcessWaitClose	Pauses script execution until a given process does not exist
Run	Runs an external program
RunAs	Runs an external program
RunAsWait	Runs an external program
RunWait	Runs an external program and pauses script execution until the program finishes
Shutdown	Shuts down the system

ProcessClose

Terminates a named process

```
ProcessClose "process"
```

Parameters

process	The title or PID of the process to terminate.

Return Value

None. (Returns 1 regardless of success/failure.)

Remarks

Process names are executables without the full path, e.g., "notepad.exe" or "winword.exe"
If multiple processes have the same name, the one with the highest PID is terminated--regardless of how recently the process was spawned.
PID is the unique number which identifies a Process. A PID can be obtained through the ProcessExists or Run commands.
In order to work under Windows NT 4.0, ProcessClose requires the file PSAPI.DLL (included in the AutoIt installation directory).
The process is polled approximately every 250 milliseconds.

Related

ProcessExists, ProcessWait, ProcessWaitClose, Run, WinClose

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.ProcessClose "notepad.exe"

PID = oAutoIt.ProcessExists("notepad.exe")
oAutoIt.ProcessClose(PID)
```

ProcessExists

Checks to see if a specified process exists

```
ProcessExists "process"
```

Parameters

process	The name or PID of the process to check.

Return Value

Success: Returns the PID of the process.
Failure: Returns 0 if process does not exist.

Remarks

Process names are executables without the full path, e.g., "notepad.exe" or "winword.exe"
PID is the unique number which identifies a Process.
In order to work under Windows NT 4.0, ProcessExists requires the file PSAPI.DLL (included in the AutoIt installation directory).
The process is polled approximately every 250 milliseconds.

Related

ProcessClose, ProcessWait, ProcessWaitClose, WinExists

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
If ProcessExists("notepad.exe") Then
    WScript.Echo "Notepad is running."
End If
```

ProcessSetPriority

Changes the priority of a process

```
ProcessSetPriority "process", priority
```

Parameters

process	The name or PID of the process to check.
priority	A flag which determines what priority to set 0 - Idle/Low 1 - Below Normal (Not supported on Windows 95/98/ME) 2 - Normal 3 - Above Normal (Not supported on Windows 95/98/ME) 4 - High 5 - Realtime (Use with caution, may make the system unstable)

Return Value

Success: Returns 1.
Failure: Returns 0 and sets oAutoIt.error to 1. May set oAutoIt.error to 2 if attempting to use an unsupported priority class.

Remarks

Above Normal and Below Normal priority classes are not supported on Windows 95/98/ME. If you try to use them on those platforms, the function will fail and oAutoIt.error will be set to 2.

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.Run "Notepad.exe"
oAutoIt.ProcessSetPriority "notepad.exe", 0
' Notepad should now have Idle/Low priority
```

ProcessWait

Pauses script execution until a given process exists

```
ProcessWait "process" [, timeout]
```

Parameters

process	The name of the process to check.
timeout	**Optional:** Specifies how long to wait (default is to wait indefinitely).

Return Value

Success: Returns 1.
Failure: Returns 0 if the wait timed out.

Remarks

Process names are executables without the full path, e.g., "notepad.exe" or "winword.exe"
In order to work under Windows NT 4.0, ProcessWait requires the file PSAPI.DLL (included in the AutoIt installation directory).
The process is polled approximately every 250 milliseconds.

This function is the only process function not to accept a PID. Because PIDs are allocated randomly, waiting for a particular PID to exist doesn't make sense.

Related

ProcessClose, ProcessExists, ProcessWaitClose, RunWait, WinWait, WinWaitActive

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.ProcessWait "notepad.exe"
```

ProcessWaitClose

Pauses script execution until a given process does not exist

```
ProcessWaitClose "process" [, timeout]
```

Parameters

process	The name or PID of the process to check.
timeout	**Optional:** Specifies how long to wait (default is to wait indefinitely).

Return Value

Success: Returns 1.
Failure: Returns 0 if wait timed out.

Remarks

Process names are executables without the full path, e.g., "notepad.exe" or "winword.exe"
PID is the unique number which identifies a Process. A PID can be obtained through the ProcessExists or Run commands.
In order to work under Windows NT 4.0, ProcessWaitClose requires the file PSAPI.DLL (included in the AutoIt installation directory).
The process is polled approximately every 250 milliseconds.

Related

ProcessClose, ProcessExists, ProcessWaitClose, RunWait, WinWaitClose

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

'waits until no instance of notepad.exe exists
oAutoIt.ProcessWaitClose "notepad.exe"

' This will wait until this particular instance of
notepad has exited
PID = oAutoIt.Run("notepad.exe")
oAutoIt.ProcessWaitClose(PID)
```

Run

Runs an external program

```
Run "filename" [, "workingdir" [, show_flag]]
```

Parameters

filename	The name of the executable (EXE, BAT, COM, or PIF) to run.
workingdir	**Optional:** The working directory.
show_flag	**Optional:** The "show" flag of the executed program: SW_HIDE = Hidden window SW_MINIMIZE = Minimized window SW_MAXIMIZE = Maximized window

Return Value

Success: The PID of the process that was launched.
Failure: see Remarks.

Remarks

After running the requested program the script continues. To pause execution of the script until the spawned program has finished use the RunWait function instead.

The **error** property is set to 1 as an indication of failure.

Related

RunAs, RunAsWait, RunWait

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.Run "Notepad.exe", "", oAutoIt.SW_MAXIMIZE
```

RunAs

Runs an external program

```
RunAs "user", "domain", "password", logon_flag,
"filename" [, "workingdir" [, show_flag]]
```

Parameters

username	The user name to use.
domain	The domain name to use.
password	The password to use.
logon_flag	0 = do not load the user profile, 1 = (default) load the user profile, 2 = use for net credentials only
filename	The name of the executable (EXE, BAT, COM, or PIF) to run.
workingdir	**Optional:** The working directory.
show_flag	**Optional:** The "show" flag of the executed program: SW_HIDE = Hidden window SW_MINIMIZE = Minimized window SW_MAXIMIZE = Maximized window

Return Value

Success: The PID of the process that was launched.
Failure: see Remarks.

Remarks

After running the requested program the script continues. To pause execution of the script until the spawned program has finished use the RunWait function instead.

The **error** property is set to 1 as an indication of failure.

Related

Run, RunAsWait, RunWait

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.Run "Notepad.exe", "", oAutoIt.SW_MAXIMIZE
```

RunAsWait

Runs an external program

```
RunAsWait "user", "domain", "password", logon_flag,
"filename" [, "workingdir" [, show_flag]]
```

Parameters

username	The user name to use.
domain	The domain name to use.
password	The password to use.
logon_flag	0 = do not load the user profile, 1 = (default) load the user profile, 2 = use for net credentials only
filename	The name of the executable (EXE, BAT, COM, or PIF) to run.
workingdir	**Optional:** The working directory.
show_flag	**Optional:** The "show" flag of the executed program: SW_HIDE = Hidden window SW_MINIMIZE = Minimized window SW_MAXIMIZE = Maximized window

Return Value

Success: The PID of the process that was launched.
Failure: see Remarks.

Remarks

After running the requested program the script pauses until the program terminates. To run a program and then immediately continue script execution use the Run function instead.

Some programs will appear to return immediately even though they are still running; these programs spawn another process - you may be able to use the ProcessWaitClose function to handle these cases.

The **error property is set to 1 as an indication of failure.**

Related

Run, RunAsWait, RunWait

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.Run "Notepad.exe", "", oAutoIt.SW_MAXIMIZE
```

RunWait

Runs an external program and pauses script execution until the program finishes

```
RunWait "filename" [, "workingdir" [, flag]]
```

Parameters

filename	The name of the executable (EXE, BAT, COM, PIF) to run.
workingdir	**Optional:** The working directory.
flag	**Optional:** The "show" flag of the executed program: SW_HIDE = Hidden window SW_MINIMIZE = Minimized window SW_MAXIMIZE = Maximized window

Return Value

Success: Returns the exit code of the program that was run.

Failure: see Remarks.

Remarks

After running the requested program the script pauses until the program terminates. To run a program and then immediately continue script execution use the Run function instead.

Some programs will appear to return immediately even though they are still running; these programs spawn another process - you may be able to use the ProcessWaitClose function to handle these cases.

The **error property is set to 1 as an indication of failure.**

Related

ProcessWait, ProcessWaitClose, RunAs, RunAsWait, RunWait

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
val = oAutoIt.RunWait("Notepad.exe", "C:\WINDOWS",
oAutoIt.SW_MAXIMIZE)

' script waits until Notepad closes
WScript.Echo "Program returned with exit code:" & v
al
```

Shutdown

Shuts down the system

```
Shutdown ( code )
```

Parameters

code	A combination of shutdown codes. See "remarks".

Return Value

Success: Returns 1.

Failure: Returns 0.

Remarks

The shutdown code is a combination of the following values:
0 = Logoff
1 = Shutdown
2 = Reboot
4 = Force
8 = Power down

Add the required values together. To shutdown and power down, for example, the code would be 9 (shutdown + power down = 1 + 8 = 9).

Standby or hibernate can be achieved with third-party software such as http://grc.com/wizmo/wizmo.htm

Related

ProcessClose

Example

```
' Force a reboot
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.Shutdown 6
```

Timer and Delay methods Reference

Below is a complete list of the methods available in AutoItX. Click on a method name for a detailed description.

Method	Description
Sleep	Pause script execution

Sleep

Pause script execution

```
Sleep delay
```

Parameters

delay	Amount of time to pause (in milliseconds).

Return Value

None.

Remarks

Maximum sleep time is 2147483647 milliseconds (24 days).

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.Sleep 5000  'five seconds
```

Window methods Reference

Below is a complete list of the methods available in AutoItX. Click on a method name for a detailed description.

Method	Description
WinActivate	Activates (gives focus to) a window
WinActive	Checks to see if a specified window exists and is currently active
WinClose	Closes a window
WinExists	Checks to see if a specified window exists
WinGetCaretPosX	Returns the coordinates of the caret in the foreground window
WinGetCaretPosY	Returns the coordinates of the caret in the foreground window
WinGetClassList	Retrieves the classes from a window
WinGetClientSizeHeight	Retrieves the size of a given window's client area
WinGetClientSizeWidth	Retrieves the size of a given window's client area
WinGetHandle	Retrieves the internal handle of a window
WinGetPosHeight	Retrieves the position and size of a given window
WinGetPosWidth	Retrieves the position and size of a given window
WinGetPosX	Retrieves the position and size of a given window
WinGetPosY	Retrieves the position and size of a given window
WinGetProcess	Retrieves the Process ID (PID) associated with a window
WinGetState	Retrieves the state of a given window
WinGetText	Retrieves the text from a window
WinGetTitle	Retrieves the full title from a window

WinKill	Forces a window to close
WinList	Retrieves a list of windows
WinMenuSelectItem	Invokes a menu item of a window
WinMinimizeAll	Minimizes all windows
WinMinimizeAllUndo	Undoes a previous WinMinimizeAll function
WinMove	Moves and/or resizes a window
WinSetOnTop	Change a window's "Always On Top" attribute
WinSetState	Shows, hides, minimizes, maximizes, or restores a window
WinSetTitle	Changes the title of a window
WinSetTrans	Sets the transparency of a window
WinWait	Pauses execution of the script until the requested window exists
WinWaitActive	Pauses execution of the script until the requested window is active
WinWaitClose	Pauses execution of the script until the requested window does not exist
WinWaitNotActive	Pauses execution of the script until the requested window is not active

Controls methods Reference

Below is a complete list of the methods available in AutoItX. Click on a method name for a detailed description.

Method	Description
ControlClick	Sends a mouse click command to a given control
ControlCommand	Sends a command to a control
ControlDisable	Disables or "grays-out" a control
ControlEnable	Enables a "grayed-out" control
ControlFocus	Sets input focus to a given control on a window
ControlGetFocus	Returns the ControlRef# of the control that has keyboard focus within a specified window
ControlGetHandle	Retrieves the internal handle of a control
ControlGetPosHeight	Retrieves the position and size of a control relative to it's window
ControlGetPosWidth	Retrieves the position and size of a control relative to it's window
ControlGetPosX	Retrieves the position and size of a control relative to it's window
ControlGetPosY	Retrieves the position and size of a control relative to it's window
ControlGetText	Retrieves text from a control
ControlHide	Hides a control
ControlListView	Sends a command to a ListView32 control
ControlMove	Moves a control within a window
ControlSend	Sends a string of characters to a control
ControlSetText	Sets text of a control
ControlShow	Shows a control that was hidden

ControlTreeView	Sends a command to a TreeView32 control
StatusbarGetText	Retrieves the text from a standard status bar control

ControlClick

Sends a mouse click command to a given control

```
ControlClick "title", "text", "controlID" [, button
[, clicks [, x [, y ]]]]
```

Parameters

title	The title of the window to access.
text	The text of the window to access.
controlID	The control to interact with. See Controls.
button	**Optional:** The button to click, "left", "right" or "middle". Default is the left button.
clicks	**Optional:** The number of times to click the mouse. Default is 1.
x	**Optional:** The x position to click within the control. Default is center.
y	**Optional:** The y position to click within the control. Default is center.

Return Value

Success: Returns 1.

Failure: Returns 0.

Remarks

Some controls will resist clicking unless they are the active window. Use the WinActive() function to force the control's window to the top before using ControlClick().
Using 2 for the number of clicks will send a double-click message to the control - this can even be used to launch programs from an explorer control!

Related

ControlCommand, MouseClick, WinActivate

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.ControlClick "Untitled -", "", "MDIClient1"
```

ControlCommand

Sends a command to a control

```
ControlCommand "title", "text", "controlID",
"command", "option"
```

Parameters

title	The title of the window to access.
text	The text of the window to access.
controlID	The control to interact with. See Controls.
command	The command to send to the control.
option	Additional parameter required by some commands; use "" if parameter is not required.

Return Value

Depends on command as table below shows. In case of an error (such as an invalid command or window/control), oAutoIt.error=1.

Command, Option	**Return Value**
"IsVisible", ""	Returns 1 if Control is visible, 0 otherwise
"IsEnabled", ""	Returns 1 if Control is enabled, 0 otherwise
"ShowDropDown", ""	Drops a ComboBox
"HideDropDown", ""	UNdrops a ComboBox
"AddString", 'string'	Adds a string to the end in a ListBox or ComboBox
"DelString", occurrence	Deletes a string according to occurrence in a ListBox or ComboBox
"FindString", 'string'	Returns occurrence ref of the exact string in a ListBox or ComboBox
"SetCurrentSelection", *occurrence*	Sets selection to occurrence ref in a ListBox or ComboBox
"SelectString", 'string'	Sets selection according to string in a ListBox or ComboBox
"IsChecked", ""	Returns 1 if Button is checked, 0 otherwise
"Check", ""	Checks radio or check Button
"UnCheck", ""	Unchecks radio or check Button
"GetCurrentLine", ""	Returns the line # where the caret is in an Edit
"GetCurrentCol", ""	Returns the column # where the caret is in an Edit
"GetCurrentSelection", ""	Returns name of the currently selected item in a ListBox or ComboBox
"GetLineCount", ""	Returns # of lines in an Edit
"GetLine", *line#*	Returns text at line # passed of an Edit
"GetSelected", ""	Returns selected text of an Edit
"EditPaste", 'string'	Pastes the 'string' at the Edit's caret position

"CurrentTab", ""	Returns the current Tab shown of a SysTabControl32
"TabRight", ""	Moves to the next tab to the right of a SysTabControl32
"TabLeft", ""	Moves to the next tab to the left of a SysTabControl32

Remarks

Certain commands that work on normal Combo and ListBoxes do not work on "ComboLBox" controls.
When using a control name in the Control functions, you need to add a number to the end of the name to indicate which control. For example, if there two controls listed called "MDIClient", you would refer to these as "MDIClient1" and "MDIClient2". Use AU3_Spy.exe to obtain a control's number.

When using text instead of ClassName# in "Control" commands, be sure to use the entire text of the control. Partial text will fail.

Related

ControlClick, ControlDisable, ControlEnable, ControlFocus, ControlGetPosX, ControlGetPosY, ControlGetText, ControlHide, ControlMove, ControlSetText, ControlShow, StatusbarGetText, WinGetClassList, WinMenuSelectItem

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.ControlCommand "Untitled - Notepad", "", "E
dit1", "GetLineCount", ""
```

ControlDisable

Disables or "grays-out" a control

```
ControlDisable "title", "text", "controlID"
```

Parameters

title	The title of the window to access.
text	The text of the window to access.
controlID	The control to interact with. See Controls.

Return Value

Success: Returns 1.

Failure: Returns 0.

Remarks

When using a control name in the Control functions, you need to add a number to the end of the name to indicate which control. For example, if there two controls listed called "MDIClient", you would refer to these as "MDIClient1" and "MDIClient2".

Related

ControlCommand, ControlEnable, ControlHide

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.ControlDisable "Untitled -", "", "MDIClient
1"
```

ControlEnable

Enables a "grayed-out" control

```
ControlEnable "title", "text", "controlID"
```

Parameters

title	The title of the window to access.
text	The text of the window to access.
controlID	The control to interact with. See Controls.

Return Value

Success: Returns 1.

Failure: Returns 0.

Remarks

Use with caution.
When using a control name in the Control functions, you need to add a number to the end of the name to indicate which control. For example, if there two controls listed called "MDIClient", you would refer to these as "MDIClient1" and "MDIClient2".

Related

ControlCommand, ControlDisable, ControlShow

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.ControlEnable "Untitled -", "", "MDIClient1
"
```

ControlFocus

Sets input focus to a given control on a window

```
ControlFocus "title", "text", "controlID"
```

Parameters

title	The title of the window to access.
text	The text of the window to access.
controlID	The control to interact with. See Controls.

Return Value

Success: Returns 1.

Failure: Returns 0.

Remarks

When using a control name in the Control functions, you need to add a number to the end of the name to indicate which control. For example, if there two controls listed called "MDIClient", you would refer to these as "MDIClient1" and "MDIClient2".

Related

ControlCommand, WinActivate

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.ControlFocus "Untitled - Notepad", "", "Edi
t1"
```

ControlGetFocus

Returns the ControlRef# of the control that has keyboard focus within a specified window

```
ControlGetFocus "title" [, "text"]
```

Parameters

title	Title of window to check.
text	**Optional:** Text from window to check.

Return Value

Success:	Returns ControlRef# of the control that has keyboard focus within a specified window.
Failure:	Returns a blank string and sets oAutoIt.error to 1 if window is not found.

Related

ControlCommand, ControlFocus, WinActive

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
a = oAutoIt.ControlGetFocus("Untitled - Notepad")
```

ControlGetHandle

Retrieves the internal handle of a control

```
ControlGetHandle "title", "text", "controlID"
```

Parameters

title	The title of the window to read.
text	The text of the window to read.
controlID	The control to interact with. See Controls.

Return Value

Success:	Returns a string containing the control handle value.
Failure:	Returns "" (blank string) and sets oAutoIt.error to 1 if no window matches the criteria.

Related

WinTitleMatchMode (Option)

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Control")
handle = oAutoIt.ControlGetHandle("Untitled - Notepad", "", "Edit1")
```

ControlGetPosX

Retrieves the position and size of a control relative to it's window

```
ControlGetPosX "title", "text", "controlID"
```

Parameters

title	The title of the window to access.
text	The text of the window to access.
controlID	The control to interact with. See Controls.

Return Value

Success: Returns the X coordinate of the control.

Failure: Sets oAutoIt.error to 1.

Remarks

When using a control name in the Control functions, you need to add a number to the end of the name to indicate which control. For example, if there two controls listed called "MDIClient", you would refer to these as "MDIClient1" and "MDIClient2".

Related

ControlCommand, ControlGetPosHeight, ControlGetPosWidth, ControlGetPosY, ControlMove, WinMove

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

posx = ControlGetPosX("Untitled - Notepad", "", "Ed
it1")
posy = ControlGetPosY("Untitled - Notepad", "", "Ed
it1")
poswidth = ControlGetPosWidth("Untitled - Notepad",
 "", "Edit1")
posheight = ControlGetPosHeight("Untitled - Notepad
", "", "Edit1")
```

ControlGetPosY

Retrieves the position and size of a control relative to it's window

```
ControlGetPosY "title", "text", "controlID"
```

Parameters

title	The title of the window to access.
text	The text of the window to access.
controlID	The control to interact with. See Controls.

Return Value

Success: Returns the Y coordinate of the control.

Failure: Sets oAutoIt.error to 1.

Remarks

When using a control name in the Control functions, you need to add a number to the end of the name to indicate which control. For example, if there two controls listed called "MDIClient", you would refer to these as "MDIClient1" and "MDIClient2".

Related

ControlCommand, ControlGetPosHeight, ControlGetPosWidth, ControlGetPosX, ControlMove, WinMove

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

posx = ControlGetPosX("Untitled - Notepad", "", "Ed
it1")
posy = ControlGetPosY("Untitled - Notepad", "", "Ed
it1")
poswidth = ControlGetPosWidth("Untitled - Notepad",
 "", "Edit1")
posheight = ControlGetPosHeight("Untitled - Notepad
", "", "Edit1")
```

ControlGetPosWidth

Retrieves the position and size of a control relative to it's window

```
ControlGetPosWidth "title", "text", "controlID"
```

Parameters

title	The title of the window to access.
text	The text of the window to access.
controlID	The control to interact with. See Controls.

Return Value

Success: Returns the width of the control.

Failure: Sets oAutoIt.error to 1.

Remarks

When using a control name in the Control functions, you need to add a number to the end of the name to indicate which control. For example, if there two controls listed called "MDIClient", you would refer to these as "MDIClient1" and "MDIClient2".

Related

ControlCommand, ControlGetPosHeight, ControlGetPosX, ControlGetPosY, ControlMove, WinMove

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

posx = ControlGetPosX("Untitled - Notepad", "", "Ed
it1")
posy = ControlGetPosY("Untitled - Notepad", "", "Ed
it1")
poswidth = ControlGetPosWidth("Untitled - Notepad",
 "", "Edit1")
posheight = ControlGetPosHeight("Untitled - Notepad
", "", "Edit1")
```

ControlGetPosHeight

Retrieves the position and size of a control relative to it's window

```
ControlGetPosHeight "title", "text", "controlID"
```

Parameters

title	The title of the window to access.
text	The text of the window to access.
controlID	The control to interact with. See Controls.

Return Value

Success: Returns the height of the control.

Failure: Sets oAutoIt.error to 1.

Remarks

When using a control name in the Control functions, you need to add a number to the end of the name to indicate which control. For example, if there two controls listed called "MDIClient", you would refer to these as "MDIClient1" and "MDIClient2".

Related

ControlCommand, ControlGetPosWidth, ControlGetPosX, ControlGetPosY, ControlMove, WinMove

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

posx = ControlGetPosX("Untitled - Notepad", "", "Ed
it1")
posy = ControlGetPosY("Untitled - Notepad", "", "Ed
it1")
poswidth = ControlGetPosWidth("Untitled - Notepad",
 "", "Edit1")
posheight = ControlGetPosHeight("Untitled - Notepad
", "", "Edit1")
```

ControlGetText

Retrieves text from a control

```
ControlGetText "title", "text", "controlID"
```

Parameters

title	The title of the window to access.
text	The text of the window to access.
controlID	The control to interact with. See Controls.

Return Value

Success: Returns text from a control.

Failure: Sets oAutoIt.error to 1 and returns a blank string of "".

Remarks

When using a control name in the Control functions, you need to add a number to the end of the name to indicate which control. For example, if there two controls listed called "MDIClient", you would refer to these as "MDIClient1" and "MDIClient2".

Related

ControlCommand, ControlSetText, StatusbarGetText, WinGetText

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
var = oAutoIt.ControlGetText("Untitled - Notepad",
"", "Edit1")
```

ControlHide

Hides a control

```
ControlHide "title", "text", "controlID"
```

Parameters

title	The title of the window to access.
text	The text of the window to access.
controlID	The control to interact with. See Controls.

Return Value

Success: Returns 1.

Failure: Returns 0 if window/control is not found.

Remarks

When using a control name in the Control functions, you need to add a number to the end of the name to indicate which control. For example, if there two controls listed called "MDIClient", you would refer to these as "MDIClient1" and "MDIClient2".

Related

ControlCommand, ControlShow, WinSetState

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.ControlHide "Untitled -", "", "MDIClient1"
```

ControlListView

Sends a command to a ListView32 control

```
ControlListView "title", "text", "controlID",
"command", "option1", "option2"
```

Parameters

title	The title of the window to access.
text	The text of the window to access.
controlID	The control to interact with. See Controls.
command	The command to send to the control (see below).
option1	Additional parameter required by some commands; use "" if parameter is not required.
option2	Additional parameter required by some commands; use "" if parameter is not required.

Return Value

Depends on command as table below shows. In case of an error (such as an invalid command or window/control could not be found) then @error is set to 1.

All items/subitems are 0 based. This means that the first item/subitem in a list is 0, the second is 1, and so on.

In a "Details" view of a ListView32 control, the "item" can be thought of as the "row" and the "subitem" as the "column".

Command, Option1, Option2	**Operation**
"DeSelect", From [, To]	Deselects one or more items.
"FindItem", "string to find" [, SubItem]	Returns the item index of the string. Returns -1 if the string is not found.
"GetItemCount"	Returns the number of list items.
"GetSelected" [, option]	Returns a string containing the item index of selected items. If option=0 (default) only the first selected item is returned. If option=1 then all the selected items are returned delimited by |, e.g: "0|3|4|10". If no items are selected a blank "" string is returned.
"GetSelectedCount"	Returns the number of items that are selected.
"GetSubItemCount"	Returns the number of subitems.
"GetText", Item, SubItem	Returns the text of a given item/subitem.
"IsSelected", Item	Returns 1 if the item is selected, otherwise returns 0.
"Select", From [, To]	Selects one or more items.
"SelectAll"	Selects all items.
"SelectClear"	Clears the selection of all items.
"SelectInvert"	Inverts the current selection.

"ViewChange", "view"	Changes the current view. Valid views are "list", "details", "smallicons", "largeicons".

Related

ControlClick, ControlCommand, ControlDisable, ControlEnable, ControlFocus, ControlGetPosX, ControlGetPosY, ControlGetText, ControlHide, ControlMove, ControlSetText, ControlShow, ControlTreeView, StatusbarGetText, WinGetClassList, WinMenuSelectItem

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.ControlListView "C:\Program Files\NSIS", ""
, "SysListView321", "SelectAll", "", ""
oAutoIt.ControlListView "C:\Program Files\NSIS", ""
, "SysListView321", "Deselect", "2", "5"
```

ControlMove

Moves a control within a window

```
ControlMove "title", "text", "controlID", x, y [,
width [, height]]
```

Parameters

title	The title of the window to move.
text	The text of the window to move.
controlID	The control to interact with. See Controls.
x	X coordinate to move to.
y	Y coordinate to move to.
width	**Optional:** New width of the window.
height	**Optional:** New height of the window.

Return Value

Success: Returns 1.

Failure: Returns 0 if window/control is not found.

Remarks

When using a control name in the Control functions, you need to add a number to the end of the name to indicate which control. For example, if there two controls listed called "MDIClient", you would refer to these as "MDIClient1" and "MDIClient2".

Related

ControlCommand, WinMove

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.ControlMove "Untitled -", "", "MDIClient1",
 0, 0, 200, 200
```

ControlSend

Sends a string of characters to a control

```
ControlSend "title", "text", "controlID", "string"
[, flag]
```

Parameters

title	The title of the window to access.
text	The text of the window to access.
controlID	The control to interact with. See Controls.
string	String of characters to send to the control.
flag	**Optional:** Changes how "keys" is processed: flag = 0 (default), Text contains special characters like + to indicate SHIFT and {LEFT} to indicate left arrow. flag = 1, keys are sent raw.

Return Value

Success: Returns 1.

Failure: Returns 0 if window/control is not found.

Remarks

ControlSend can be quite useful to send capital letters without messing up the state of "Shift."

When using a control name in the Control functions, you need to add a number to the end of the name to indicate which control. For example, if there two controls listed called "MDIClient", you would refer to these as "MDIClient1" and "MDIClient2".
Note, this function cannot send all the characters that the usual Send function can (notably ALT keys) but it can send most of them--even to non-active or hidden windows!

Related

ControlCommand, Send

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.ControlSend "Untitled", "", "Edit1", "This
is a line of text in the notepad window"
```

ControlSetText

Sets text of a control

```
ControlSetText "title", "text", "controlID", "new
text"
```

Parameters

title	The title of the window to access.
text	The text of the window to access.
controlID	The control to interact with. See Controls.
new text	The new text to be set into the control.

Return Value

Success: Returns 1.

Failure: Returns 0 if window/control is not found.

Remarks

When using a control name in the Control functions, you need to add a number to the end of the name to indicate which control. For example, if there two controls listed called "MDIClient", you would refer to these as "MDIClient1" and "MDIClient2".

Related

ControlCommand, ControlGetText

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

oAutoIt.Run "notepad.exe"
oAutoIt.WinWait "Untitled -"
oAutoIt.ControlSetText "Untitled -", "", "Edit1", "
New Text Here"
```

ControlShow

Shows a control that was hidden

```
ControlShow "title", "text", "controlID"
```

Parameters

title	The title of the window to access.
text	The text of the window to access.
controlID	The control to interact with. See Controls.

Return Value

Success: Returns 1.

Failure: Returns 0 if window/control is not found.

Remarks

When using a control name in the Control functions, you need to add a number to the end of the name to indicate which control. For example, if there two controls listed called "MDIClient", you would refer to these as "MDIClient1" and "MDIClient2".

Related

ControlCommand, ControlEnable, ControlHide

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.ControlShow "Untitled -", "", "MDIClient1"
```

ControlTreeView

Sends a command to a TreeView32 control

```
ControlTreeView "title", "text", "controlID",
"command", "option1", "option2"
```

Parameters

title	The title of the window to access.
text	The text of the window to access.
controlID	The control to interact with. See Controls.
command	The command to send to the control (see below).
option1	Additional parameter required by some commands; use "" if parameter is not required.
option2	Additional parameter required by some commands; use "" if parameter is not required.

Return Value

Depends on command as table below shows. In case of an error (such as an invalid command or window/control could not be found) then @error is set to 1.

Command, Option1, Option2	**Operation**
"Check", "item"	Checks an item (if the item supports it).
"Collapse", "item"	Collapses an item to hide its children.
"Exists", "item"	Returns 1 if an item exists, otherwise 0.
"Expand", "item"	Expands an item to show its children.
"GetItemCount", "item"	Returns the number of children for a selected item.
"GetSelected" [, UseIndex]	Returns the item reference of the current selection using the text reference of the item (or index reference if UseIndex is set to 1).
"GetText", "item"	Returns the text of an item.
"IsChecked"	Returns the state of an item. 1:checked, 0:unchecked, -1:not a checkbox.
"Select", "item"	Selects an item.
"Uncheck", "item"	Unchecks an item (if the item supports it).

Related

ControlClick, ControlCommand, ControlDisable, ControlEnable, ControlFocus, ControlGetPosX, ControlGetPosY, ControlGetText, ControlHide, ControlListView, ControlMove, ControlSetText, ControlShow, StatusbarGetText, WinGetClassList, WinMenuSelectItem

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.ControlListView "C:\", "", "SysTreeView321"
, "SelectAll", "", ""
oAutoIt.ControlListView "C:\", "", "SysTreeView321"
, "Deselect", "2", "5"
```

StatusbarGetText

Retrieves the text from a standard status bar control

```
StatusbarGetText "title" [, "text" [, part]]
```

Parameters

title	The title of the window to check.
text	**Optional:** The text of the window to check.
part	**Optional:** The "part" number of the status bar to read - the default is 1. 1 is the first possible part and usually the one that contains the useful messages like "Ready" "Loading...", etc.

Return Value

Success:	Returns the text read.
Failure:	Returns empty string and sets oAutoIt.error to 1 if no text could be read.

Remarks

This functions attempts to read the first standard status bar on a window (Microsoft common control: msctls_statusbar32). Some programs use their own status bars or special versions of the MS common control which StatusbarGetText cannot read. For example, StatusbarText does not work on the program TextPad; however, the first region of TextPad's status bar can be read using ControlGetText("TextPad", "", "HSStatusBar1")
StatusbarGetText can work on windows that are minimized or even hidden.

Related

ControlCommand, ControlGetText

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

oAutoIt.AutoItSetOption("WinTitleMatchMode", 2)
text = oAutoIt.StatusbarGetText "Internet Explorer"
WScript.Echo "Internet Explorer's status bar says:
" & text
```

WinActivate

Activates (gives focus to) a window

```
WinActivate "title" [, "text"]
```

Parameters

title	The title of the window to activate.
text	**Optional:** The text of the window to activate.

Return Value

None.

Remarks

You can use the WinActive function to check if WinActivate succeeded. If multiple windows match the criteria, the window that was most recently active is the one activated. WinActivate works on minimized windows. However, a window that is "Always On Top" could still cover up a window you Activated.

Related

WinClose, WinSetState, WinTitleMatchMode (Option)

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.WinActivate "Untitled - Notepad", ""
```

WinActive

Checks to see if a specified window exists and is currently active

```
WinActive "title" [, "text"]
```

Parameters

title	The title of the window to activate.
text	**Optional:** The text of the window to activate.

Return Value

Success: Returns 1.

Failure: Returns 0 if window is not active.

Related

WinExists, WinTitleMatchMode (Option), WinWait, WinWaitActive, WinWaitClose, WinWaitNotActive

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

If oAutoIt.WinActive("Untitled -") Then
    WScript.Echo "Window was active"
End If
```

WinClose

Closes a window

```
WinClose "title" [, "text"]
```

Parameters

title	The title of the window to close.
text	**Optional:** The text of the window to close.

Return Value

None.

Remarks

This function sends a close message to a window, the result depends on the window (it may ask to save data, etc.). To force a window to close, use the WinKill function. If multiple windows match the criteria, the window that was most recently active is closed.

Related

WinActivate, WinExists, WinKill, WinSetState, WinTitleMatchMode (Option)

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.WinClose "Untitled - Notepad", ""
```

WinExists

Checks to see if a specified window exists

```
WinExists "title" [, "text"]
```

Parameters

title	The title of the window to check.
text	**Optional:** The text of the window to check.

Return Value

Returns 1 if the window exists, otherwise returns 0.

Remarks

WinExist will return 1 even if a window is hidden.

Related

WinActive, WinTitleMatchMode (Option), WinWait, WinWaitActive, WinWaitClose, WinWaitNotActive

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

If oAutoIt.WinExists("Untitled -") Then
    WScript.Echo "Window exists"
EndIf
```

WinGetCaretPosX

Returns the coordinates of the caret in the foreground window

```
WinGetCaretPosX
```

Return Value

Success: Returns the X coordinate of the caret.

Failure: Sets oAutoIt.error to 1.

Remarks

WinGetCaretPos might not return accurate values for Multiple Document Interface (MDI) applications if absolute CaretCoordMode is used. See example for a workaround. Note: Some applications report static coordinates regardless of caret position!

Related

CaretCoordMode (Option)

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
x = oAutoIt.WinGetCaretPosX()
y = oAutoIt.WinGetCaretPosY()
```

WinGetCaretPosY

Returns the coordinates of the caret in the foreground window

```
WinGetCaretPosY
```

Return Value

Success: Returns the Y coordinate of the caret.

Failure: Sets oAutoIt.error to 1.

Remarks

WinGetCaretPos might not return accurate values for Multiple Document Interface (MDI) applications if absolute CaretCoordMode is used. See example for a workaround. Note: Some applications report static coordinates regardless of caret position!

Related

CaretCoordMode (Option)

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
x = oAutoIt.WinGetCaretPosX()
y = oAutoIt.WinGetCaretPosY()
```

WinGetClassList

Retrieves the classes from a window

```
WinGetClassList "title" [, "text"]
```

Parameters

title	The title of the window to read.
text	**Optional:** The text of the window to read.

Return Value

Success:	Returns a string containing the window classes read.
Failure:	Returns numeric 1 and sets oAutoIt.error to 1 if no window matches the criteria.

Remarks

Class names are linefeed separated. WinGetClassList works on both minimized and hidden windows. Up to 64KB of text can be retrieved. If multiple windows match the criteria, the classes are read from the most recently active window.

Related

ControlCommand, WinGetText

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

text = oAutoIt.WinGetClassList("Untitled -", "")
WScript.Echo "Text read was:" & text
```

WinGetClientSizeWidth

Retrieves the size of a given window's client area

```
WinGetClientSizeWidth "title" [, "text"]
```

Parameters

title	The title of the window to read.
text	**Optional:** The text of the window to read.

Return Value

Success:	Returns the width of the window's client area
Failure:	Returns numeric 1 and sets oAutoIt.error to 1 if windows is not found.

Remarks

If the window is minimized, the returned width and height values are both zero. However, WinGetClientSize works correctly on (non-minimized) hidden windows. If the window title "Program Manager" is used, the function will return the size of the desktop. WinGetClientSize("") matches the active window. If multiple windows match the criteria, the most recently active window is used.

Related

WinGetPosX, WinGetPosY, WinMove

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

$width = oAutoIt.WinGetClientSizeWidth("")
$height = oAutoIt.WinGetClientSizeHeight("")
```

WinGetClientSizeHeight

Retrieves the size of a given window's client area

```
WinGetClientSizeHeight "title" [, "text"]
```

Parameters

title	The title of the window to read.
text	**Optional:** The text of the window to read.

Return Value

Success:	Returns the height of the window's client area
Failure:	Returns numeric 1 and sets oAutoIt.error to 1 if windows is not found.

Remarks

If the window is minimized, the returned width and height values are both zero. However, WinGetClientSize works correctly on (non-minimized) hidden windows. If the window title "Program Manager" is used, the function will return the size of the desktop. WinGetClientSize("") matches the active window. If multiple windows match the criteria, the most recently active window is used.

Related

WinGetPosX, WinGetPosY, WinMove

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

$width = oAutoIt.WinGetClientSizeWidth("")
$height = oAutoIt.WinGetClientSizeHeight("")
```

WinGetHandle

Retrieves the internal handle of a window

```
WinGetHandle "title" [, "text"]
```

Parameters

title	The title of the window to read.
text	**Optional:** The text of the window to read.

Return Value

Success:	Returns a string containing the window handle value.
Failure:	Returns "" (blank string) and sets oAutoIt.error to 1 if no window matches the criteria.

Remarks

This function is for use with the advanced WinTitleMatchMode options that allow you to use classnames and handles to specify windows rather than "title" and "text".
Once you have obtained the handle you can access the required window even if its title changes.

Related

WinTitleMatchMode (Option)

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
handle = oAutoIt.WinGetHandle("[CLASS:Notepad]", ""
)
```

WinGetPosX

Retrieves the position and size of a given window

```
WinGetPosX "title" [, "text"]
```

Parameters

title	The title of the window to read.
text	**Optional:** The text of the window to read.

Return Value

Success: Returns the X coordinate of the window.

Failure: Returns numeric 1 and sets oAutoIt.error to 1 if windows is not found.

Remarks

WinGetPosX returns negative numbers such as -32000 for minimized windows, but works fine with (non-minimized) hidden windows.
If the window title "Program Manager" is used, the function will return the size of the desktop. If multiple windows match the criteria, the most recently active window is used.

Related

WinGetClientSizeHeight, WinGetClientSizeWidth, WinMove

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
x = oAutoIt.WinGetPosX("")
y = oAutoIt.WinGetPosY("")
width = oAutoIt.WinGetPosWidth("")
height = oAutoIt.WinGetPosHeight("")
```

WinGetPosY

Retrieves the position and size of a given window

```
WinGetPosY "title" [, "text"]
```

Parameters

title	The title of the window to read.
text	**Optional:** The text of the window to read.

Return Value

Success:	Returns the Y coordinate of the window.
Failure:	Returns numeric 1 and sets oAutoIt.error to 1 if windows is not found.

Remarks

WinGetPosY returns negative numbers such as -32000 for minimized windows, but works fine with (non-minimized) hidden windows.
If the window title "Program Manager" is used, the function will return the size of the desktop. If multiple windows match the criteria, the most recently active window is used.

Related

WinGetClientSizeHeight, WinGetClientSizeWidth, WinMove

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
x = oAutoIt.WinGetPosX("")
y = oAutoIt.WinGetPosY("")
width = oAutoIt.WinGetPosWidth("")
height = oAutoIt.WinGetPosHeight("")
```

WinGetPosWidth

Retrieves the position and size of a given window

```
WinGetPosWidth "title" [, "text"]
```

Parameters

title	The title of the window to read.
text	**Optional:** The text of the window to read.

Return Value

Success: Returns the width of the window.

Failure: Returns numeric 1 and sets oAutoIt.error to 1 if windows is not found.

Remarks

WinGetPos returns negative numbers such as -32000 for minimized windows, but works fine with (non-minimized) hidden windows.
If the window title "Program Manager" is used, the function will return the size of the desktop. If multiple windows match the criteria, the most recently active window is used.

Related

WinGetClientSizeHeight, WinGetClientSizeWidth, WinMove

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
x = oAutoIt.WinGetPosX("")
y = oAutoIt.WinGetPosY("")
width = oAutoIt.WinGetPosWidth("")
height = oAutoIt.WinGetPosHeight("")
```

WinGetPosHeight

Retrieves the position and size of a given window

```
WinGetPosHeight "title" [, "text"]
```

Parameters

title	The title of the window to read.
text	**Optional:** The text of the window to read.

Return Value

Success: Returns the height of the window.

Failure: Returns numeric 1 and sets oAutoIt.error to 1 if windows is not found.

Remarks

WinGetPos returns negative numbers such as -32000 for minimized windows, but works fine with (non-minimized) hidden windows.
If the window title "Program Manager" is used, the function will return the size of the desktop. If multiple windows match the criteria, the most recently active window is used.

Related

WinGetClientSizeHeight, WinGetClientSizeWidth, WinMove

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
x = oAutoIt.WinGetPosX("")
y = oAutoIt.WinGetPosY("")
width = oAutoIt.WinGetPosWidth("")
height = oAutoIt.WinGetPosHeight("")
```

WinGetProcess

Retrieves the Process ID (PID) associated with a window

```
WinGetProcess "title" [, "text"]
```

Parameters

title	The title of the window to read.
text	**Optional:** The text of the window to read.

Return Value

Success:	Returns a string containing the numeric Process ID (PID).
Failure:	Returns "".

Related

ProcessWait, ProcessWaitClose

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Control")
WScript.Echo oAutoIt.WinGetProcess("Untitled - Notepad")
```

WinGetState

Retrieves the state of a given window

```
WinGetState "title" [, "text"]
```

Parameters

title	The title of the window to read.
text	**Optional:** The text of the window to read.

Return Value

Success:	Returns a value indicating the state of the window. Multiple values are added together so use BitAND() to examine the part you are interested in: 1 = Window exists 2 = Window is visible 4 = Windows is enabled 8 = Window is active 16 = Window is minimized 32 = Windows is maximized
Failure:	Returns 0 and sets oAutoIt.error to 1 if the window is not found.

Related

WinGetPosX, WinGetPosY

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
state = oAutoIt.WinGetState "Untitled", ""
```

WinGetText

Retrieves the text from a window

```
WinGetText "title" [, "text"]
```

Parameters

title	The title of the window to read.
text	**Optional:** The text of the window to read.

Return Value

Returns a string containing the window text read.

Remarks

Up to 64KB of window text can be retrieved. WinGetText works on minimized windows, but only works on hidden windows if you've set AutoItSetOption("WinDetectHiddenText", 1)
If multiple windows match the criteria for WinGetText, the information for the most recently active match is returned.
Use WinGetText("") to get the active window's text.

Related

ControlGetText, WinGetTitle

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
text = oAutoIt.WinGetText("Untitled -", "")
WScript.Echo "Text read was:" & text
```

WinGetTitle

Retrieves the full title from a window

```
WinGetTitle "title" [, "text"]
```

Parameters

title	The title of the window to read.
text	**Optional:** The text of the window to read.

Return Value

Returns a string containing the complete window title. Returns numeric 0 if no title match.

Remarks

WinGetTitle("") returns the active window's title. WinGetTitle works on both minimized and hidden windows. If multiple windows match the criteria, the most recently active window is used.

Related

WinGetText, WinSetTitle, WinTitleMatchMode (Option)

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

title = oAutoIt.WinGetTitle "Untitled -", ""
WScript.Echo "Full title read was:" & title
```

WinKill

Forces a window to close

```
WinKill "title" [, "text"]
```

Parameters

title	The title of the window to close.
text	**Optional:** The text of the window to close.

Return Value

None. (Always returns 1 regardless of success.)

Remarks

The difference between this function and WinClose is that WinKill will forcibly terminate the window if it doesn't close quickly enough. Consequently, a user might not have time to respond to dialogs prompting the user to save data. Although WinKill can work on both minimized and hidden windows, some windows (notably explorer windows) can only be terminated using WinClose.

Related

ProcessClose, WinActivate, WinClose, WinSetState

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.WinKill "Untitled - ", ""
```

WinList

Retrieves a list of windows

```
WinList "title" [, "text"]
```

Parameters

title	The title of the window to find.
text	**Optional:** The text of the window to find.

Return Value

Success:	Returns a 2 dimensional array containing the window titles and corresponding handles. The number of entries is in element 0,0 (see example).

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

' Find all instances of the notepad window
val = oAutoIt.WinList("[CLASS:Notepad]")
For i = 1 to val(0,0)
  WScript.Echo "Title:" & val(0,i) & " - Handle:" &
 val(1,i)
Next

' Find all windows (may be hundreds!)
val = oAutoIt.WinList("[ALL]")
For i = 1 to val(0,0)
  WScript.Echo "Title:" & val(0,i) & " - Handle:" &
 val(1,i)
Next
```

WinMenuSelectItem

Invokes a menu item of a window

```
WinMenuSelectItem "title", "text", "item" [, "item"
[, "item" [, "item" [, "item" [, "item" [,
"item"]]]]]]
```

Parameters

title	The title of the window to read.
text	The text of the window to read.
item	Text of Menu Item
item	**Optional:** Text of SubMenu item
item	**Optional:** Text of SubMenu item
item	**Optional:** Text of SubMenu item
item	**Optional:** Text of SubMenu item
item	**Optional:** Text of SubMenu item
item	**Optional:** Text of SubMenu item

Return Value

Success: Returns 1.

Failure: Returns 0 if the menu could not be found.

Remarks

You should note that underlined menu items actually contain a & character to indicate the underlining. Thus, the menu item **File** would actually require the text "&File;", and **Convert** would require "Con|" You can access menu items up to six levels deep; and the window can be inactive, minimized, and/or even hidden.

WinMenuSelectItem will only work on standard menus. Unfortunately, many menus in use today are actually custom written or toolbars "pretending" to be menus. This is true for most Microsoft applications.

Related

ControlCommand, Send

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Control")

' This will select File, Page Setup in notepad
oAutoIt.WinMenuSelectItem "Untitled - ", "", "&File", "Page Set&up..."
```

WinMinimizeAll

Minimizes all windows

```
WinMinimizeAll
```

Return Value

None.

Remarks

Send("#m") is a possible alternative.

Related

WinMinimizeAllUndo, WinSetState

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.WinMinimizeAll
```

WinMinimizeAllUndo

Undoes a previous WinMinimizeAll function

```
WinMinimizeAllUndo
```

Return Value

None.

Remarks

Send("#+m") is a possible alternative.

Related

WinMinimizeAll, WinSetState

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.WinMinimizeAllUndo
```

WinMove

Moves and/or resizes a window

```
WinMove "title", "text", x, y [, width [, height]]
```

Parameters

title	The title of the window to move/resize.
text	The text of the window to move/resize.
x	X coordinate to move to.
y	Y coordinate to move to.
width	**Optional:** New width of the window.
height	**Optional:** New height of the window.

Return Value

None.

Remarks

WinMove has no effect on minimized windows, but WinMove works on hidden windows.
If very width and height are small (or negative), the window will go no smaller than 112 x 27 pixels. If width and height are large, the window will go no larger than approximately [12+@DesktopWidth] x [12+@DesktopHeight] pixels.
Negative values are allowed for the x and y coordinates. In fact, you can move a window off screen; and if the window's program is one that remembers its last window position, the window will appear in the corner (but fully on-screen) the next time you launch the program.
If multiple windows match the criteria, the most recently active window is used.

Related

WinActivate, WinClose, WinGetClientSizeHeight, WinGetClientSizeWidth, WinGetPosX, WinGetPosY, WinSetState

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.WinMove "Untitled -", "", 0, 0, 200, 200
```

WinSetOnTop

Change a window's "Always On Top" attribute

```
WinSetOnTop "title", "text", flag
```

Parameters

title	The title of the window to affect.
text	The text of the window to affect.
flag	Determines whether the window should have the "TOPMOST" flag set. 1=set on top flag, 0 = remove on top flag

Return Value

None.

Remarks

Third-party programs which add an "Always On Top" context menu entry might not update their menu entry to reflect the AutoIt-induced change in TOPMOST status.

Related

WinSetState

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")
oAutoIt.WinSetOnTop "Untitled -", "", 1
```

WinSetState

Shows, hides, minimizes, maximizes, or restores a window

```
WinSetState "title", "text", flag
```

Parameters

title	The title of the window to show.
text	The text of the window to show.
flag	The "show" flag of the executed program: SW_HIDE = Hide window SW_SHOW = Shows a previously hidden window SW_MINIMIZE = Minimize window SW_MAXIMIZE = Maximize window SW_RESTORE = Undoes a window minimization or maximization

Return Value

None.

Remarks

WinSetState is a replacement for the badly named WinShow function. WinShow is accepted as an alias but this may be withdrawn in the future.

If multiple windows match the criteria, the most recently active window is used. SW_MINIMIZE and SW_MAXIMIZE even work on modal dialog windows.

Related

WinActivate, WinClose

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

oAutoIt.WinSetState "Untitled -", "", oAutoIt.SW_HI
DE
oAutoIt.Sleep 3000
oAutoIt.WinSetState "Untitled -", "", oAutoIt.SW_SH
OW
```

WinSetTitle

Changes the title of a window

```
WinSetTitle "title", "text", "newtitle"
```

Parameters

title	The title of the window to change.
text	The text of the window to change.
newtitle	The new title to give to the window.

Return Value

None.

Remarks

If multiple windows match the criteria the title of most recently active window is changed.

Related

WinGetTitle, WinTitleMatchMode (Option)

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Control")
oAutoIt.WinSetTitle "Untitled - ", "", "My New Notepad"
```

WinSetTrans

Sets the transparency of a window

```
WinSetTrans "title", "text", transparency
```

Parameters

title	The title of the window to change.
text	The text of the window to change.
transparency	A number in the range 0 - 255. The larger the number, the more transparent the window will become.

Return Value

Non-zero on success, zero on failure. oAutoIt.error will be set to 1 if the function isn't supported on an OS.

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

oAutoIt.Run("notepad.exe", "", oAutoIt.SW_MINIMIZE)
oAutoIt.WinWaitActive("Untitled - Notepad")
oAutoIt.WinSetTrans "Untitled - Notepad","", 50
```

WinWait

Pauses execution of the script until the requested window exists

```
WinWait "title" [, "text" [, timeout]]
```

Parameters

title	The title of the window to check.
text	**Optional:** The text of the window to check.
timeout	**Optional:** Timeout in seconds

Return Value

Success: Returns 1.

Failure: Returns 0 if timeout occurred.

Remarks

The script polls for window match every 250 milliseconds or so.

Related

WinActive, WinExists, WinWait, WinWaitActive, WinWaitClose, WinWaitDelay (Option), WinWaitNotActive

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

'Wait for the window "Untitled" to exist

oAutoIt.Run "notepad.exe"
oAutoIt.WinWait "Untitled"

'Wait a maximum of 5 seconds for "Untitled" to exis
t
oAutoIt.WinWait "Untitled", "", 5
```

WinWaitActive

Pauses execution of the script until the requested window is active

```
WinWaitActive "title", ["text"], [timeout]
```

Parameters

title	The title of the window to check.
text	**Optional:** The text of the window to check.
timeout	**Optional:** Timeout in seconds

Return Value

Success: Returns 1.

Failure: Returns 0 if timeout occurred.

Remarks

AutoIt polls for a window match every 250 milliseconds or so.

Related

WinActive, WinExists, WinWait, WinWaitClose, WinWaitDelay (Option), WinWaitNotActive

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Control")

'Wait for the window "Untitled" to exist and be active
oAutoIt.WinWaitActive "Untitled"

'Wait a maximum of 5 seconds for "Untitled" to exist and be active
oAutoIt.WinWaitActive "Untitled", "", 5
```

WinWaitClose

Pauses execution of the script until the requested window does not exist

```
WinWaitClose "title" [, "text" [, timeout]]
```

Parameters

title	The title of the window to check.
text	**Optional:** The text of the window to check.
timeout	**Optional:** Timeout in seconds

Return Value

Success: Returns 1.

Failure: Returns 0 if timeout occurred.

Remarks

If the window already doesn't exist when this function is called it will return 0 immediately. The window is polled every 250 milliseconds or so.

Related

WinActive, WinExists, WinWait, WinWaitActive, WinWaitDelay (Option), WinWaitNotActive

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

'Wait for the window "Untitled" to not exist
oAutoIt.WinWaitClose "Untitled"

'Wait a maximum of 5 seconds for "Untitled" to not
exist
oAutoIt.WinWaitClose "Untitled", "", 5
```

WinWaitNotActive

Pauses execution of the script until the requested window is not active

```
WinWaitNotActive "title" [, "text" [, timeout]]
```

Parameters

title	The title of the window to check.
text	**Optional:** The text of the window to check.
timeout	**Optional:** Timeout in seconds

Return Value

Returns 0 if the timeout occurred, otherwise returns 1.

Remarks

The script polls for a window match every 250 milliseconds or so.

Related

WinActive, WinExists, WinWait, WinWaitActive, WinWaitClose, WinWaitDelay (Option)

Example

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Contro
l")

'Wait for the window "Untitled" to not be active
oAutoIt.WinWaitNotActive "Untitled"

' Wait a maximum of 5 seconds for "Untitled" to not
 be active
oAutoIt.WinWaitNotActive "Untitled", "", 5
```

Properties Reference

Below is an list of all the properties available in AutoItX.

e.g.

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Control")
error = oAutoIt.error()
```

Macro	Description
error	Status of the error flag (equivalent to the @error macro in AutoIt v3)
version	autoitX dll version (equivalent to @AutoItVersion macro in AutoIt v3)
SW_HIDE	Hides the window and activates another window.
SW_MAXIMIZE	Maximizes the specified window.
SW_MINIMIZE	Minimizes the specified window and activates the next top-level window in the Z order.
SW_RESTORE	Activates and displays the window. If the window is minimized or maximized, the system restores it to its original size and position. An application should specify this flag when restoring a minimized window.
SW_SHOW	Activates the window and displays it in its current size and position.
SW_SHOWDEFAULT	Sets the show state based on the SW_ value specified by the program that started the application.
SW_SHOWMAXIMIZED	Activates the window and displays it as a maximized window.

SW_SHOWMINIMIZED	Activates the window and displays it as a minimized window.
SW_SHOWMINNOACTIVE	Displays the window as a minimized window. This value is similar to SW_SHOWMINIMIZED, except the window is not activated.
SW_SHOWNA	Displays the window in its current size and position. This value is similar to SW_SHOW, except the window is not activated.
SW_SHOWNOACTIVATE	Displays a window in its most recent size and position. This value is similar to SW_SHOWNORMAL, except the window is not actived.
SW_SHOWNORMAL	Activates and displays a window. If the window is minimized or maximized, the system restores it to its original size and position. An application should specify this flag when displaying the window for the first time.

Using the DLL Interface

AutoItX can be used as a standard DLL from any language capable of calling functions in external DLLs.

The following files are provided to allow you to use the DLL in C++:

AutoItX3_DLL.h - C language header file showing the exported functions and parameters

AutoItX3_DLL.lib - Microsoft format import library (x86)

AutoItX3_x64_DLL.lib - Microsoft format import library (x64)

AutoItX3.dll - The main AutoItX DLL (x86)

AutoItX3_x64.dll - The main AutoItX DLL (x64)

Documentation for each function is not provided as the function calls are similar to the COM version and it should be easy to work out what is required with the COM documentation and the header file. The main difference to the COM version is that the functions that **receive** text - such as AU3_WinGetTitle have an additional parameter called **nBufSize** - this should be set to the available size of the output text buffer.

e.g. The prototype for AU3_WinGetText is:

```
void AU3_WinGetTitle(LPCWSTR szTitle, LPCWSTR szText,
```

The return buffer is szRetText, and an example (C++ language) usage would be:

```
WCHAR szMyTitle[200];
AU3_WinGetTitle("Untitled - Notepad", "", szMyTitle, 2
MessageBox(NULL, szMyTitle, "Returned window title was
```

nBufSize is simply used to avoid buffer overruns in any text that is passed back.

Using the .NET Interface

A .NET Assembly interface is provided to allow the use of AutoItX DLL functions without the trouble of working with DllImport. The assembly uses .NET 2.0 / CLR 2.0 and is strongly signed for maximum compatibility with your projects.

The following files are provided to allow .NET use:

AutoItX3.Assembly.dll - The .NET Assembly for using AutoItX.

AutoItX3.Assembly.xml - The Visual Studio Intellisense help file for the .NET Assembly.

AutoItX3.dll - The main AutoItX DLL (x86)

AutoItX3_x64.dll - The main AutoItX DLL (x64)

Using the Assembly from VB/C# within in Visual Studio is very easy:

- Add a reference to **AutoItX3.Assembly.dll** to your project

- Add a **using AutoIt;** statement in the files you want to use AutoIt functions

- Write code like this C# example:

```
using AutoIt;
...

// Wow, this is C#!
AutoItX.Run("notepad.exe");
AutoItX.WinWaitActive("Untitled");
AutoItX.Send("I'm in notepad");
IntPtr winHandle = AutoItX.WinGetHandle("Untitled"
AutoItX.WinKill(winHandle);
```

- Distribute your final executable with the files **AutoItX3.Assembly.dll, AutoItX3.dll, AutoItX3_x64.dll**.

Using the PowerShell CmdLets

AutoItX can be used to provide PowerShell CmdLets.

The following files are provided to allow PowerShell use:

AutoItX.psd1 - The signed PowerShell import manifest, this is the file you will import into your script.

AutoItX3.PowerShell.dll - The PowerShell module code

AutoItX3.Assembly.dll - The .NET Assembly for using AutoItX. This is used by the PowerShell module.

AutoItX3.dll - The main AutoItX DLL (x86)

AutoItX3_x64.dll - The main AutoItX DLL (x64)

Usage of the cmdlets wihin PowerShell is as follows:

```
# Import the module manfiest.
# If AutoIt is installed on this machine this step
Import-Module .\AutoItX.psd1

# Get the list of AutoItX cmdlets
Get-Command *AU3*

# Get detailed help for a particular cmdlet
Get-Help Get-AU3WinText
```

Appendix Reference

ASCII Character Codes

This list is helpful with the Asc and Chr functions. Taken from ASCII Character Set

Control Characters (mostly non-printing; most useful of these are highlighted in yellow.)

Char...	Dec	Hex	Oct	Description
NUL	0	00	000	Null character
SOH	1	01	001	Start of heading, = console interrupt
STX	2	02	002	Start of text, maintenance mode on HP console
ETX	3	03	003	End of text
EOT	4	04	004	End of transmission, not the same as ETB
ENQ	5	05	005	Enquiry, goes with ACK; old HP flow control
ACK	6	06	006	Acknowledge, clears ENQ logon hand
BEL	7	07	007	Bell, rings the bell... (Plays Windows' default beep)
BS	8	08	010	Backspace, works on HP terminals/computers
HT	9	09	011	Horizontal tab, move to next tab stop
LF	10	0a	012	Line Feed
VT	11	0b	013	Vertical tab
FF	12	0c	014	Form Feed, page eject
CR	13	0d	015	Carriage Return
SO	14	0e	016	Shift Out, alternate character set
SI	15	0f	017	Shift In, resume default character set
DLE	16	10	020	Data link escape
DC1	17	11	021	XON, with XOFF to pause listings; ":okay to send".
DC2	18	12	022	Device control 2, block-mode flow control
DC3	19	13	023	XOFF, with XON is TERM=18 flow control

DC4	20	14	024	Device control 4
NAK	21	15	025	Negative acknowledge
SYN	22	16	026	Synchronous idle
ETB	23	17	027	End transmission block, not the same as EOT
CAN	24	18	030	Cancel line, MPE echoes !!!
EM	25	19	031	End of medium, Control-Y interrupt
SUB	26	1a	032	Substitute
ESC	27	1b	033	Escape, next character is not echoed
FS	28	1c	034	File separator
GS	29	1d	035	Group separator
RS	30	1e	036	Record separator, block-mode terminator
US	31	1f	037	Unit separator
DEL	127	7f	177	Delete (rubout), cross-hatch box

Printing Characters (standard characters)

Char	Dec	Hex	Oct	Description
	32	20	040	Space
!	33	21	041	Exclamation mark
"	34	22	042	Quotation mark (" in HTML)
#	35	23	043	Cross hatch (number sign)
$	36	24	044	Dollar sign
%	37	25	045	Percent sign
&	38	26	046	Ampersand
`	39	27	047	Closing single quote (apostrophe)
(	40	28	050	Opening parentheses
)	41	29	051	Closing parentheses

*	42	2a	052	Asterisk (star, multiply)
+	43	2b	053	Plus
,	44	2c	054	Comma
-	45	2d	055	Hyphen, dash, minus
.	46	2e	056	Period
/	47	2f	057	Slant (forward slash, divide)
0	48	30	060	Zero
1	49	31	061	One
2	50	32	062	Two
3	51	33	063	Three
4	52	34	064	Four
5	53	35	065	Five
6	54	36	066	Six
7	55	37	067	Seven
8	56	38	070	Eight
9	57	39	071	Nine
:	58	3a	072	Colon
;	59	3b	073	Semicolon
<	60	3c	074	Less than sign
=	61	3d	075	Equals sign
>	62	3e	076	Greater than sign
?	63	3f	077	Question mark
@	64	40	100	At-sign
A	65	41	101	Uppercase A
B	66	42	102	Uppercase B
C	67	43	103	Uppercase C
D	68	44	104	Uppercase D
E	69	45	105	Uppercase E

F	70	46	106	Uppercase F
G	71	47	107	Uppercase G
H	72	48	110	Uppercase H
I	73	49	111	Uppercase I
J	74	4a	112	Uppercase J
K	75	4b	113	Uppercase K
L	76	4c	114	Uppercase L
M	77	4d	115	Uppercase M
N	78	4e	116	Uppercase N
O	79	4f	117	Uppercase O
P	80	50	120	Uppercase P
Q	81	51	121	Uppercase Q
R	82	52	122	Uppercase R
S	83	53	123	Uppercase S
T	84	54	124	Uppercase T
U	85	55	125	Uppercase U
V	86	56	126	Uppercase V
W	87	57	127	Uppercase W
X	88	58	130	Uppercase X
Y	89	59	131	Uppercase Y
Z	90	5a	132	Uppercase Z
[	91	5b	133	Opening square bracket
\	92	5c	134	Reverse slant (Backslash)
]	93	5d	135	Closing square bracket
^	94	5e	136	Caret (Circumflex)
_	95	5f	137	Underscore
`	96	60	140	Opening single quote
a	97	61	141	Lowercase a

b	98	62	142	Lowercase b
c	99	63	143	Lowercase c
d	100	64	144	Lowercase d
e	101	65	145	Lowercase e
f	102	66	146	Lowercase f
g	103	67	147	Lowercase g
h	104	68	150	Lowercase h
i	105	69	151	Lowercase i
j	106	6a	152	Lowercase j
k	107	6b	153	Lowercase k
l	108	6c	154	Lowercase l
m	109	6d	155	Lowercase m
n	110	6e	156	Lowercase n
o	111	6f	157	Lowercase o
p	112	70	160	Lowercase p
q	113	71	161	Lowercase q
r	114	72	162	Lowercase r
s	115	73	163	Lowercase s
t	116	74	164	Lowercase t
u	117	75	165	Lowercase u
v	118	76	166	Lowercase v
w	119	77	167	Lowercase w
x	120	78	170	Lowercase x
y	121	79	171	Lowercase y
z	122	7a	172	Lowercase z
{	123	7b	173	Opening curly brace
|	124	7c	174	Vertical line
}	125	7d	175	Closing curly brace

~	126	7e	176	Tilde (approximate)

Extended Character Set (ANSI)

Char	Dec	Hex	Oct	Description
€	128	80	200	
	129	81	201	Note: does not display in this compiled html help file
'	130	82	202	
f	131	83	203	
"	132	84	204	
…	133	85	205	
†	134	86	206	
‡	135	87	207	
ˆ	136	88	210	
‰	137	89	211	
Š	138	8A	212	
‹	139	8B	213	
Œ	140	8C	214	
	141	8D	215	Note: does not display in this compiled html help file
Ž	142	8E	216	
	143	8F	217	Note: does not display in this compiled html help file
?	144	90	220	Note: does not display in this compiled html help file
'	145	91	221	
'	146	92	222	
"	147	93	223	
"	148	94	224	
•	149	95	225	

-	150	96	226	
-	151	97	227	
˜	152	98	230	
™	153	99	231	
š	154	9A	232	
›	155	9B	233	
œ	156	9C	234	
	157	9D	235	Note: does not display in this compiled html help file
ž	158	9E	236	
Ÿ	159	9F	237	
	160	A0	240	
¡	161	A1	241	
¢	162	A2	242	
£	163	A3	243	
¤	164	A4	244	
¥	165	A5	245	
¦	166	A6	246	
§	167	A7	247	
¨	168	A8	250	
©	169	A9	251	
ª	170	AA	252	
«	171	AB	253	
¬	172	AC	254	
	173	AD	255	Note: may not display in this compiled html help file
®	174	AE	256	
¯	175	AF	257	
°	176	B0	260	
±	177	B1	261	

²	178	B2	262	
³	179	B3	263	
´	180	B4	264	
µ	181	B5	265	
¶	182	B6	266	
·	183	B7	267	
¸	184	B8	270	
¹	185	B9	271	
º	186	BA	272	
»	187	BB	273	
¼	188	BC	274	
½	189	BD	275	
¾	190	BE	276	
¿	191	BF	277	
À	192	C0	300	
Á	193	C1	301	
Â	194	C2	302	
Ã	195	C3	303	
Ä	196	C4	304	
Å	197	C5	305	
Æ	198	C6	306	
Ç	199	C7	307	
È	200	C8	310	
É	201	C9	311	
Ê	202	CA	312	
Ë	203	CB	313	
Ì	204	CC	314	
Í	205	CD	315	

Î	206	CE	316	
Ï	207	CF	317	
Đ	208	D0	320	
Ñ	209	D1	321	
Ò	210	D2	322	
Ó	211	D3	323	
Ô	212	D4	324	
Õ	213	D5	325	
Ö	214	D6	326	
×	215	D7	327	
Ø	216	D8	330	
Ù	217	D9	331	
Ú	218	DA	332	
Û	219	DB	333	
Ü	220	DC	334	
Ý	221	DD	335	
Þ	222	DE	336	
ß	223	DF	337	
à	224	E0	340	
á	225	E1	341	
â	226	E2	342	
ã	227	E3	343	
ä	228	E4	344	
å	229	E5	345	
æ	230	E6	346	
ç	231	E7	347	
è	232	E8	350	

é	233	E9	351	
ê	234	EA	352	
ë	235	EB	353	
ì	236	EC	354	
í	237	ED	355	
î	238	EE	356	
ï	239	EF	357	
ð	240	F0	360	
ñ	241	F1	361	
ò	242	F2	362	
ó	243	F3	363	
ô	244	F4	364	
õ	245	F5	365	
ö	246	F6	366	
÷	247	F7	367	
ø	248	F8	370	
ù	249	F9	371	
ú	250	FA	372	
û	251	FB	373	
ü	252	FC	374	
ý	253	FD	375	
þ	254	FE	376	
ÿ	255	FF	377	

Send Key list

Quick reference for the **Send("keys" [, flag])** Command. ^ Ctrl ! Alt + Shift # Win

AutoIt can send all ASCII and Extended ASCII characters (0-255), to send UNICODE characters you must use the "ASC" option and the code of the character you wish to Send(see {ASC} below).

To send the ASCII value A (same as pressing ALT+065 on the numeric keypad)

```
Send("{ASC 065}")
```

(When using 2 digit ASCII codes you must use a leading 0, otherwise an obsolete 437 code page is used).

To send UNICODE characters enter the character code, for example this sends a Chinese character

```
Send("{ASC 2709}")
```

Single keys can also be repeated, e.g.

```
Send("{DEL 4}") ; Presses the DEL key 4 times
Send("{S 30}") ; Sends 30 'S' characters
Send("+{TAB 4}") ; Presses Shift + Tab 4 times
```

To hold a key down

```
Send("{a down}") ; Holds the A key down
Send("{a up}") ; Releases the A key
```

If you with to use a variable for the count, try

```
Local $iCount = 4
Send("+{TAB " & $iCount & "}")
```

If you wish to send the ASCII value A four times, then try

```
Local $iChr = Chr(65)
Send("{" & $iChr & " 4}")
```

Most laptop computer keyboards have a special Fn key. This key cannot be simulated.

Note, by setting the flag parameter to 1 the "keys" parameter is sent RAW. This is useful when you want to send some text copied from a variable and you want the text sent exactly as written.

Send is quite useful because windows can be navigated without needing a mouse.

For example, open Folder Options (in the control panel) and try the following:

Send("{TAB}")	Navigate to next control (button, checkbox, etc)
Send("+{TAB}")	Navigate to previous control.
Send("^{TAB}")	Navigate to next WindowTab (on a Tabbed dialog window)
Send("^+{TAB}")	Navigate to previous WindowTab.
Send("{SPACE}")	Can be used to toggle a checkbox or click a button.
Send("{+}")	Usually checks a checkbox (if it's a "real" checkbox.)
Send("{-}")	Usually unchecks a checkbox.

Send("{NumPadMult}")	Recursively expands folders in a SysTreeView32.

Use Alt-key combos to access menu items. Also, open Notepad and try the following:

Send("!f")	Send Alt+f, the access key for Notepad's file menu. *Try other letters!*
Send("{DOWN}")	Move down a menu.
Send("{UP}")	Move up a menu.
Send("{LEFT}")	Move leftward to new menu or expand a submenu.
Send("{RIGHT}")	Move rightward to new menu or collapse a submenu.

See Windows' Help--press *Win+F1*--for a complete list of keyboard shortcuts if you don't know the importance of Alt+F4, PrintScreen, Ctrl+C, and so on.

Send Command (if zero flag)	Resulting Keypress
{!}	!
{#}	#
{+}	+
{^}	^
{{}	{
{}}	}
{SPACE}	SPACE
{ENTER}	ENTER key on the main keyboard

{ALT}	ALT
{BACKSPACE} or {BS}	BACKSPACE
{DELETE} or {DEL}	DELETE
{UP}	Up arrow
{DOWN}	Down arrow
{LEFT}	Left arrow
{RIGHT}	Right arrow
{HOME}	HOME
{END}	END
{ESCAPE} or {ESC}	ESCAPE
{INSERT} or {INS}	INS
{PGUP}	PGUP
{PGDN}	PGDN
{F1} - {F12}	Function keys
{TAB}	TAB
{PRINTSCREEN}	PRINTSCR
{LWIN}	Left Windows key
{RWIN}	Right Windows key
{NUMLOCK}	NUMLOCK
{CAPSLOCK}	CAPSLOCK
{SCROLLLOCK}	SCROLLLOCK
{BREAK}	for Ctrl+Break processing
{PAUSE}	PAUSE
{NUMPAD0} - {NUMPAD9}	Numpad digits
{NUMPADMULT}	Numpad Multiply
{NUMPADADD}	Numpad Add
{NUMPADSUB}	Numpad Subtract

{NUMPADDIV}	Numpad Divide
{NUMPADDOT}	Numpad period
{NUMPADENTER}	Enter key on the numpad
{APPSKEY}	Windows App key
{LALT}	Left ALT key
{RALT}	Right ALT key
{LCTRL}	Left CTRL key
{RCTRL}	Right CTRL key
{LSHIFT}	Left Shift key
{RSHIFT}	Right Shift key
{SLEEP}	Computer SLEEP key
{ALTDOWN}	Holds the ALT key down until {ALTUP} is sent
{SHIFTDOWN}	Holds the SHIFT key down until {SHIFTUP} is sent
{CTRLDOWN}	Holds the CTRL key down until {CTRLUP} is sent
{LWINDOWN}	Holds the left Windows key down until {LWINUP} is sent
{RWINDOWN}	Holds the right Windows key down until {RWINUP} is sent
{ASC nnnn}	Send the ALT+nnnn key combination
{BROWSER_BACK}	Select the browser "back" button
{BROWSER_FORWARD}	Select the browser "forward" button
{BROWSER_REFRESH}	Select the browser "refresh" button
{BROWSER_STOP}	Select the browser "stop" button
{BROWSER_SEARCH}	Select the browser "search" button
{BROWSER_FAVORITES}	Select the browser "favorites" button
{BROWSER_HOME}	Launch the browser and go to the home page

{VOLUME_MUTE}	Mute the volume
{VOLUME_DOWN}	Reduce the volume
{VOLUME_UP}	Increase the volume
{MEDIA_NEXT}	Select next track in media player
{MEDIA_PREV}	Select previous track in media player
{MEDIA_STOP}	Stop media player
{MEDIA_PLAY_PAUSE}	Play/pause media player
{LAUNCH_MAIL}	Launch the email application
{LAUNCH_MEDIA}	Launch media player
{LAUNCH_APP1}	Launch user app1
{LAUNCH_APP2}	Launch user app2

Credits

AutoItX was adapted from the main AutoIt source code by Jonathan Bennett.
Please see the main AutoIt help file for full credits.

History

IMPORTANT: See here for recent script-breaking changes.

Please see the main AutoIt helpfile which details the history of all components including AutoItX.

Script Breaking Changes in Recent Versions

This page contains a list of all changes made in recent updates that will almost certainly **break existing scripts/sources**. Please read this list carefully when upgrading if you have not been keeping up with developments in the beta versions. If any of these issues affect you then **you will need to modify your scripts/sources**.

23rd December, 2013 - v3.3.10.0

AutoItX:

- Removed: Removed DevC files and Visual Studio 6 files.
- Removed: ANSI versions of Send/WinWait functions from the native DLL.
- Removed: DLL Imports: AU3_WinGetPosX, AU3_WinGetPosY, AU3_WinGetPosWidth, AU3_WinGetPosHeight.
- Removed: DLL Imports: AU3_ControlGetPosX, AU3_ControlGetPosY, AU3_ControlGetPosWidth, AU3_ControlGetPosHeight.
- Removed: DLL Imports: AU3_MousePosX, AU3_MouseGetPosY.
- Removed: DLL Imports: AU3_WinGetClientSizeWidth, AU3_WinGetClientSizeHeight.
- Removed: DLL Imports: AU3_WinGetCaretPosX, AU3_WinGetCaretPosY.
- Removed: DLL Imports: AU3_CDTray, AU3_BlockInput.
- Removed: DLL Imports: AU3_RunAsSet.
- Removed: COM Methods: CDTray, BlockInput.
- Removed: COM Methods: RunAsSet.
- Removed: All registry functions. The support was limited and the host language will certainly have registry functions.
- Removed: All Ini file functions. As per registry functions.
- Changed: Renamed AutoIt3.h to AutoItX3_DLL.h.

18th December, 2009 - v3.3.2.0

AutoItX:

- Removed: "ColorMode" option removed from AutoItSetOption().

24th December, 2008 - v3.3.0.0

AutoItX:

- The native version of the DLL now exclusively uses Unicode strings (LPWSTR and LPCWTSR). If you really need ANSI strings then continue to use v3.2.12.1. The easiest workaround though is to use a "wrapper" function to simply convert between Unicode and ANSI. A limited selection of functions (WinWait... and Send) have an ANSI version but it is not intended to add ANSI versions for any others unless there is a large demand.